Fortinet, Inc.

Fortigate 5.6

Assurance Activity Report

Version 1.3

May 2019

Document prepared by

www.lightshipsec.com

Table of Contents

1 Introduction

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

1.1 Evaluation Identifiers

Table 1: Evaluation Identifiers

Scheme	Canadian Common Criteria Scheme
Evaluation Facility	Lightship Security
Developer/Sponsor	Fortinet, Inc.
TOE	Fortigate 5.6
Security Target	FortiGate/FortiOS 5.6 Security Target, v1.3, May 2019
Protection Profile	collaborative Protection Profile for Stateful Traffic Filter Firewalls, v2.0e+20180314 (FWcPP) Network Device Collaborative Protection Profile (NDcPP)/Stateful Traffic Filter Firewall Collaborative Protection Profile (FWcPP) Extended Package VPN Gateway, v2.1, 2017-03-08 collaborative Protection Profile for Network Devices/collaborative Protection Profile for Stateful Traffic Filter Firewalls Extended Package (EP) for Intrusion Prevention Systems (IPS), v2.11, 15 June 2017

1.2 Evaluation Methods

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

Table 2: Evaluation Methods

Evaluation Criteria	CC v3.1R4
Evaluation Methodology	CEM v3.1R4
Supporting Documents	Evaluation Activities for Stateful Traffic Filter Firewalls cPP, v2.0e+20180314 (FWcPP-SD)

1.3 Reference Documents

Table 3: List of Reference Documents

Ref	Evidence
[ST]	FortiGate/FortiOS 5.6 Security Target, v1.3, May 2019
[FNLOG]	FortiOS 5.6.7 Log Reference, November 27, 2018, 01-565-414447-20181127
[ADMIN]	FortiOS Handbook, February 19, 2019, 01-567-497911-20190219
[CLI]	FortiOS Handbook - CLI Reference, January 31, 2019, 01-567-498240-20190131
[IPS]	Fortinet IPS Signature Syntax Guide, 00-108-229429-20140522.
[SUPP]	FortiOS 5.6 and FortiGate NGFW Appliances FIPS140-2 and Common Criteria Technote, Doc No. 01-567-535352-20190122

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:	This information was found in the ST TSS in section 6.1. Actions which can affect private cryptographic keys include generating a CSR (which implicitly includes a private key).

4 For distributed TOEs the evaluator shall examine the TSS to ensure it describes which auditable events are generated and recorded by which TOE components. The evaluator shall confirm that all components defined as generating audit information for a particular SFR should also contribute to that SFR as defined in the mapping of SFRs to TOE components, and that the audit records generated by each component cover all the SFRs that it implements.

Findings:	The TOE is not a distributed TOE.

2.1.1.2 Guidance Documentation

Technical Decisions:	The following assurance activities have been modified by TD0410.

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:	Audit events and format are presented in the [FNLOG] document. There are extensive samples of the expected audit messages provided.

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.

Findings:	These tests are conducted throughout the test plan.

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

Findings:	The TOE is not a distributed TOE.

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

2.1.2 FAU_GEN.1 Audit data generation (VPN GW EP)

2.1.2.1 TSS

10 The evaluator shall verify that the TSS describes how the TSF can be configured to log network traffic associated with applicable rules. Note that this activity should have been addressed with a combination of the TSS assurance activities for FPF_RUL_EXT.1.

Findings:	Addressed by TSS assurance activities for FPF_RUL_EXT.1.

11 The evaluator shall verify that the TSS describes how the TOE behaves when one of its interfaces is overwhelmed by network traffic. It is acceptable for the TOE to drop

packets that it cannot process, but under no circumstances is the TOE allowed to pass packets that do not satisfy a rule that allows the permit operation or belong to an allowed established session. It may not always be possible for the TOE to audit dropped packets due to implementation limitations. These limitations and circumstances in which the event of dropped packets is not audited shall be described in the TSS.

Findings:	This information is found in section 6.10 of the ST TSS. The TOE will drop and attempt to log packets.

2.1.2.2 Guidance Documentation

Technical Decisions:	The following assurance activities have been modified by TD0248.

12 The evaluator shall verify that the operational guidance describes how to configure the TSF to result in applicable network traffic logging. Note that this activity should have been addressed with a combination of the guidance assurance activities for FPF_RUL_EXT.1.

Findings:	Addressed by guidance activities for FPF_RUL_EXT.1.

13 The evaluator shall verify that the TSS describes how the TOE behaves when one of its interfaces is overwhelmed by network traffic. It is acceptable for the TOE to drop packets that it cannot process, but under no circumstances is the TOE allowed to pass packets that do not satisfy a rule that allows the permit operation or belong to an allowed established session. It may not always be possible for the TOE to audit dropped packets due to implementation limitations. These limitations and circumstances in which the event of dropped packets is not audited shall be described in the TSS.

Findings:	Addressed by TSS above.

2.1.2.3 Tests

14 The following test is expected to execute outside the context of the other requirements. While testing the TOE's compliance against the SFRs, either specific tests are developed and run in the context of this SFR, or as is typically done, the audit capability is turned on while testing the TOE's behavior in complying with the other SFRs in this EP.

15 Test 1: The evaluator shall attempt to flood the TOE with network packets such that the TOE will be unable to process all the packets. This may require the evaluator to configure the TOE to limit the bandwidth the TOE is capable to handling (e.g., use of a 10 MB interface). The evaluator shall then review the audit logs to verify that the TOE correctly records that it is unable to process all of the received packets and verify that the TOE logging behavior is consistent with the TSS.

High-Level Test Description
The evaluator limits the bandwidth capable on the device and shows that when the bandwidth is exceeded an audit message is emitted showing that the TOE was unable to process all of the packets.
Findings: PASS

2.1.3 FAU_GEN.1/IPS Audit Data Generation (IPS EP)

2.1.3.1 TSS

16 The evaluator shall verify that the TSS describes how the TOE can be configured to log IPS data associated with applicable policies.

Findings:	IPS logging can be configured as part of each policy and needs to be enabled per rule/signature as per section 6.14 of the ST TSS.

17 The evaluator shall verify that the TSS describes what (similar) IPS event types the TOE will combine into a single audit record along with the conditions (e.g., thresholds and time periods) for so doing. The TSS shall also describe to what extent (if any) that may be configurable.

Findings:	Section 6.14 describes that groups are based on 5-second time periods. It is not claimed to be configurable.

18 For IPS_SBD_EXT.1, for each field, the evaluator shall verify that the TSS describes how the field is inspected and if logging is not applicable, any other mechanism such as counting that is deployed.

Findings:	In section 6.14 of the ST TSS, each of the fields described in IPS_SBD_EXT.1.1 can be inspected. Logging is applicable.

2.1.3.2 Guidance Documentation

19 The evaluator shall verify that the operational guidance describes how to configure the TOE to result in applicable IPS data logging.

Findings:	No additional configuration was needed to log the necessary information. When the administrator needs to capture packet data, this can be done by configuring the IPS rule to enable “packet-capture” specifically for that rule. Packet captures are both stored locally as well as being transmitted automatically to the remote logging server.

20 The evaluator shall verify that the operational guidance provides instructions for any configuration that may be done in regard to logging similar events (e.g., setting thresholds, defining time windows, etc.).

Findings:	There are no configurable options to modify how similar event logging is defined.

2.1.3.3 Test

21 Test 1: The evaluator shall test that the interfaces used to configure the IPS polices yield expected IPS data in association with the IPS policies. A number of IPS policy combination and ordering scenarios need to be configured and tested by attempting to pass both allowed and anomalous network traffic matching configured IPS policies in order to trigger all required IPS events. Note that this activity should have been addressed with a combination of the Test assurance activities for the other IPS requirements.

Findings:	These tests are conducted throughout the test plan.

2.1.4 FAU_GEN.2 User identity association

2.1.4.1 Tests

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

23 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.

2.1.5 FAU_STG_EXT.1 Protected audit event storage

2.1.5.1 TSS

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

Findings:	This information was found in section 6.1 of the ST and states that audit data is transferred to a Fortinet FortiAnalyzer device via TLS.

25 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:	This information was found in section 6.1 of the ST. Local log capacities are dependent on the hardware model and the storage characteristics which are presented in Table 4: TOE Hardware Modles in section 2.4 of the ST.

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

Findings:	The TOE claims that it will overwrite the oldest records as detailed in section 6.1 of the ST when the local log capacity has been filled.

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

Findings:	Section 6.1 of the ST specifies that the TOE transmits the audit information immediately.

28 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.

29 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.

2.1.5.2 Guidance Documentation

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

Findings:	The TOE is required to communicate with a FortiAnalyzer logging device. This information is found in the [SUPP] in the sub-sections under “Log Specific Settings”. The FortiAnalyzer communicates over TLS. The configuration of the logging server communication details are found in the [SUPP] and [ADMIN] guidance documents. The evaluator was able to configure the logging server using the provided guides.

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

Findings:	The [SUPP] in the sub-sections under “Log Specific Settings” describes the relationship between local and remote logs. The [SUPP] characterizes the local logs as being “cached” before being transmitted to the remote logging server. In the [ADMIN] document, this relationship is expanded upon when describing the specific configuration items. The TOE is capable of caching for a short period of time (eg. 1 minute or 5 minutes) or transmitting in real-time. This is done using the “upload-option” setting in the CLI.

32 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.

2.1.5.3 Tests

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

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

Findings	Verification that the data is encrypted is satisfied by FTP_ITC.1 for the logging channel. The logging server is a FortiAnalyzer 200D running v5.6.0-build1671 180503 (Interim) as described in the Test Setup. The evaluator witnessed logging events being received by the remote logging server without intervention.

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

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

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

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

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

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

High-Level Test Description
Log into the device and show that log entries are stored on the local system via the “Forward Traffic” and “Local Traffic” panels. Using a debugging interface, fill up the log disk that stores the traffic event logs. Show that creating more log entries erases the oldest log files.
Findings: PASS

High-Level Test Description
Log into the device and show that log entries are stored on the local system via the “Forward Traffic” and “Local Traffic” panels and that they are sourced from the memory log instead of the disk log. Ensure that once the log is filled, the oldest logs are dropped off.
Findings: PASS

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

Findings:	The TOE does not claim this functionality.

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

Findings:	The TOE is not a distributed TOE.

2.2 Cryptographic Support (FCS)

2.2.1 Algorithm Validations

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

Module Name	CAVP Cert	Services in the scope of the evaluation
Fortinet FortiOS FIPS Cryptographic Library v5.6	C468	Password hashing IKE
Fortinet FortiOS SSL[1] Cryptographic Library v5.6	C530	TLS SSH IPsec Trusted update signature verification
Fortinet FortiOS RBG Cryptographic Library v5.6	C529	DRBG
Fortinet CP8 Cryptographic Library v5.6	C469	Hardware acceleration of IPsec/IKE, TLS and SSH related cryptographic operations if supported and the CP8 module is present.
Fortinet CP9 Cryptographic Library v5.6	C531	Hardware acceleration of IPsec/IKE, TLS and SSH related cryptographic operations if supported and the CP9 module is present.

Findings: The evaluators considered all of the certificates and found evidence that each of the claimed processor architectures are present for each of the claimed algorithms and cryptographic services. This work was supported by a spreadsheet to help keep track of the certificates, architectures and claimed module.

The way the hardware acceleration of the (ARM SoC3), CP8 and CP9 ASICs work is if the algorithm is implemented by the ASIC and the functionality is enabled by administrative configuration, then the ASIC will perform the cryptographic function. If the algorithm is not implemented by the ASIC or if the functionality is disabled by administrative configuration, then the firmware implementation will perform the cryptographic function.

[1] Note that the name 'FortiOS SSL Cryptographic Library' is not restricted to only TLS protocol support. It is a general-purpose cryptographic library used by multiple components.

2.2.2 FCS_CKM.1 Cryptographic Key Generation

2.2.2.1 TSS

35 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:	Table 16 in section 6.2 of the ST shows the key generation characteristics for all claimed key generation schemes.

2.2.2.2 Guidance Documentation

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

Findings:	The TOE's cryptographic properties can be configured for IPSec VPNs, and TLS server trusted path. IPSec VPNs are configured as per the VPN configuration items as described in the [ADMIN] document in Chapter 16. The TLS server trusted path can have the Diffie-Hellman parameters configured as per the [CLI] documentation under 'system global; set dh-params …'. TLS channels to the FortiAnalyzer and SSH trusted path cryptographic characteristics are not modifiable by the user.

2.2.2.3 Tests

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

Key Generation for FIPS PUB 186-4 RSA Schemes

38 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*.

39 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

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

implementation by comparing values generated by the TSF with those generated from a known good implementation.

Findings:	CAVP certificate numbers are described in section 2.2.1. For RSA key generation, this is validated by CAVP C530.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

41 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

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

Findings:	CAVP certificate numbers are described in section 2.2.1. For ECC key generation, this is validated by CAVP C530. If a model contains a CP9 or CP9-Lite ASIC and it is enabled, then ECC key generation can be accelerated and validated by CAVP C531 or C610, respectively.

Key Generation for Finite-Field Cryptography (FFC)

43 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.

44 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

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

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

46 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.

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

48 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.

49 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

50 for each FFC parameter set and key pair.

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

Technical Decisions:	The following assurance activities have been modified by TD0291.

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

2.2.3 FCS_CKM.1/IKE Cryptographic Key Generation (for IKE Peer Authentication) (VPN GW EP)

2.2.3.1 TSS

52 The evaluator shall check to ensure that the TSS describes how the key-pairs are generated.

Findings:	IKE RSA and ECDSA keys are generated via a CSR or via a direct import as described in section 6.2.2 of the ST.

53 In order to show that the TSF implementation complies with FIPS PUB 186-4, the evaluator shall ensure that the TSS contains the following information:

- The TSS shall list all sections of Appendix B to which the TOE complies.
- For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not"), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;
- For each applicable section of Appendix B, any omission of functionality related to "shall" or "should" statements shall be described;

Findings:	In section 6.2 of the ST, this information is presented in table 16. The TOE complies with all 'shall' and 'should' statements and does not implement any 'should not' or 'shall not' statements. Those implementations of 'should' statements are clarified.

54 Any TOE-specific extensions, processing that is not included in the Appendices, or alternative implementations allowed by the Appendices that may impact the security requirements the TOE is to enforce shall be described.

Findings:	No such specific extensions have been claimed; no alternative implementations are claimed.

2.2.3.2 Guidance Documentation

55 The evaluator shall check that the operational guidance describes how the key generation functionality is invoked, and describes the inputs and outputs associated with the process for each signature scheme supported. The evaluator shall also check that guidance is provided regarding the format and location of the output of the key generation process.

Findings:	In [ADMIN] under Chapter 3 for 'Certificate-based Authentication', the vendor documentation provides information on how to generate a CSR necessary to authenticate to the VPN peer. Further, Chapter 16 of the [ADMIN] document describes the process for configuring the IKEv1 and IKEv2 to use the TOE's certificate. [ADMIN] describes the CSR format as RFC 2986 which is a well-known interoperable standard for X.509 certificate generation. In addition to on-board CSR generation, the TOE is capable of importing certificate pairs from the environment. The process is described in Chapter 3 of [ADMIN] for both web-based GUI and CLI.

2.2.3.3 Tests

56 The evaluator shall use the key pair generation portions of "The FIPS 186-4 Elliptic Curve Digital Signature Algorithm Validation System (ECDSA2VS)" and "The RSA Validation System (RSA2VS)" as a guide in testing the requirement above, depending on the selection performed by the ST author. This will require that the evaluator have a trusted reference implementation of the algorithms that can produce test vectors that are verifiable during the test.

Findings:	The TOE uses its CAVP-validated RSA and ECDSA key generator to satisfy key generation. The associated CAVP certificate numbers are described in section 2.2.1 above.

2.2.4 FCS_CKM.2 Cryptographic Key Establishment

2.2.4.1 TSS

57 The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme (including whether the TOE acts as a sender, a recipient, or both). If Diffie-Hellman group 14 is selected from FCS_CKM.2.1, the TSS shall describe how the implementation meets RFC 3526 Section 3.

Findings:	Table 17 in the ST in section 6.2 illustrate the various key establishment schemes. These are consistent with the key generation mechanisms described in the same section. To be clear, Diffie-Hellman group 14 is a finite-field based scheme. DH group 14 is selected from FCS_CKM.2 and the TSS in section 6.2 indicates that the TOE implements 2048-bit MODP group in RFC3526, section 3.

2.2.4.2 Guidance Documentation

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

Findings:	The TOE permits the user to configure DH groups only for IPSec VPN channels. IPSec VPNs are configured as per the VPN configuration items as described in the [ADMIN] document in Chapter 16. TLS and SSH trusted paths for management and TLS trusted channels to the FortiAnalyzer are not modifiable by the user.

2.2.4.3 Tests

Key Establishment Schemes

59 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

60 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

61 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.

62 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.

63 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.

64 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.

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

Validity Test

66 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.

67 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).

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

Findings:	CAVP certificate numbers are described in section 2.2.1. For elliptic-key based key exchange, this is validated as per CAVP C530 (KAS-ECC Component).

RSA-based key establishment schemes

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

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

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

Diffie-Hellman Group 14

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

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

2.2.5 FCS_CKM.4 Cryptographic Key Destruction

2.2.5.1 TSS

71 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[2]). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

Findings:	All relevant keys are described in table 19 in section 6.2.2 which include their origin and their storage location. Keys live in both persistent Flash as well as in RAM. RAM-based keys are plaintext and a handful of keys are – as indicated – encrypted in the Flash memory. The Flash encrypting key is called the "Configuration Encryption Key" which is persistently stored in plaintext in the Flash memory. Table 19 describes all relevant keys. The TOE claims cryptographic channels covering TLS for trusted channels, IPSec for VPN gateway functionality and SSH for secure management. The TOE would be required to persistently store private keys and X.509 public key certificates when acting as a server/peer in SSH, TLS and IPSec capacities which is consistent with the given table. The various session keys are consistent with the protocols.

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

Findings:	The mechanism by which the TOE destroys plaintext keys in non-volatile memory is described in the ST in section 6.2 – which is via an OS kernel call using a single-pass overwrite of zeros followed by a read-verify.

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

Findings:	The TOE claims in FCS_CKM.4 that for non-volatile memory, zeroization occurs via an invocation of an interface. The interface is described in ST section 6.2 as an OS kernel call which is consistent with what would be used for non-volatile storage media.

[2] 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.

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

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

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

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

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

Findings:	The TOE does not claim this selection.

2.2.5.2 Guidance Documentation

77 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.

78 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[3] 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.

[3] 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).

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

2.2.6.1 Tests

AES-CBC Known Answer Tests

79 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.

80 **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.

81 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.

82 **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.

83 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.

84 **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].

85 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.

86 **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].

87 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

88 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.

89 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

90 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]
```

91 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.

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

Findings:	CAVP certificate numbers are described in section 2.2.1. AES-CBC cryptographic operations are validated under CAVP C530. If a model contains a CP8, CP9, or CP9-Lite ASIC and it is enabled, then this hardware acceleration for AES-CBC is validated by CAVP C469, CAVP C531, or CAVP C610, respectively.

AES-GCM Test

93 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.

94 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.

95 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.

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

Findings: CAVP certificate numbers are described in section 2.2.1. AES-GCM cryptographic operations are validated under CAVP C530. If a model contains a CP9, or CP9-Lite ASIC and it is enabled, then this hardware acceleration for AES-GCM is validated by CAVP C531 or CAVP C610, respectively.

AES-CTR Known Answer Tests

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

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

98 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.

99 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.

100 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.

101 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].

102 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

103 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

104 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]
```

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

Findings:	The TOE does not claim AES-CTR mode ciphers.

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

2.2.7.1 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

106 For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message

a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

Findings:	CAVP certificate numbers are described in section 2.2.1. ECDSA SigGen operations are validated under CAVP C530. If a model contains a CP9 or CP9-Lite ASIC and it is enabled, then this hardware acceleration for ECDSA SigGen is validated by CAVP C531 or CAVP C610, respectively.

ECDSA FIPS 186-4 Signature Verification Test

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

Findings:	CAVP certificate numbers are described in section 2.2.1. ECDSA SigVer operations are validated under CAVP C530. If a model contains a CP9 or CP9-Lite ASIC and it is enabled, then this hardware acceleration for ECDSA SigVer is validated by CAVP C531 or CAVP C610, respectively.

RSA Signature Algorithm Tests

Signature Generation Test

108 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.

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

Findings:	CAVP certificate numbers are described in section 2.2.1. RSA SigGen operations are validated under CAVP C530. If a model contains a CP8, CP9 or CP9-Lite ASIC and it is enabled, then this hardware acceleration for RSA SigGen is validated by CAVP C469, CAVP C531, or CAVP C610, respectively.

Signature Verification Test

110 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.

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

Findings:	CAVP certificate numbers are described in section 2.2.1. RSA SigVer operations are validated under CAVP C530. If a model contains a CP8, CP9, or CP9-Lite ASIC and

	it is enabled, then this hardware acceleration for RSA SigVer is validated by CAVP C469, CAVP C531, or CAVP C610, respectively.

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

2.2.8.1 TSS

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

Findings:	This information is documented in section 6.2.1 of the ST.

2.2.8.2 Guidance Documentation

113 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.

2.2.8.3 Tests

114 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.

115 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

116 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

117 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

118 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

119 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

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

Findings:	CAVP certificate numbers are described in section 2.2.1. Secure hashing services are validated under CAVP C530. If a model contains a CP8 ASIC and it is enabled, then hardware acceleration for SHA1 and SHA2-256 is validated by CAVP C469. If a model contains a CP9 or CP9-Lite ASIC and it is enabled, then hardware acceleration for SHA1, SHA2-256, 384 and 512 are validated by CAVP C531 or CAVP C610, respectively.

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

2.2.9.1 TSS

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

Findings:	The key length, block size, hash function and output MAC length are found in table 18 of the ST.

2.2.9.2 Tests

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

Findings:	CAVP certificate numbers are described in section 2.2.1. Secure hashing services are validated under CAVP C530. If a model contains a CP8 ASIC and it is enabled, then hardware acceleration for HMAC-SHA1 and HMAC-SHA2-256 is validated by CAVP C469. If a model contains a CP9 or CP9-Lite ASIC and it is enabled, then hardware acceleration for HMAC-SHA1, HMAC-SHA2-256, 384 and 512 are validated by CAVP C531 or CAVP C610, respectively.

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

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

2.2.10.1 TSS

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

Findings:	The DRBG type has been documented in table 18 in the ST. Details about the seeding mechanism, assumed min-entropy and noise sources are provided in section 6.2.4 of the ST.

2.2.10.2 Guidance Documentation

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

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

2.2.10.3 Tests

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

127 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).

128 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.

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

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

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

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

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

Findings:	CAVP certificate numbers are described in section 2.2.1. CAVP certificate number C529 covers all claimed platforms for the CTR-DRBG.

2.3 User Data Protection (FDP)

2.3.1 FDP_RIP.2 Full Residual Information Protection

2.3.1.1 TSS

130 “Resources” in the context of this requirement are network packets being sent through (as opposed to “to”, as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs.

Findings:	The ST describes in section 6.6 that network packet buffers are zeroized prior to re-use.

2.4 Firewall (FFW)

2.4.1 FFW_RUL_EXT.1 Stateful Traffic Filtering

2.4.1.1 TSS

131 The evaluator shall verify that the TSS provides a description of the TOE's initialization/startup process, which clearly indicates where processing of network packets begins to take place, and provides a discussion that supports the assertion that packets cannot flow during this process.

Findings:	In section 6.10.1, the ST describes the initialization process. Firewall rules are loaded after a series of cryptographic and TOE self-tests and before the network is transitioned to a link-up state. Without the link being in an up state, no packets can flow.

132 The evaluator shall verify that the TSS also include a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describe the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets.

Findings:	Section 6.13 of the ST TSS provides an overview of the processing flow and how abnormal circumstances result in the TOE fails to a secure (ie. closed) state. This information was found to be consistent with Chapter 22 of the [ADMIN] "Parallel Path Processing - Life of a Packet".

2.4.1.2 Guidance

133 The guidance documentation associated with this requirement is assessed in the subsequent test assurance activities.

2.4.1.3 Tests

134 Test 1: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would otherwise be denied by the ruleset should be sourced and be directed at a host. The evaluator shall verify using a packet sniffer that none of the generated network traffic is permitted through the firewall during initialization.

High-Level Test Description
Ensure that all traffic rules have been cleared out and that the default state of the TOE is to filter all traffic. Power the TOE down. Start a wireshark capture on the outside workstation. Using the inside workstation, send a steady stream of TCP packets to the outside network workstation. Power on the TOE and wait for it to be completely initialized. Verify on the wireshark capture that no traffic from the inside workstation has penetrated the TOE.
Findings: PASS

135 Test 2: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would be permitted by the ruleset should be sourced and be directed at a host. The evaluator shall verify using a packet sniffer that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

High-Level Test Description
Ensure that all traffic rules have been cleared out except for a single rule that permits any traffic to be sent to the outside network. Power the TOE down. Start a wireshark capture on the outside workstation. Using the inside workstation, send a steady stream of packets to the outside workstation. Power on the TOE and wait for it to be completely initialized. Verify on the wireshark capture that no traffic from the inside workstation has penetrated the TOE until after initialization.
Findings: PASS

136 Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test assurance activities.

2.4.2 FFW_RUL_EXT.1.2/FFW_RUL_EXT.1.3/FFW_RUL_EXT.1.4

2.4.2.1 TSS

137 The evaluator shall verify that the TSS describes a stateful packet filtering policy and the following attributes are identified as being configurable within stateful traffic filtering rules for the associated protocols:

- ICMPv4
  - Type
  - Code
- ICMPv6
  - Type
  - Code
- IPv4
  - Source address
  - Destination Address
  - Transport Layer Protocol
- IPv6
  - Source address
  - Destination Address

- Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP
  - Source Port
  - Destination Port
- UDP
  - Source Port
  - Destination Port

138 The evaluator shall verify that each rule can identify the following actions: permit or drop with the option to log the operation. The evaluator shall verify that the TSS identifies all interface types subject to the stateful packet filtering policy and explains how rules are associated with distinct network interfaces.

Findings:	This information is provided in the ST in section 6.13.

2.4.2.2 Guidance Documentation

139 The evaluators shall verify that the guidance documentation identifies the following attributes as being configurable within stateful traffic filtering rules for the associated protocols:

- ICMPv4
  - Type
  - Code
- ICMPv6
  - Type
  - Code
- IPv4
  - Source address
  - Destination Address
  - Transport Layer Protocol
- IPv6
  - Source address
  - Destination Address
  - Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields
- TCP

- Source Port
- Destination Port

- UDP
  - Source Port
  - Destination Port

140 The evaluator shall verify that the guidance documentation indicates that each rule can identify the following actions: permit, drop, and log.

Findings:	The [ADMIN] guide in Chapter 9, under “Object Configuration” > “Services” describes the process by which each of the protocol properties can be configured for use in the firewall policy table. Once the object is configured, specifying the action is described under Chapter 9, under “Firewall Policies” in [ADMIN]. Policies can be set to “ACCEPT” or “DENY”. Independently, policies can be set to log the traffic and optionally capture specific packets associated with the rule.

141 The evaluator shall verify that the guidance documentation explains how rules are associated with distinct network interfaces.

Findings:	In Chapter 9, under “Firewall Policies” in [ADMIN], firewall rules are associated with specific incoming and outcoming network interfaces.

2.4.2.3 Tests

142 Test 1: The evaluator shall use the instructions in the guidance documentation to test that stateful packet filter firewall rules can be created that permit, drop, and log packets for each of the following attributes:

- ICMPv4
  - Type
  - Code
- ICMPv6
  - Type
  - Code
- IPv4
  - Source address
  - Destination Address
  - Transport Layer Protocol
- IPv6
  - Source address
  - Destination Address

- Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP
  - Source Port
  - Destination Port
- UDP
  - Source Port
  - Destination Port

Note	The construction and effectiveness of the rules shall be tested as part of FFW_RUL_EXT.1.8 in accordance with the note given in the FW SD.

143 Test 2: Repeat the test assurance activity above to ensure that stateful traffic filtering rules can be defined for each distinct network interface type supported by the TOE.

Note	The construction and effectiveness of the rules shall be tested as part of FFW_RUL_EXT.1.8 in accordance with the note given in the FW SD.

144 Note that these test activities should be performed in conjunction with those of FFW_RUL_EXT.1.8 where the effectiveness of the rules is tested. The test activities for FFW_RUL_EXT.1.8 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfil the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

2.4.3 FFW_RUL_EXT.1.5

2.4.3.1 TSS

145 The evaluator shall verify that the TSS identifies the protocols that support stateful session handling. The TSS shall identify TCP, UDP, and ICMP if selected by the ST author.

Findings:	The ST, in section 6.13, identifies ICMPv4, ICMPv6, IPv4, IPv6, TCP and UDP as protocols that support stateful session handling.

146 The evaluator shall verify that the TSS describes how stateful sessions are established (including handshake processing) and maintained.

Findings:	Section 6.13 of the ST describes in detail how stateful session are established. Effectively an existing session database is consulted or a new session is created if it is permitted. For TCP connections, the handshake process for session handling is described in section 6.13.

147 The evaluator shall verify that for TCP, the TSS identifies and describes the use of the following attributes in session determination: source and destination addresses, source and destination ports, sequence number, and individual flags.

Findings:	In section 6.13, it is claimed the TOE will track sessions based on "*[a] number of variables (such as source/destination address and ports, sequence numbers, flags…*".

148 The evaluator shall verify that for UDP, the TSS identifies and describes the following attributes in session determination: source and destination addresses, source and destination ports.

Findings:	In section 6.13, it is claimed the TOE will track sessions based on "*[a] number of variables (such as source/destination address and ports…*".

149 The evaluator shall verify that for ICMP (if selected), the TSS identifies and describes the following attributes in session determination: source and destination addresses, other attributes chosen in FFW_RUL_EXT.1.5.

Findings:	In section 6.13, it is claimed the TOE will track sessions based on "*[a] number of variables (such as source/destination address…*".

150 The evaluator shall verify that the TSS describes how established stateful sessions are removed. The TSS shall describe how connections are removed for each protocol based on normal completion and/or timeout conditions. The TSS shall also indicate when session removal becomes effective (e.g., before the next packet that might match the session is processed).

Findings:	In section 6.13, the ST describes old sessions as being removed when their time-to-live has been exceeded or when the sessions have been closed on their own.

2.4.3.2 Guidance Documentation

151 The evaluator shall verify that the guidance documentation describes stateful session behaviours. For example, a TOE might not log packets that are permitted as part of an existing session.

Findings:	In [ADMIN], Chapter 22 describes the flow of packet data as it enters a physical interface on the TOE. The description provides an extensive view of how the stateful nature of established sessions are handled. It includes a discussion of hardware accelerated capabilities vs. non-accelerated behaviours. Diagrams and flow charts are provided to give the reader an understanding of the process.

2.4.3.3 Tests

152 Test 1: The evaluator shall configure the TOE to permit and log TCP traffic. The evaluator shall initiate a TCP session. While the TCP session is being established, the evaluator shall introduce session establishment packets with incorrect flags to determine that the altered traffic is not accepted as part of the session (i.e., a log event is generated to show the ruleset was applied). After a TCP session is successfully established, the evaluator shall alter each of the session determining attributes (source and destination addresses, source and destination ports, sequence number, flags) one at a time in order to verify that the altered packets are not accepted as part of the established session.

High-Level Test Description
Ensure that all traffic rules have been cleared out except for a single rule that permits and logs all IPv4 and IPv6 TCP traffic to be sent to the outside workstation. Start a wireshark capture on the inside and outside workstations. Using a script, initiate a TCP 3-way handshake to TCP port 22 on the inside workstation by sending a SYN packet, but before it is finalized, send packets with different flags and show they are logged and do not make it to the outside workstation. Then finally properly complete the 3-way handshake which establishes a session for the connection-oriented TCP protocol. Now construct a valid TCP packet, in turn, with the following invalid characteristics and send it to the outside workstation and show they are logged and dropped: • A different destination IP address; • A different source IP address; • A different destination port; • A different source port; • A different sequence number; and • A different set of flags.
Findings: PASS

153 Test 2: The evaluator shall terminate the TCP session established per Test 1 as described in the TSS. The evaluator shall then immediately send a packet matching the former session definition in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

High-Level Test Description
Perform a valid TCP 3-day handshake to establish a session and then tear down the connection. Immediately send a TCP packet to the outside target using the same session characteristics as the session that was just torn down. Verify on the outside workstation's wireshark capture that this invalid packet does not make it to the outside workstation.
Findings: PASS

154 Test 3: The evaluator shall expire (i.e., reach timeout) the TCP session established per Test 1 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

High-Level Test Description
Perform a valid TCP 3-day handshake to establish a session and then wait for the connection to terminate as defined in the ST. Wait for the prescribed amount of time until the TOE terminates the session and then immediately send a data packet using the same session characteristics as the session that was just torn down. Verify on the outside workstation's wireshark capture that this invalid packet does not make it to the outside workstation.
Findings: PASS

155 Test 4: The evaluator shall configure the TOE to permit and log UDP traffic. The evaluator shall establish a UDP session. Once a UDP session is established, the evaluator shall alter each of the session determining attributes (source and destination addresses, source and destination ports) one at a time in order to verify that the altered packets are not accepted as part of the established session.

High-Level Test Description
Ensure that all traffic rules have been cleared out except for a single rule that permits and logs all IPv4 and IPv6 traffic to be sent to the outside network. Start a wireshark capture on the inside and outside workstations. Using a script, transmit a UDP packet from the inside to the outside workstation. This will establish a new session with the TOE with an established reply path tracked by the TOE. Now construct and send (several) valid UDP packet on the outside workstation with the following invalid characteristics to the inside workstation and show they are logged and either dropped or comprise a new session on the TOE: • A different destination IP address; • A different source IP address; • A different destination port; • A different source port; As a sanity check, a return UDP packet using appropriate session-specific characteristics will be sent back to the inside server to ensure that the session-tracking mechanism works correctly. Verify on the outside workstation's wireshark capture that none of the invalid traffic from the inside workstation has been received by the outside workstation.
Findings: PASS

156 Test 5: The evaluator shall expire (i.e., reach timeout) the UDP session established per Test 4 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

High-Level Test Description
Building on the previous test case, establish a UDP session by transmitting a packet from the inside network to the outside network. As a sanity check, ensure that a valid return UDP packet can be received by the inside network. Then wait for the UDP session to terminate as described in the Security Target. After it is terminated, construct a UDP packet that would have normally been received on the return path, but show it is dropped and logged. Verify on the outside workstation's wireshark capture that this invalid packet does not make it to the outside workstation.
Findings: PASS

157 Test 6: If ICMP is selected, the evaluator shall configure the TOE to permit and log ICMP traffic. The evaluator shall establish a session for ICMP as defined in the TSS. Once an ICMP session is established, the evaluator shall alter each of the session determining attributes (source and destination addresses, other attributes chosen in FFW_RUL_EXT.1.5) one at a time in order to verify that the altered packets are not accepted as part of the established session.

High-Level Test Description
Ensure that all traffic rules have been cleared out except for a single rule that permits and logs all traffic to be sent to the outside network. Start a wireshark capture on the inside and outside workstations. Using a script, establish an ICMP session from the inside workstation to the outside workstation. Now construct and send a valid ICMP packet, in turn, with the following invalid characteristics to the inside workstation and show they are logged and dropped: • A different destination IP address; • A different source IP address; • A different type than expected as a reply; and • A different code than expected as a reply. As a sanity check, send back an ICMP packet that should match the session and ensure it is received. Verify on the outside workstation's wireshark capture that none of the invalid traffic from the inside workstation has transited the TOE to the outside workstation.
Findings: PASS

158 Test 7: If applicable, the evaluator shall terminate the ICMP session established per Test 6 as described in the TSS. The evaluator shall then immediately send a packet matching the former session definition in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

High-Level Test Description
Building upon the previous test case, establish an ICMP session and then tear down the connection properly using the teardown methodology described in the Security Target. Immediately send an ICMP packet to the inside target using the same session characteristics as the session that was just torn down. Verify on the inside workstation's wireshark capture that this invalid packet does not make it to the inside workstation.
Findings: PASS

159 Test 8: The evaluator shall expire (i.e., reach timeout) the ICMP session established per Test 6 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

High-Level Test Description
Building upon the previous test case, establish an ICMP session and then wait for the ICMP session to be torn down by timeout. Immediately after timeout send an ICMP packet to the inside target using the same session characteristics as the session that was just expired. Verify on the inside workstation's wireshark capture that this invalid packet does not make it to the inside workstation.
Findings: PASS

2.4.4 FFW_RUL_EXT.1.6

2.4.4.1 TSS

160 The evaluator shall verify that the TSS identifies the following as packets that will be automatically dropped and are counted or logged:

a) Packets which are invalid fragments, including a description of what constitutes an invalid fragment

b) Fragments that cannot be completely re-assembled

c) Packets where the source address is defined as being on a broadcast network

d) Packets where the source address is defined as being on a multicast network

e) Packets where the source address is defined as being a loopback address

f) The TSF shall reject and be capable of logging network packets where the source or destination address of the network packet is defined as being unspecified (i.e. 0.0.0.0) or an address “reserved for future use” (i.e. 240.0.0.0/4) as specified in RFC 5735 for IPv4;

g) The TSF shall reject and be capable of logging network packets where the source or destination address of the network packet is defined as an “unspecified address” or an address “reserved for future definition and use” (i.e. unicast addresses not in this address range: 2000::/3) as specified in RFC 3513 for IPv6;

h) Packets with the IP options: Loose Source Routing, Strict Source Routing, or Record Route specified

i) Other packets defined in FFW_RUL_EXT.1.6

Findings:	This information is provided in an extensive bullet-point list in section 6.13 of the ST. The definition of an invalid fragment is given as fragments which have sizes abnormal for the packet specification or offsets abnormal for the packet specifications.

2.4.4.2 Guidance Documentation

161 The evaluator shall verify that the guidance documentation describes packets that are discarded and potentially logged by default. If applicable protocols are identified, their descriptions need to be consistent with the TSS. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets.

Findings:	In the [SUPP] document under “Miscellaneous Logging”, the document describes the types of events and packets for which logging is enabled by default without configuration. Specifically, this is “dropped ICMP packets, dropped invalid IP packets”. Additional logging is configurable as described in the [SUPP] under “Firewall Specific Changes” with additional clarification of their effects as written in the [CLI] under the heading “system global”, “system settings” and “log settings”.

2.4.4.3 Tests

162 Test 1: The evaluator shall test each of the conditions for automatic packet rejection in turn. In each case, the TOE should be configured to allow all network traffic and the evaluator shall generate a packet or packet fragment that is to be rejected. The evaluator shall use packet captures to ensure that the unallowable packet or packet fragment is not passed through the TOE.

High-Level Test Description
Construct a rule permitting all traffic between the WAN and LAN networks. Then send packets with the following characteristics: a) IP fragments that are not valid; b) IP fragmented packets which cannot be re-assembled completely; c) IP packets where the source address is defined as being on a broadcast network (xxx.xxx.xxx.255 and 255.255.255.255; there is no equivalent of broadcast in IPv6); d) IP packets where the source address is defined as being on a multicast network (IPv4 of 224.0.0.0/24 or IPv6 of ff08::/8); e) IP packets where the source address is defined as being a loopback address (IPv4 of 127.0.0.0/8 or IPv6 of ::1/32); f) IP packets where the source address is defined as being unspecified (IPv4 of 0.0.0.0 or IPv6 of ::) g) IP packets where the destination address is defined as being unspecified (IPv4 of 0.0.0.0 or IPv6 of ::) h) IP packets where the source address is defined as an address “reserved for future use” (i.e. 240.0.0.0/4) as specified in RFC 5735 for IPv4; i) IP packets where the destination address is defined as an address “reserved for future use” (i.e. 240.0.0.0/4) as specified in RFC 5735 for IPv4; j) IP packets where the source address is defined as an “unspecified address” or an address “reserved for future definition and use” (i.e. unicast addresses not in this address range: 2000::/3) as specified in RFC 3513 for IPv6; k) IP packets where the destination address is defined as an “unspecified address” or an address “reserved for future definition and use” (i.e. unicast addresses not in this address range: 2000::/3) as specified in RFC 3513 for IPv6; l) IP packets with the Loose Source Routing option enabled; m) IP packets with the Strict Source Routing option enabled; and n) IP packets with the Record Route specified. Show these packets are dropped and logged and do not transit the TOE.
Findings: PASS

163 Test 2: For each of the cases above, the evaluator shall use any applicable guidance to enable dropped packet logging or counting. In each case above, the evaluator shall ensure that the rejected packet or packet fragment was recorded (either logged or an appropriate counter incremented).

Note	The logging and review of logs are done in the previous test case.

2.4.5 FFW_RUL_EXT.1.7

2.4.5.1 TSS

164 The evaluator shall verify that the TSS explains how the following traffic can be dropped and counted or logged:

a) Packets where the source address is equal to the address of the network interface where the network packet was received

b) Packets where the source or destination address of the network packet is a link-local address

c) Packets where the source address does not belong to the networks associated with the network interface where the network packet was received, including a description of how the TOE determines whether a source address belongs to a network associated with a given network interface

Findings:	The ST describes how the TOE handles packets that meet these criteria in section 6.13. The TOE only logs packets. For packets which do not match the source address of the given network interface, the TOE uses functionality called "Reverse Path Forwarding" (RPF) which prevents an IP packet from being forwarded if its source IP address either does not belong to a locally attached subnet (local interface), or be a hop on the routing between the TOE and another source.

2.4.5.2 Guidance Documentation

165 The evaluator shall verify that the guidance documentation describes how the TOE can be configured to implement the required rules. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets.

Findings:	In the [SUPP] document under "Additional default Firewall policies", the document describes the types of events and packets for which logged rules are created by default without configuration. Specifically, this is "Block local link traffic", "Block Class E traffic" and "Restrict the IPv6 address space to the allocated global unicast space.". In addition, when "strict-src-check" is enabled (see [CLI] under 'system settings') as part of the evaluated configuration, this option will prevent packets in which the source address is the TOE interface and where the source address does not belong to a network associated with a given network interface.

2.4.5.3 Tests

166 Test 1: The evaluator shall configure the TOE to drop and log network traffic where the source address of the packet matches that of the TOE network interface upon which the traffic was received. The evaluator shall generate suitable network traffic to match the configured rule and verify that the traffic is dropped and a log message generated.

High-Level Test Description
Using the available user-interfaces, construct rules that will log and drop traffic in which the source address of the packet matches the TOE interface upon which the traffic was received. Then, transmit network packets against the interface and show that the traffic is logged and dropped.
Findings: PASS

167 Test 2: The evaluator shall configure the TOE to drop and log network traffic where the source IP address of the packet fails to match the network reachability information of the interface to which it is targeted, e.g. if the TOE believes that network network 192.168.1.0/24 is reachable through interface 2, network traffic with a source address

from the 192.168.1.0/24 network should be generated and sent to an interface other than interface 2. The evaluator shall verify that the network traffic is dropped and a log message generated.

High-Level Test Description
As part of the CC evaluated configuration the strict source check flag is enabled. With this flag enabled the TOE will drop packets where detected network reachability is violated. Packets will be constructed and sent where the source address is of a network that is not reachable on the given interface. These will be logged and dropped. Packets will be constructed that have a source or destination address of the link-local address (IPv4 of 169.254.0.0/16 or IPv6 of FE80::/10) and will be shown to be logged and dropped.
Findings: PASS

2.4.6 FFW_RUL_EXT.1.8

2.4.6.1 TSS

168 The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset.

Findings:	This information is provided in the opening paragraphs of section 6.13 of the ST TSS. Default rules are processed before administrator-defined rules. Administrator-defined rules are ordered in a defined sequence order and applied as such.

2.4.6.2 Guidance Documentation

169 The evaluator shall verify that the guidance documentation describes how the order of stateful traffic filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing.

Findings:	In [ADMIN] Chapter 9 “Firewall Concepts” > “Firewall Policies” > “Policy Order”, the order of policies is described. The [CLI] describes the commands necessary to adjust the precedence with the move command in the section “firewall {policy | policy6}.

2.4.6.3 Tests

170 Test 1: The evaluator shall devise two equal stateful traffic filtering rules with alternate operations – permit and drop. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.

High-Level Test Description
After clearing all rules on the TOE, construct a rule that permits specific traffic and one that denies the exact same traffic. Start packet tracing on both the outside and inside networks. Send traffic that meets the rule and show that it is logged and accepted. Then reorder the rules such that the deny rule is ordered first. Send traffic that meets the rule and show that it is logged and dropped. Do this for both IPv4 and IPv6 traffic.
Findings: PASS

171 Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

High-Level Test Description
After clearing all rules on the TOE, construct a rule (i) that permits all traffic to a specific address and (ii) one that permits traffic to an entire network segment that encompasses the specific address. Start packet tracing on both the outside and inside networks. Send traffic from the outside network that meets rule (i) and show that it is logged and dropped. Then reorder the rules such that the rule (ii) is ordered first. Send traffic that meets the rule and show that it is logged and accepted. Do this for both IPv4 and IPv6 traffic.
Findings: PASS

2.4.7 FFW_RUL_EXT.1.9

2.4.7.1 TSS

172 The evaluator shall verify that the TSS describes the process for applying stateful traffic filtering rules and also that the behavior (either by default, or as configured by the administrator) is to deny packets when there is no rule match unless another required conditions allows the network traffic (i.e., FFW_RUL_EXT.1.5 or FFW_RUL_EXT.2.1).

Findings:	The ST describes the process for applying stateful traffic filtering rules in section 6.13 of the TSS. Furthermore, it claims in section 6.13 of the TSS that "*[i]f no matching rules is found, the TOE will automatically deny the packets and generate a log entry accordingly.*"

2.4.7.2 Guidance Documentation

173 The evaluator shall verify that the guidance documentation describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the guidance documentation provides the appropriate instructions to configure the behavior to deny packets with no matching rules.

Findings:	In [ADMIN] Chapter 9, "Firewall" > "Firewall Concepts" > "How Packets are handled by FortiOS" > "What is not expressly allowed is denied", the document describes that packets are denied by default. This behaviour is not configurable.

2.4.7.3 Tests

174 For each attribute in FFW_RUL_EXT.1.2, the evaluator shall construct a test to demonstrate that the TOE can correctly compare the attribute from the packet header to the ruleset, and shall demonstrate both the permit and deny for each case. The evaluator shall check the log in each case to confirm that the relevant rule was applied. The evaluator shall record a packet capture for each test to demonstrate the correct TOE behaviour.

High-Level Test Description
After clearing all rules on the TOE, construct a rule pair for each of the listed conditions

High-Level Test Description

• ICMPv4
  ○ type
  ○ code
• ICMPv6
  ○ type
  ○ code
• IPv4
  ○ Source address
  ○ Destination address
  ○ Transport layer protocol
• IPv6
  ○ Source address
  ○ Destination address
  ○ Transport layer protocol
  ○ Extensions
• TCP
  ○ Source port
  ○ Destination port
• UDP
  ○ Source port
  ○ Destination port
• Interface
 	
The rules will be constructed in an active state such that only one permit/deny pair will be active at any given time. All rules will be logged. Packets will be constructed to match the rule criteria and permit rules will be shown to be accepted (and logged) and deny rules will be shown to be denied and logged.

Findings: PASS

2.4.8 FFW_RUL_EXT.1.10

2.4.8.1 TSS

175 The evaluator shall verify that the TSS describes how the TOE tracks and maintains information relating to the number of half-open TCP connections. The TSS should identify how the TOE behaves when the administratively defined limit is reached and should describe under what circumstances stale half-open connections are removed (e.g. after a timer expires).

Findings:	Information about how half-open TCP sessions are maintained is described in section 6.13 of the ST TSS. This description includes how those sessions are limited and that they are closed when their time-to-live expires (or if cleared manually by an administrator).

2.4.8.2 Guidance Documentation

176 The evaluator shall verify that the guidance documentation describes the behaviour of imposing TCP half-open connection limits and its default state if unconfigured. The evaluator shall verify that the guidance clearly indicates the conditions under which new connections will be dropped e.g. per-destination or per-client.

Findings:	As per the [ST], the TOE does not differentiate (out of the box) between maximum half-open TCP sessions and maximum total TCP sessions. The various hardware

	guides describe the concurrent TCP sessions which apply for each model. The [ADMIN] (in Chapter 9 section “DoS Protection”) and [CLI] (in section “firewall {DoS-policy | DoS-policy6}”) both describe how a denial of service (DoS) policy can be established to limit the number of concurrent open TCP sessions by limiting the “tcp_dst_session” parameter in a DoS policy to the appropriate amount.

2.4.8.3 Tests

177 Test 1: The evaluator shall define a TCP half-open connection limit on the TOE. The evaluator shall generate TCP SYN requests to pass through the TOE to the target system using a randomised source IP address and common destination IP address. The number of SYN requests should exceed the TCP half-open threshold defined on the TOE. TCP SYN-ACK messages should not be acknowledged. The evaluator shall verify through packet capture that once the defined TCP half-open threshold has been reached, subsequent TCP SYN packets are not transmitted to the target system. The evaluator shall verify that when the configured threshold is reached that, depending upon the selection, either a log entry is generated or a counter is incremented.

High-Level Test Description
Clear all rules from the TOE. As noted in Section 6.13 of the ST the TOE does not differentiate between half-open and full TCP sessions. Therefore, configure a DOS Policy to block packets after the tcp_dst_session threshold (V) has been passed. Using a packet generator, send V*2 SYN packets to the inside workstation with randomized source IPv4 and IPv6 addresses that reside within the reachable network segment. The inside workstation should receive only V SYN packets. The rest should be logged and dropped.
Findings: PASS

2.5 Packet Filtering (FPF)

2.5.1 FPF_RUL_EXT.1.1 Rules for Packet Filtering

2.5.1.1 TSS

178 The evaluator shall verify that the TSS provide a description of the TOE’s initialization/startup process, which clearly indicates where processing of network packets begins to take place, and provides a discussion that supports the assertion that packets cannot flow during this process.

Findings:	In section 6.10.1, the ST describes the initialization process. Firewall rules are loaded after a series of cryptographic and TOE self-tests and before the network is transitioned to a link-up state. Without the link being in an up state, no packets can flow.

179 The evaluator shall verify that the TSS also includes a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describes the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets.

Findings:	Section 6.13 of the ST TSS provides an overview of the processing flow and how abnormal circumstances result in the TOE fails to a secure (ie. closed) state. This information was found to be consistent with Chapter 22 of the [ADMIN] “Parallel Path Processing - Life of a Packet”.

2.5.1.2 Guidance Documentation

180 The operational guidance associated with this requirement is assessed in the subsequent test assurance activities.

2.5.1.3 Tests

181 The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would otherwise be denied by the ruleset should be directed at the TOE’s interfaces, with packet sniffers listening to see if any network traffic is allowed through.

182 Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test assurance activities.

Findings:	This SFR is a subset of the functionality required for FFW_RUL_EXT.1.1.

2.5.2 FPF_RUL_EXT.1.2

2.5.2.1 TSS

183 The evaluator shall verify that the TSS indicates that the following protocols are supported:

- RFC 791 (IPv4)
- RFC 2460 (IPv6)
- RFC 793 (TCP)
- RFC 768 (UDP)

184 The evaluator shall verify that the TSS describes how conformance with the identified RFCs has been determined by the TOE developer (e.g., third party interoperability testing, protocol compliance testing).

Findings:	The ST in section 6.14 claims that conformance with RFC 791, 2460, 793 and 768 are achieved through compliance testing during development and release process with changes being made as required to ensure conformance with the requirements.

2.5.2.2 Guidance Documentation

185 The evaluator shall verify that the operational guidance indicates that the following protocols are supported:

- RFC 791 (IPv4)
- RFC 2460 (IPv6)
- RFC 793 (TCP)

- RFC 768 (UDP)

186 The guidance will describe the other protocols contained within the ST (e.g., IPsec, IKE, potentially HTTPS, SSH, and TLS) that are processed by the TOE. The evaluator ensures it is made clear what protocols were not considered as part of the TOE evaluation.

Findings:	The [ADMIN] guide in Chapter 9, under "Object Configuration" > "Services" describes the process by which each of the protocol properties can be configured for use in the firewall policy table. Once the object is configured, specifying the action is described under Chapter 9, under "Firewall Policies" in [ADMIN]. Policies can be set to "ACCEPT" or "DENY". Independently, policies can be set to log the traffic and optionally capture specific packets associated with the rule.

2.5.2.3 Tests

187 The testing associated with this requirement is addressed in the subsequent test assurance activities.

2.5.3 FPF_RUL_EXT.1.5

2.5.3.1 TSS

188 The evaluator shall verify that the TSS describes a Packet Filtering policy and the following attributes are:

- IPv4
  - Source address
  - Destination Address
  - Protocol
- IPv6
  - Source address
  - Destination Address
  - Next Header (Protocol)
- TCP
  - Source Port
  - Destination Port
- UDP
  - Source Port
  - Destination Port

189 The evaluator shall verify that each rule can identify the following actions: permit, deny, and log.

Findings:	This information is provided in the ST in section 6.13.

190 The evaluator shall verify that the TSS identifies all interface types subject to the Packet Filtering policy and explains how rules are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used) they can be treated collectively as a distinct network interface.RFCs has been determined by the TOE developer (e.g., third party interoperability testing, protocol compliance testing).

Findings:	This information is provided in the ST in section 6.13.

2.5.3.2 Guidance Documentation

191 The evaluators shall verify that the operational guidance identifies the following attributes as being configurable within Packet filtering rules for the associated protocols:

- IPv4
  - Source address
  - Destination Address
  - Protocol
- IPv6
  - Source address
  - Destination Address
  - Next Header (Protocol)
- TCP
  - Source Port
  - Destination Port
- UDP
  - Source Port
  - Destination Port

192 The evaluator shall verify that the operational guidance indicates that each rule can identify the following actions: permit, deny, and log.

Findings:	The [ADMIN] guide in Chapter 9, under “Object Configuration” > “Services” describes the process by which each of the protocol properties can be configured for use in the firewall policy table. Once the object is configured, specifying the action is described under Chapter 9, under “Firewall Policies” in [ADMIN]. Policies can be set to

	"ACCEPT" or "DENY". Independently, policies can be set to log the traffic and optionally capture specific packets associated with the rule.

193 The evaluator shall verify that the operational guidance explains how rules are associated with distinct network interfaces.

Findings:	In Chapter 9, under "Firewall Policies" in [ADMIN], firewall rules are associated with specific incoming and outcoming network interfaces.

2.5.3.3 Tests

194 The evaluator shall perform the following tests:

195 Test 1: The evaluator shall use the instructions in the operational guidance to test that packet filter rules can be created that permit, deny, and log packets for each of the following attributes:

- IPv4
  - Source address
  - Destination Address
  - Protocol
- IPv6
  - Source address
  - Destination Address
  - Next Header (Protocol)
- TCP
  - Source Port
  - Destination Port
- UDP
  - Source Port
  - Destination Port

Findings:	This SFR is a subset of the functionality required for FFW_RUL_EXT.1.4. Please refer to those findings for details.

196 Test 2: Repeat the test assurance activity above to ensure that Packet filtering rules can be defined for each distinct network interface type supported by the TOE.

Findings:	This SFR is a subset of the functionality required for FFW_RUL_EXT.1.4. Please refer to those findings for details.

197 Note that these test activities should be performed in conjunction with those of FPF_RUL_EXT.1.7 where the effectiveness of the rules is tested; here the evaluator

is just ensuring the guidance is sufficient and the TOE supports the administrator creating a ruleset based on the above attributes. The test activities for FPF_RUL_EXT.1.7 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfill the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

2.5.4 FPF_RUL_EXT.1.6

2.5.4.1 TSS

198 The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset.

Findings:	This information is provided in the opening paragraphs of section 6.13 of the ST TSS. Default rules are processed before administrator-defined rules. Administrator-defined rules are ordered in a defined sequence order and applied as such.

2.5.4.2 Guidance Documentation

199 The evaluator shall verify that the operational guidance describes how the order of Packet filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing.

Findings:	In [ADMIN] Chapter 9 “Firewall Concepts” > “Firewall Policies” > “Policy Order”, the order of policies is described. The [CLI] describes the commands necessary to adjust the precedence.

200 The evaluator shall verify that the operational guidance explains how rules are associated with distinct network interfaces.

Findings:	In Chapter 9, under “Firewall Policies” in [ADMIN], firewall rules are associated with specific incoming and outcoming network interfaces.

2.5.4.3 Tests

201 The evaluator shall perform the following tests:

202 Test 1: The evaluator shall devise two equal Packet filtering rules with alternate operations – permit and deny. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.

Findings:	This SFR is functionally equivalent to FFW_RUL_EXT.1.8. Please refer to those findings for details.

203 Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

Findings:	This SFR is functionally equivalent to FFW_RUL_EXT.1.8. Please refer to those findings for details.

2.5.5 FPF_RUL_EXT.1.7

2.5.5.1 TSS

204 The evaluator shall verify that the TSS describes the process for applying Packet filtering rules and also that the behavior (either by default, or as configured by the administrator) is to deny packets when there is no rule match unless another required conditions allows the network traffic (i.e., FPF_RUL_EXT.1.6 or FPF_RUL_EXT.1.7).

Findings:	The ST describes the process for applying stateful traffic filtering rules in section 6.13 of the TSS. Furthermore, it claims in section 6.13 of the TSS that "*[i]f no matching rules is found, the TOE will automatically deny the packets and generate a log entry accordingly.*"

2.5.5.2 Guidance Documentation

205 The evaluator shall verify that the operational guidance describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the operational guidance provides the appropriate instructions to configure the behavior to deny packets with no matching rules.

Findings:	In [ADMIN] Chapter 9, "Firewall" > "Firewall Concepts" > "How Packets are handled by FortiOS" > "What is not expressly allowed is denied", the document describes that packets are denied by default. This behaviour is not configurable.

2.5.5.3 Tests

206 The evaluator shall perform the following tests:

207 Test 1: The evaluator shall configure the TOE to permit and log each defined IPv4 Transport Layer Protocol (see table 5-2) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

High-Level Test Description
After clearing all rules on the TOE, construct a rule for each of the listed conditions. Generate traffic that will match the given rule and show that the packet is permitted. We use the TOE's ability to construct Address Objects, custom Services and custom Service Groups (collections of Service objects) to achieve the test case.
Findings: PASS

208 Test 2: The evaluator shall configure the TOE to permit all traffic except to deny and log each defined IPv4 Transport Layer Protocol (see table 5-2) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The

evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

High-Level Test Description
After clearing all rules on the TOE, construct a rule for each of the listed conditions. Generate traffic that will match the given rule and show that the packet is not permitted and is logged. We use the TOE's ability to construct Address Objects, custom Services and custom Service Groups (collections of Service objects) to achieve the test case.
Findings: PASS

Technical Decisions: The following assurance activities have been modified by TD0242.

209 Test 3: The evaluator shall configure the TOE to permit and log each defined IPv4 Transport Layer Protocol (see table 5-2) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to deny and log each defined IPv4 Transport Layer Protocol (See table 5-2) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

High-Level Test Description
After clearing all rules on the TOE, construct a rule for each of the listed conditions. Generate traffic that will match the given rule and show that the packet is not permitted and is logged. We use the TOE's ability to construct Address Objects, custom Services and custom Service Groups (collections of Service objects) to achieve the test case.
Findings: PASS

210 Test 4: The evaluator shall configure the TOE to permit and log each defined IPv6 Transport Layer Protocol (see table 5-2) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

High-Level Test Description
After clearing all rules on the TOE, construct a rule for each of the listed conditions. Generate traffic that will match the given rule and show that the packet is permitted.

High-Level Test Description
We use the TOE's ability to construct Address Objects, custom Services and custom Service Groups (collections of Service objects) to achieve the test case.
Findings: PASS

211 Test 5: The evaluator shall configure the TOE to permit all traffic except to deny and log each defined IPv6 Transport Layer Protocol (see table 5-2) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

High-Level Test Description
After clearing all rules on the TOE, construct a rule for each of the listed conditions. Generate traffic that will match the given rule and show that the packet is not permitted and is logged. We use the TOE's ability to construct Address Objects, custom Services and custom Service Groups (collections of Service objects) to achieve the test case.
Findings: PASS

212 Test 6: The evaluator shall configure the TOE to permit and log each defined IPv6 Transport Layer Protocol (see table 5-2) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to deny and log each defined IPv6 Transport Layer Protocol (see table 5-2) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that they are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged.

High-Level Test Description
After clearing all rules on the TOE, construct a rule for each of the listed conditions. Generate traffic that will match the given rule and show that the packet is not permitted and is logged. We use the TOE's ability to construct Address Objects, custom Services and custom Service Groups (collections of Service objects) to achieve the test case.
Findings: PASS

213 Test 7: The evaluator shall configure the TOE to permit and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

Note:	This test is conducted as a subset of FFW_RUL_EXT.1.9.

214 Test 8: The evaluator shall configure the TOE to deny and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Note:	This test is conducted as a subset of FFW_RUL_EXT.1.9.

215 Test 9: The evaluator shall configure the TOE to permit and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Here the evaluator ensures that the UDP port 500 (IKE) is included in the set of tests.

High-Level Test Description
This test is conducted as a subset of FFW_RUL_EXT.1.9. However, we will show that the firewall is capable of permitting UDP port 500 to the TOE interfaces. After clearing all rules on the TOE, construct a local-in rule in which UDP port 500 is permitted. Generate traffic that will match the given rule and show that the packet is allowed and is logged.
Findings: PASS

216 Test 10: The evaluator shall configure the TOE to deny and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Again, the evaluator ensures that UDP port 500 is included in the set of tests.

High-Level Test Description
After clearing all rules on the TOE, construct a local-in rule in which UDP port 500 is blocked. Generate traffic that will match the given rule and show that the packet is not permitted and is logged.
Findings: PASS

2.6 Identification and Authentication (FIA)

2.6.1 FIA_AFL.1 Authentication Failure Management

2.6.1.1 TSS

217 The evaluator shall examine the TSS to determine that it contains a description, for each supported method for remote administrative actions, of how successive

unsuccessful authentication attempts are detected and tracked. The TSS shall also describe the method by which the remote administrator is prevented from successfully logging on to the TOE, and the actions necessary to restore this ability.

Findings:	The ST describes the information in section 6.7 of the TSS. Specifically, each defined administrative interface except the local console will enforce authentication failures in a uniform way. Locked accounts are locked out until an administrator-defined time limit expires.

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

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

2.6.1.2 Guidance Documentation

219 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:	In the [CLI] under “system global” there are appropriate configurations documented for managing administrator lockout. This is also described in [ADMIN] under Chapter 28 in “Administrators” > “Administrator Lockout”. Only time-based lockouts are claimed and described.

220 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:	Since the account lockout does not affect the local console by default, no additional actions are needed to ensure administrator access will always be maintained.

2.6.1.3 Tests

221 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.

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

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

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

Note:	The TOE only claims time-based lockout.

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

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

2.6.2 FIA_PMG_EXT.1 Password Management

2.6.2.1 Guidance Documentation

222 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:	Appropriate guidance is given on the existing complexity requirements in [SUPP] in section "The FIPS-CC Mode of Operation". Appropriate guidance is provided on the need for secure passwords as given in [ADMIN] in Chapter 2 ("Getting Started") > "Basic Administration" > "Passwords". The [CLI] provides instructions under "system password-policy" to modify the minimum-length requirements of administrator passwords. The [ADMIN] guide provides the same type of guidance in Chapter 2 > "Basic Administration" > "Password Policy". The minimum length of 8 characters is enforced by the TOE and described in [SUPP].

2.6.2.2 Tests

223 The evaluator shall perform the following tests.

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

High-Level Test Description
Set the minimum password length to 15 characters. Attempt to set a password less than the minimum length and show it is not accepted. Attempt to set passwords that fail to include characters from the out-of-the-box password complexity requirements and show they are not accepted. Attempt to set a password that meets the complexity and length requirements and show it is accepted. Show the password can be used on applicable management interfaces to log in successfully. Show that an admin with privileges can change another user's password and that the audit log reflects this capability.
Findings: PASS

2.6.3 FIA_PSK_EXT.1 Pre-Shared Key Composition (VPN GW EP)

2.6.3.1 TSS

224 The evaluator shall examine the TSS to ensure that it identifies all protocols that allow both text-based and bit-based pre-shared keys, and states that text-based pre-shared keys of 22 characters are supported. For each protocol identified by the requirement,

the evaluator shall confirm that the TSS states the conditioning that takes place to transform the text-based preshared key from the key sequence entered by the user (e.g., ASCII representation) to the bit string used by the protocol, and that this conditioning is consistent with the last selection in the FIA_PSK_EXT.1.3 requirement.

Findings:	The TOE claims both text-based and bit-based PSKs as per section 6.5 of the ST TSS. The TOE claims text-based PSKs between 6 and 128 characters. Pre-shared keys are conditioned using SHA1 or the preconfigured IKE PRF as per RFC 409 for IKEv1 or RFC 4306 for IKEv2. These selections are consistent with the last selection in FIA_PSK_EXT.1.3.

2.6.3.2 Guidance Documentation

225 The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on the composition of strong text-based pre-shared keys, and (if the selection indicates keys of various lengths can be entered) that it provides information on the merits of shorter or longer preshared keys. The guidance must specify the allowable characters for preshared keys, and that list must be a super-set of the list contained in FIA_PSK_EXT.1.2.

Findings:	Appropriate guidance is provided on the need for secure passwords as given in [ADMIN] in Chapter 2 (“Getting Started”) > “Basic Administration” > “Passwords”. The [ADMIN] guide provides the same type of guidance in Chapter 2 > “Basic Administration” > “Password Policy”. The [ADMIN] guide specifies the allowable characters for pre-shared keys and the list was confirmed to be the same as the list given in FIA_PSK_EXT.1.2.

226 The evaluator shall confirm the operational guidance contains instructions for either entering bit-based pre-shared keys for each protocol identified in the requirement, or generating a bit-based pre-shared key (or both). The evaluator shall also examine the TSS to ensure it describes the process by which the bit-based pre-shared keys are generated (if the TOE supports this functionality), and confirm that this process uses the RBG specified in FCS_RBG_EXT.1 in the base PP.

Findings:	The TOE supports entering bit-based pre-shared keys only. The [CLI] guide under (‘ipsec vpn phase1-interface’ for the ‘psksecret’ value) describes that bit-based PSKs are entered by using a leading “0x” indicator to type out hexadecimal-based keys.

2.6.3.3 Tests

227 The evaluator shall also perform the following tests for each protocol (or instantiation of a protocol, if performed by a different implementation on the TOE). Note that one or more of these tests can be performed with a single test case.

228 Test 1: The evaluator shall compose a pre-shared key of 22 characters that contains a combination of the allowed characters in accordance with the operational guidance, and demonstrates that a successful protocol negotiation can be performed with the key.

High-Level Test Description
Modify the pre-shared key to meet or exceed 22 characters in length using an appropriate combination of the permitted characters and show that the connection can be established.
Findings: PASS

229 Test 2 [conditional]: If the TOE supports pre-shared keys of multiple lengths, the evaluator shall repeat Test 1 using the minimum length; the maximum length; and an invalid length. The minimum and maximum length tests should be successful, and the invalid length must be rejected by the TOE.

High-Level Test Description
For each key, configure the TOE with the key of the given length and show that for the minimum and maximum key length sizes, the tunnel is established. For the invalid length key, the key is rejected.
Findings: PASS

230 Test 3 [conditional]: If the TOE does not generate bit-based pre-shared keys, the evaluator shall obtain a bit-based pre-shared key of the appropriate length and enter it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

High-Level Test Description
Configure the TOE with a bit-based key by entering it in hexadecimal format. Ensure that the remote non-peer uses a different format to ensure that simple ASCII comparisons are not taking place.
Findings: PASS

231 Test 4 [conditional]: If the TOE does generate bit-based pre-shared keys, the evaluator shall generate a bit-based pre-shared key of the appropriate length and use it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

Note:	The TOE does not generate bit-based keys.

2.6.4 FIA_UIA_EXT.1 User Identification and Authentication

2.6.4.1 TSS

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

Findings:	The login process is described in section 6.7 of the ST TSS. It describes the process uniformly for all defined administrative interfaces (local/serial, remote/SSH, remote/web). This description includes username and passwords for all interfaces or SSH public keys when the SSH interface is used to complete the logon process and a key is provided.

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

Findings:	These actions are provided in the context of FMT_MTD.1/CoreData in section 6.9 of the ST which is an identical TSS requirement. The TOE claims no functions other than displaying a TOE banner or viewing the TOE version number via the GUI.

234 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.

235 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.

2.6.4.2 Guidance Documentation

236 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:	Other than constructing good passwords, the [CLI] guide (in 'system admin') instructs the user to compose SSH public/private key pairs using their SSH application for loading into the TOE. Once credential material is available, the TOE provides clear instructions for logging on using the serial console, SSH or the Web GUI. This information is found in [ADMIN] under Chapter 2 "Getting Started" in "Using the GUI" and "Using the CLI".

2.6.4.3 Tests

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

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

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

High-Level Test Description
Ensure the JSConsole activation indicates the user and origin of the attempt. The reason is that the JSConsole looks like a serial console, but is remotely accessible and therefore requires a more distinct origin of attempt.
Findings: PASS

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

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

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

High-Level Test Description
The device does not have any services configured prior to I&A aside from a TOE banner and being able to view the version of the TOE.
Findings: PASS

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

Test Not Applicable	The TOE is not a distributed TOE

2.6.5 FIA_UAU_EXT.2 Password-based Authentication Mechanism

238 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.

2.6.6 FIA_UAU.7 Protected Authentication Feedback

2.6.6.1 Tests

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

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

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

2.7 Security management (FMT)

2.7.1 General requirements for distributed TOEs

2.7.1.1 TSS

240 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.

2.7.1.2 Guidance Documentation

241 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.

2.7.1.3 Tests

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

Findings:	The TOE is not a distributed TOE.

2.7.2 FMT_MOF.1/ManualUpdate

2.7.2.1 TSS

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

Findings:	The TOE is not a distributed TOE.

2.7.2.2 Guidance Documentation

244 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:	A discussion of the steps needed to update the TOE can be found in the [SUPP] in the subsections of “Installing the CC Certified Firmware”. Additional information is found in [ADMIN] in Chapter 2 “Firmware”. The documentation indicates that the TOE will reboot after successfully installing the firmware image.

245 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.

2.7.2.3 Tests

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

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

247 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.

Note	This test case is covered in FPT_TUD_EXT.1.

2.7.3 FMT_MTD.1/CoreData Management of TSF Data

2.7.3.1 TSS

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

Findings:	The administrative functions available prior to I&A are described in FIA_UIA_EXT.1 above. According to the described functionality in section 6.9 of the ST, users are not permitted to manipulate TSF data and therefore it is acceptable not to argue how this information is prevented from being manipulated.

2.7.3.2 Guidance Documentation

249 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 combination of the [CLI], [ADMIN] and [SUPP] list all of the functions that can be used to manipulate TSF data.

2.7.4 FMT_SMF.1 Specification of Management Functions

250 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.

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

Technical Decisions:	The following assurance activities have been modified by TD0408.

251 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).

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

Findings:	The information was found in section 6.9 of the ST.

	The evaluator confirmed the TSS in section 6.7 describes the local administrative interface as a “console port”, which is unambiguously different from the stated remote connections. In addition, the guidance document in [ADMIN] clearly indicate the local console is a console in the industry-accepted sense of the word, requiring serial-port style settings to successfully connect.

253 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.

2.7.4.2 Guidance Documentation

254 See section 4.4.4.1.

2.7.4.3 Tests

255 The evaluator tests management functions as part of testing the SFRs identified in section 4.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

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

2.7.5 FMT_SMF.1 Specification of Management Functions (VPN GW EP)

Technical Decisions:	The following assurance activities have been modified by TD0319.

2.7.5.1 TSS

256 The evaluator shall verify that the TSS describes how the traffic filter rules for VPN traffic can be configured. Note that this activity can be addressed in parallel with the TSS assurance activities for FPF_RUL_EXT.1.

Findings:	The required information is provided in section 6.13 of the ST TSS. Note that the TOE claims both the FWcPP and the VPN GW EP. The traffic filter firewall rules can be configured precisely the same regardless of whether the traffic is destined for a VPN tunnel or in general.

2.7.5.2 Guidance Documentation

257 The evaluator shall verify that the operational guidance describes how to configure the traffic filter rules, including how to set any configurable defaults and how to configure each of the applicable rule attributes, actions, and associated interfaces. The evaluator must ensure that the operational guidance also provides instruction that would allow an administrator to ensure that configured rules are properly ordered.

Note that this activity should have been addressed with the Guidance assurance activities for FPF_RUL_EXT.1.

Findings:	Chapter 9 of the [ADMIN] describes stateful firewalls in general and how the TOE implements the required functionality. The chapter describes the TOE's firewall policies, the applicable configurable rule attributes, actions, how to enable logging, how to assign policies to interfaces and how to ensure they are ordered correctly.

2.7.5.3 Tests

258 The evaluator shall devise tests that demonstrate that the functions used to configure the TSF yield expected changes in the rules and that they are correctly enforced. A number of rule combination and ordering scenarios need to be configured and tested by attempting to pass both valid and invalid network traffic through the TOE. Note that this activity should have been addressed with a combination of the Test assurance activities for FPF_RUL_EXT.1.

Note:	The purpose of FMT_SMF.1 in the VPN GW EP is to augment the functionality from the same SFR found in the NDcPP/FWcPP. Unfortunately, because VPN GW EP is modified independently of the NDcPP/FWcPP, this SFR is out-of-sync. Based on the application notes in the EP and the notes from NIAP TD 319 (https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=183), we assume the following: The items in **Bold** font are *added* to the FMT_SMF.1.1 element. Note that the item "*Ability to enable, disable, determine and modify the behavior of all the security functions of the TOE identified in this EP to the Administrator;*" is a direct reference to a no-longer-available SFR called FMT_MOF.1(1)/AdminAct which is not defined in the VPN GW EP nor the NDcPP/FWcPP. It remains in the SFR, but has no testable qualities. The testing assurance activities (TSS, Guidance, Test cases) are all entirely related to FPF_RUL_EXT.1. Therefore, no test activities appear to be required for this SFR. Note that all of the Bold font additions to FMT_SMF.1 for the VPN GW EP technically are handled in other SFRs anyway.

2.7.6 FMT_SMF.1/IPS Specification of Management Functions (IPS)

2.7.6.1 TSS

259 The evaluator shall verify that the TSS describes how the IPS data analysis and reactions can be configured. Note that this activity should have been addressed with the TSS assurance activities for IPS_ABD_EXT.1, IPS_IPB_EXT.1 and IPS_ABD_EXT.1

Findings:	Refer to TSS assurance activities for IPS_ABD_EXT.1, IPS_IPB_EXT.1 and IPS_ABD_EXT.1.

2.7.6.2 Guidance Documentation

260 The evaluator shall verify that the operational guidance describes the instructions for each function defined in the SFR, describes how to configure the IPS data analysis and reactions, including how to set any configurable defaults and how to configure each of the applicable analysis pattern matching methods and reaction modes.

Findings:	The IPS functionality fully described in [ADMIN] in Chapter 25 “Security profiles” > “Intrusion Prevention”. This chapter describes how to configure IPS policies and reactions and how the IPS integrates into the overall UTM nature of the TOE.

2.7.6.3 Test

261 The evaluator shall perform the following tests:

262 Test 1: The evaluator shall use the operational guidance to create a signature and enable it on an interface. The evaluator shall then generate traffic that would be successfully triggered by the signature. The evaluator should observe the TOE applying the corresponding reaction in the signature.

High-Level Test Description
Ensure that all IDS/IPS firewall rules have been cleared out for both IPv4 and IPv6. Create a new IDS rule that will block traffic when it detects the EICAR virus example being transmitted over HTTP. Then send the EICAR signature from the outside workstation to the inside workstation. Do this for both IPv4 and IPv6 rulesets. Then delete the rules and generate the same traffic patterns and show that all packets are received.
Findings: PASS

263 Test 2: The evaluator shall then disable the signature and attempt to regenerate the same traffic and ensure that the TOE allows the traffic to pass with no reaction.

Note:	This was conducted as part of the previous test.

264 Test 3: The evaluator shall use the operational guidance to import signatures and repeat the test conducted in Test 1.

High-Level Test Description
Ensure that all IDS/IPS firewall rules have been cleared out for both IPv4 and IPv6. Import a previously constructed IDS/IPS ruleset using the identified TSFI. Generate traffic that is suitable to be caught by the imported IDS/IPS ruleset and show that the traffic is caught and dropped. Do this for both IPv4 and IPv6 rulesets. Then delete the rules and generate the same traffic patterns and show that all packets are received.
Findings: PASS

265 Note that all other functions should have been address with a combination of the test assurance activities for IPS_ABD_EXT.1, IPS_SBD_EXT.1.

2.7.7 FMT_SMR.2 Restrictions on security roles

2.7.7.1 Guidance Documentation

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

Findings:	The [ADMIN] guide describes in great detail in Chapter 2 “Getting Started” > “Using the GUI” and “Using the “CLI” as methods to administer the TOE using local and remote management interfaces. Those cited sections contain instructions for configuring, for example, the local serial client (baud rate, etc.).

2.7.7.2 Tests

267 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.

2.8 Protection of the TSF (FPT)

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

2.8.1.1 TSS

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

Findings:	The CSPs, PSKs, etc. are described in section 6.2 of the ST. The information is not protected at rest, but no interfaces are provided to access this material.

2.8.2 FPT_APW_EXT.1 Protection of Administrator Passwords

2.8.2.1 TSS

269 The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext

password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Findings:	Administrative passwords are described in section 6.10 of the ST. The information is protected at rest using AES encryption. No interfaces are provided to access this material directly.

2.8.3 FPT_FLS.1/SelfTest Fail Secure (Self-test Failures) (VPN GW EP)

2.8.3.1 TSS

270 The evaluator shall ensure the TSS describes how the TOE ensures a shutdown upon a self-test failure, a failed integrity check of the TSF executable image, or a failed health test of the noise source. If there are instances when a shut-down does not occur, e.g., a failure is deemed nonsecurity relevant, those cases are identified and a rationale supporting the classification and justification why the TOE's ability to enforce its security policies is not affected.

Findings:	Section 6.10 of the ST TSS provides this information. The TOE will halt awaiting a manual shutdown if any initialization self-tests fail which includes all of the required test categories described in the requirement.

2.8.4 FPT_TST_EXT.1 TSF testing

2.8.4.1 TSS

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

Findings:	Detailed self-tests are described in section 6.10 ST. These tests include CPU and BIOS self-tests, boot loader image verification, noise source tests and FIPS 140-2 KATs. These tests are argued as being sufficient. The evaluator agrees with the assessment.

272 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.

2.8.4.2 Guidance Documentation

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

Findings:	The [SUPP] describes the FIPS Error Mode which can occur and how to resolve the issue if encountered. FIPS Error Mode can occur on bootup in response to failed KATs which run at startup. In addition, in the [SUPP] also describes in "Potential Firmware issues" and "Potential hardware issues" that may occur as a result of the BIOS, hardware or firmware being corrupted. Information is provided on how to get support for these advanced topics. Finally, if the entropy seeding mechanism is unable to gather enough entropy, the [SUPP] describes ways in which this can be troubleshooted (in the 'Entropy" section).

274 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.

2.8.4.3 Tests

275 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.

276 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.

277 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.

278 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

2.8.5 FPT_TUD_EXT.1 Trusted Update

2.8.5.1 TSS

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

Findings:	The active version can be queried using interfaces provided in both the GUI and the CLI as per section 6.10 of the ST. Updates are applied immediately upon installation as described in the process provided in section 6.10.

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

Findings:	As per section 6.10 of the ST, the TOE relies on a 2048-bit digital signature to ensure integrity of the software/firmware update package. Verification occurs before installation and installation fails if the signature verification fails for any reason (missing or corrupt binary or signature).

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

Findings:	The TOE claims in section 6.10 of the ST to support automatic checking for updates, those updates are not applied automatically.

282 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.

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

Findings:	Certificate-based mechanisms are not used for this TOE.

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

Findings:	The TOE does not rely on hash-based integrity mechanisms.

2.8.5.2 Guidance Documentation

285 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] describes the recommended way of determining the current active version of the firmware by using the “get system status” command. Delayed activation is not claimed or supported.

286 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:	While the [SUPP] describes in section “Verifying the integrity of the firmware build” the process for validating that firmware has been downloaded from the public support server without any integrity issues. The [ADMIN] guide states in Chapter 2 “*Lastly, firmware images are signed and the signature is attached to the code as it is built. When upgrading an image, the running OS will generate a signature and compare it with the signature attached to the image. If the signatures do not match, the new OS will not load.*” This description corresponds to the description given in the TSS.

287 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.

288 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.

289 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.

290 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.

2.8.5.3 Tests

291 The evaluator shall perform the following tests:

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

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

b) Test 2 (if digital signatures are used): The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

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

2) An image that has not been signed

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

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

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

c) Test 3 (if published hash is verified on the TOE): If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

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

Test Not Applicable	The TOE does not support published hashes.

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

Test Not Applicable	The TOE does not support published hashes.

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

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

292 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.

293 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.

294 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.

2.8.6 FPT_STM_EXT.1 Reliable Time Stamps

2.8.6.1 TSS

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

Findings:	The evaluator found the required information in section 6.10 of the ST. The TOE has a built-in time source which is manually set by an administrator on-demand. The various functions that make use of the time are also disclosed.

2.8.6.2 Guidance Documentation

296 The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the guidance documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

Findings:	The ability to set the time is described in [ADMIN] in Chapter > “Basic Administration” > “System Settings” > “System Time”. It is also described in [CLI] under “execute date”. NTP is not claimed and must be disabled as part of the evaluated configuration. This configuration is described in [SUPP] under “Disable NTP”.

2.8.6.3 Tests

297 The evaluator shall perform the following tests:

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

High-Level Test Description
Get the current date and time. Change the date/time in the past by 1 day, 1 hour and 42 minutes. Verify the date/time was set properly. Change the date/time in the future by 7 days, 1 hour and 42 minutes. 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.

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

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

2.9 TOE Access (FTA)

2.9.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

2.9.1.1 Guidance Documentation

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

Findings:	The ability to set the idle timeout for session termination is described in [ADMIN] in Chapter > “Basic Administration” > “System Settings” > “Administration Settings”. It is also described in [CLI] under “system settings”.

2.9.1.2 Tests

300 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; 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

2.9.2 FTA_SSL.3 TSF-initiated Termination

2.9.2.1 Guidance Documentation

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

Findings:	The ability to set the idle timeout for both local and remote session termination is described in [ADMIN] in Chapter > "Basic Administration" > "System Settings" > "Administration Settings". It is also described in [CLI] under "system settings".

2.9.2.2 Tests

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

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

High-Level Test Description
For each of 1, 3, 5 minutes: Change the idle timeout to this value; Log into the device; Wait for the full duration of the timeout. The session should terminate. Note that because the system uses a single command to control all idle timers, we will set in one interface and check in another.
Findings: PASS

2.9.3 FTA_SSL.4 User-initiated Termination

2.9.3.1 Guidance Documentation

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

Findings:	This information is found in the [SUPP] in section "Administration" > "Logging out from the GUI and CLI".

2.9.3.2 Tests

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

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

High-Level Test Description
Log into the serial console. Log out using the TSFI previous discussed. Verify that the session has been terminated.
Findings: PASS

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

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

2.9.4 FTA_TAB.1 Default TOE Access Banners

2.9.4.1 TSS

Technical Decisions: The following assurance activities have been modified by TD0338.

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

Findings: The TSS indicates the methods of access in section 6.11 of the ST. The TOE banner is indicated on each of those mechanisms.

2.9.4.2 Guidance Documentation

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

Findings:	Using the GUI or the CLI, the administrator can modify the banner message by changing the replacement messages specific to login. This is described in [ADMIN] in Chapter 13 “Hardening” > “Security Best Practices” > “System administrator best practices”. The access disclaimer must be configured to occur before I&A. The [SUPP] describes how to do this in “Admin access disclaimer”.

2.9.4.3 Tests

307 The evaluator shall also perform the following test:

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

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

2.10 Trusted path/channels (FTP)

2.10.1 FTP_ITC.1 Inter-TSF trusted channel

2.10.1.1 TSS

Technical Decisions:	The following assurance activities have been modified by TD0290.

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

Findings:	Trusted channels are described in section 6.12 of the ST. The TOE provides a channel between syslog and VPN endpoints. Each channel is categorized according to how the communication can be initiated. The channels are described with their respective cryptographic protocols (TLS for syslog, IPSec for VPN). The protocols (TLS and VPN) are listed in sections 6.3.3 and 6.5.

2.10.1.2 Guidance Documentation

309 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:	For the logging server, the [SUPP] describes how to set up the TOE to communicate with the FortiAnalyzer in the "Logging to external devices" section. The FAZ, if unintentionally broken, can be rescued by following the instructions given in the section "Reconnecting to FortiAnalyzer". For IPSec VPN connections can be configured as per the [SUPP] in section "Phase 1/Phase2 encryption strength", but primarily in [ADMIN] Chapter 16. The TOE will continuously attempt to bring the VPN interface back up. If a network link is unintentionally broken and then restored, the VPN connection will automatically be retried. If the automatic reconnection fails, then the troubleshooting instructions given in Chapter 16 of [ADMIN] can help repair the VPN tunnel.

2.10.1.3 Tests

310 The evaluator shall perform the following tests:

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

Note	The only trusted channels are the remote audit log and the VPN IPSec, which are set up as per the evaluated configuration. They are 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
Logging device: 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. VPN IPSec: Engage wireshark over the appropriate interface. Start the VPN on the TOE side as the initiator.

High-Level Test Description
Examine wireshark and verify that the TOE sends an initial IKE message on UDP port 500.
Findings: PASS

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

High-Level Test Description
Logging device: Engage wireshark over the appropriate interface to capture log message traffic. Log into the serial device and logout. Examine wireshark and verify that the log interface sends encrypted traffic to the remote logging server IP endpoint. VPN IPSec: Engage wireshark over the appropriate interface to capture ESP traffic. Start the VPN and send traffic through it. Examine wireshark and verify that the IPSec interface sends encrypted traffic to the remote peer.
Findings: PASS

Technical Decisions:	The following assurance activities have been modified by TD0290.

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

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

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

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

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The

interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

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

High-Level Test Description
VPN IPSec: Establish a VPN tunnel and engage wireshark over the VPN interface. Ensure there are appropriate SPDs in place (firewall rules) to permit access to port 22 to the inside node. Transfer a large file over the VPN network using SSH. Physically disconnect the peer from the network which will disrupt the IPSec VPN tunnel while the transfer is occurring. Immediately plug the cable back in to emulate a MAC layer interruption. Review the communications to determine if the channel has been interrupted. Ensure that packets continue to be encapsulated by ESP and are unreadable. Repeat the above with a longer timeout until the TOE detects the interface is down.
Findings: PASS

311 Further assurance activities are associated with the specific protocols.

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

Test Not Applicable	The TOE is not a distributed TOE.

2.10.2 FTP_TRP.1/Admin Trusted Path

2.10.2.1 TSS

313 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:	Trusted paths are described in section 6.12 of the ST. The TOE provides both a CLI over SSH and a web GUI over HTTPS. The protocols listed in section 6.3.1 and 6.4 support these claims.

2.10.2.2 Guidance Documentation

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

Findings:	Use of SSH for the remote CLI or HTTP/TLS for the remote web GUI are described in Chapter 2 of the [ADMIN] guide under “Using the CL” and “Using the GUI”, respectively. The [SUPP] also describes the cryptographic parameters of the web GUI TLS channel in section “Web browser requirements”.

2.10.2.3 Tests

315 The evaluator shall perform the following tests:

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

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

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

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

Technical Decisions:	The following assurance activities have been modified by TD0290.

c) Test 3: *This test is removed.*

316 Further assurance activities are associated with the specific protocols.

317 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.

3 Evaluation Activities for Optional Requirements

3.1 Security management (FMT)

3.1.1 FMT_MTD.1/CryptoKeys Management of TSF Data

3.1.1.1 TSS

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

Findings:	The TOE is not a distributed TOE.

3.1.1.2 Tests

319 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.

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

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

3.1.2 FMT_MOF.1/Services Management of security functions behaviour

3.1.2.1 TSS

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

Findings:	The TOE is not a distributed TOE.

3.1.2.2 Tests

322 The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) without prior authentication as security administrator (either by authenticating as a user with no administrator privileges, if possible, or without prior authentication at all). The attempt to enable/disable this service/these services 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 enable/disable this service/these services 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.

323 The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) with prior authentication as security administrator. The attempt to enable/disable this service/these services should be successful.

High-Level Test Description
Create an unprivileged user. As the unprivileged user, attempt to stop each of the predefined applicable services (trusted paths, trusted channels). The attempt will be unsuccessful. As the privileged user, attempt to start and stop each of the predefined applicable services (trusted paths, trusted channels). The attempt will be successful.
Findings: PASS

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

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

Findings:	The ST in section 6.3.1 provides a description as to how the TOE conforms with RFC 2818.

4.1.1.2 Tests

325 The evaluator shall perform the following tests:

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

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

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

Note	Please refer to FCS_TLSS_EXT.1 for applicable test cases.

327 If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:

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

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

4.1.2 FCS_IPSEC_EXT.1 IPsec Protocol

4.1.2.1 TSS

FCS_IPSEC_EXT.1.1

328 The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

329 As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Findings:	The ST provides an overview of the packet processing ruleset in section 6.5 of the TSS. The TOE provides for BYPASS, DISCARD and PROTECT actions against administrator-defined rules. The rules are processed in the order they are defined. The order is administrator-controlled. Since the rules are administrator-defined and administrator-controlled for order, the description is sufficient to cover the requirements regarding how rules are applied and for which packet types.

FCS_IPSEC_EXT.1.3

330 The evaluator checks the TSS to ensure it states that the VPN can be established to operate in transport mode and/or tunnel mode (as identified in FCS_IPSEC_EXT.1.3).

Findings:	The TSS states that the TOE can operate in both transport mode and tunnel mode as in section 6.5 of the ST.

FCS_IPSEC_EXT.1.4

331 The evaluator shall examine the TSS to verify that the selected algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1/KeyedHash Cryptographic Operations (for keyed-hash message authentication).

Findings:	The TSS, in section 6.5, indicates that both AES-CBC and AES-GCM are implemented for key sizes of 128- and 256-bits. SHA is implemented for HMAC. These conform with the selections made in FCS_IPSEC_EXT.1.4 in section 5.3.2 of the ST. The data encryption and integrity algorithms are claimed in FCS_COP.1/DataEncryption and FCS_COP.1/KeyedHash as in section 5.3.2 of the ST and further described in the TSS section 6.2.

FCS_IPSEC_EXT.1.5

332 The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

333 For IKEv1 implementations, the evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

Findings:	In the ST, in section 6.5, IKEv1 and IKEv2 are permitted. IKEv1 does not permit aggressive mode. This is not configurable.

FCS_IPSEC_EXT.1.6

334 The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms chosen in the selection of the requirement are included in the TSS discussion.

Findings:	The TSS, in section 6.5, indicates that both AES-CBC and AES-GCM are implemented for key sizes of 128- and 256-bits. These conform with the selections made in FCS_IPSEC_EXT.1.6 in section 5.3.2 of the ST. The data encryption algorithms are claimed in FCS_COP.1/DataEncryption as in section 5.3.2 of the ST and further described in the TSS section 6.2.

FCS_IPSEC_EXT.1.7

335 The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 1 SA lifetime and/or the IKEv2 SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Findings:	The lifetime configuration methods are described in section 6.5 of the ST. IKEv1 Phase 1 SA and IKEv2 SA lifetimes are based on time only. This corresponds to the selections made in FCS_IPSEC_EXT.1.5 (regarding IKE versions supported).

FCS_IPSEC_EXT.1.8

336 The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 2 SA lifetime and/or the IKEv2 Child SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Findings:	The lifetime configuration methods are described in section 6.5 of the ST. IKEv1 Phase 2 SA and IKEv2 Child SA lifetimes are based on time or data volume. This corresponds to the selections made in FCS_IPSEC_EXT.1.5 (regarding IKE versions supported).

FCS_IPSEC_EXT.1.9

337 The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x". The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" meets the stipulations in the requirement.

Findings:	Section 6.5 of the TSS in the ST describes the process for generating the exponent ‘x’. The value of ‘x’ is supposed to be twice the security strength of the claimed cipher. Only DH groups 14, 19 and 20 are permitted and required. These represent values of ‘x’ of 224, 256 and 384, respectively. These values are claimed in the ST.

FCS_IPSEC_EXT.1.10

338 If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that

meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

Findings:	SFR FCS_IPSEC_EXT.1 claims the second selection as per section 5.3.2 of the ST.

339 If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

Findings:	SFR FCS_IPSEC_EXT.1 claims the second selection as per section 5.3.2 of the ST. Nonces are generated as described in section 6.5. They are generated using the validated DRBG: 128-bits for SHA1 and SHA2-256; and 256-bits for SHA2-384 and SHA2-512. These lengths meet the stipulated requirements since nonces must be at least 128-bits in length or at least half the length of the output size for the negotiated PRF. The largest output size is SHA2-512 which has an output size of 512-bits.

FCS_IPSEC_EXT.1.11

340 The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Findings:	The TOE supports DH groups 14, 19 and 20. As described in the ST in section 6.5, the selection of the DH group is based on the established policy configuration.

FCS_IPSEC_EXT.1.12

341 The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

Findings:	This information is provided in section 6.5 of the ST TSS. The TOE ensures that the cryptographic strengths of the algorithms selected f IKEv1 Phase 2 SA or IKEv2 Child SA are greater than or equal to those selected in the IKEv1 Phase 1 SA or IKEv2 SA (respectively).

FCS_IPSEC_EXT.1.13

342 The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1/SigGen Cryptographic Operations (for cryptographic signature).

Findings:	The TOE supports both RSA and ECDSA as per the TSS in section 6.5. These claims are consistent with the selections in FCS_COP.1/SigGen which claims both RSA and ECDSA.

343 If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

Findings:	As described in section 6.5 of the ST TSS, the TOE can use predefined text-based or bit-based PSKs. The TOE does not generate PSKs.

FCS_IPSEC_EXT.1.14

Technical Decisions:	The following assurance activities have been modified by TD0343.

344 The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN). If the TOE simultaneously supports the same identifier type in the CN and SAN, the TSS shall describe how the TOE prioritizes the comparisons (e.g. the result of comparison if CN matches but SAN does not). If the location (e.g. CN or SAN) of non-DN identifier types must explicitly be configured as part of the reference identifier, the TSS shall state this. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate, including what field(s) are compared and which fields take precedence in the comparison.

Findings:	In section 6.5 of the ST TSS, the TOE only claims that the Distinguished Name can be used for identification. The TOE does not claim CN or SAN identifiers. There is no evidence to suggest that the specific field containing the reference identifier can be configured by the operator. No other identifier types are included by the ST author.

4.1.2.2 Guidance Documentation

FCS_IPSEC_EXT.1.1

345 The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases – a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

Findings:	The TOE claims route-based VPNs (also known as interface-based VPNs). These VPNs operate as first-class interfaces in the TOE. The SPD is implemented as firewall policies against the VPN interface. Policy rule definition for allowing traffic to flow over the VPN, be blocked and bypass are defined in [ADMIN] Chapter 16 under "Defining VPN Security Policies"

FCS_IPSEC_EXT.1.3

346 The evaluator shall confirm that the guidance documentation contains instructions on how to configure the connection in each mode selected.

Findings:	Tunnel mode is the default operational IPSec mode in the TOE. If transport mode is desired, then [CLI] describes under "vpn ipsec phase2-interface" how to modify the connection type by using the "encapsulation transport" setting. To revert back to the "tunnel" mode, the [CLI] option "encapsulation tunnel" can be used.

FCS_IPSEC_EXT.1.4

347 The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

Findings:	Encryption and integrity algorithms can be selected in the web interface in the "phase 2" proposal settings as described in [ADMIN] Chapter 16 > "IPSec VPN in the web-based manager". The [CLI] can set the "vpn ipsec phase2-interface" proposal strings to the claimed and permitted algorithms.

FCS_IPSEC_EXT.1.5

348 The evaluator shall check the guidance documentation to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and how to configure the TOE to perform NAT traversal (if selected).

Findings:	IKE version can be selected using the web interface as described in Chapter 16 of the [ADMIN] guide under "IPSec VPN in the web-based manager". Alternatively, the CLI can be used to set the version of IKE in the "vpn ipsec phase1-interface" settings using the "ike-version" parameter. NAT traversal is configured in the GUI as per Chapter 16 of [ADMIN]. It can be configured to be enabled or disabled in the GUI by unselecting the check box. It can also be set to "Forced" to always operate even if no NAT is present. Within the [CLI] under "vpn ipsec phase1-interface", the "nattraversal" parameter can be used to specify the NAT mode of operation.

349 If the IKEv1 Phase 1 mode requires configuration of the TOE prior to its operation, the evaluator shall check the guidance documentation to ensure that instructions for this configuration are contained within that guidance.

Findings:	IKEv1 operates in main mode by default.

FCS_IPSEC_EXT.1.6

350 The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement.

Findings:	Encryption and integrity algorithms for IKEv1/v2 can be selected in the web interface in the "phase 1" proposal settings as described in [ADMIN] Chapter 16 > "IPSec VPN in the web-based manager". The [CLI] can set the "vpn ipsec phase1-interface" proposal strings to the claimed and permitted algorithms.

FCS_IPSEC_EXT.1.7

351 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 1 SA values for 24 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

Findings:	The TOE only claims time-based limits for IKEv1 phase 1 and IKEv2 SA. These limits are modified using the GUI as per [ADMIN] Chapter 16 > "IPSec VPN in the web-based manager". The [CLI] can also set the time-based limits using the "vpn ipsec phase1-interface" and setting the "keylife" parameter to the number of seconds until rekey (between 1 and 172800 seconds – 48 hours).

FCS_IPSEC_EXT.1.8

352 The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, the evaluator ensures that the Administrator is able to configure Phase 2 SA values for 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

Findings:	The TOE claims both volume-based and time-based limits for IKEv1 phase 2 and IKEv2 CHILD SA. These limits are modified using the GUI as per [ADMIN] Chapter 16 > "IPSec VPN in the web-based manager". The [CLI] can also set the volume and time-based limits using the "vpn ipsec phase2-interface" and setting the "keylife-type" parameter to either "seconds" or "bytes" or "both". Depending on the value of the "keylife-type", new CLI parameters called "keylifeseconds" or "keylifekbs" are made available denoting limits for number of seconds until rekey (between 1 and 172800 seconds – 48 hours) or number of KB until rekey, respectively.

FCS_IPSEC_EXT.1.11

353 The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement.

Findings:	Key exchange algorithms for IKEv1/v2 can be selected in the web interface in the "phase 1" proposal settings as described in [ADMIN] Chapter 16 > "IPSec VPN in the web-based manager". The [CLI] can set the "vpn ipsec phase1-interface" key exchange strings using the "dhgrp" parameter to the claimed and permitted algorithms.

FCS_IPSEC_EXT.1.13

354 The evaluator ensures the guidance documentation describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

Findings:	In [ADMIN] Chapter 3, "Authentication" > "Certificate-based Authentication" > "Configuring certificate-based authentication", there is a section devoted to configuring IPSec connections with X.509 certificates. This section describes how to load and configure X.509 certificates for the TOE and the peer and assign them to the VPN configuration by creating VPN users and user groups. Chapter 16 of [ADMIN] also goes into detail about configuring the IPSec VPN for digital signature authentication. The CLI has equivalent expressive capabilities by manually configuring the VPN users through the "user peer" CLI branch. Certificates are managed using the GUI only as per the [SUPP] section "VPN and Certificate Specific Settings".

355 The evaluator shall check that the guidance documentation describes how pre-shared keys are to be generated and established. The description in the guidance documentation shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

Findings:	Pre-shared keys can be configured in the web interface in the "phase 1" proposal settings as described in [ADMIN] Chapter 16 > "IPSec VPN in the web-based manager". The TOE only uses pre-generated PSKs: it does not generate PSKs.

356 The evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA, and ensure a valid certificate for that CA is loaded into the TOE and marked "trusted".

Findings:	The TOE does not connect to an external CA for any PKI operations except for automatically refreshing CRLs. CAs are loaded manually by the administrator. Chapter 3 in [ADMIN] describes the CA trust store.

FCS_IPSEC_EXT.1.14

Technical Decisions:	The following assurance activities have been modified by TD0343.

357 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 does not guarantee unique identifiers, 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:	Chapter 3 in [ADMIN] under "Certificate-based Authentication" > "Configuring certificate-based authentication" provides an overview of how to set reference identifiers for VPN users and peers. This chapter provides information on the 'subject'. The [SUPP] instructs administrators to use only the Web Interface to manage and assign certificates. This includes manging them in the context of peer authentication. The Web GUI presents an interface to insert the reference DN to match. [SUPP] also gives explicit direction that SANs are not used in the check of peer identities. This explicit statement is found in section "VPN and Certificate Specific Settings" > "Miscellaneous".

4.1.2.3 Tests

FCS_IPSEC_EXT.1.1

358 The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

a) Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behaviour: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

High-Level Test Description
Using a VPN which is configured with a pre-shared key, create three rules in the IPv4 firewall table which will permit traffic to bypass the VPN or enter (and pass) the VPN or enter (and be blocked) at the VPN. Show positive and negative tests that meet the rules.

High-Level Test Description
Findings: PASS

b) Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and guidance documentation. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation.

Note:	This test is conducted extensively as part of FWcPP v2.0e. Since the VPN "SPD" is implemented as firewall policy rules, the behaviour seen in the firewall policy rules can be used to show correct behaviour for the given requirements. Note that use of the firewall policy engine as the SPD implementation is verified in test 1 as well as throughout the remainder of this test plan. Also, see FPF_RUL_EXT.1 for additional firewall/SPD based testing.

FCS_IPSEC_EXT.1.2

359 The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

360 The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

361 The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet.

362 The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a "TOE created" final entry that discards packets that do not match any previous entries). The evaluator sends the packet, and observes that the packet was dropped.

Note:	This test is performed as part of FCS_IPSEC_EXT.1.1 in which a plaintext packet was successfully transmitted through the TOE.

FCS_IPSEC_EXT.1.3

363 The evaluator shall perform the following test(s) based on the selections chosen:

a) Test 1: If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic

algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

High-Level Test Description
Configure the outside workstation VPN and TOE to operate the VPN in tunnel mode. Initiate a connection and show that the VPN is established.
Findings: PASS

b) Test 2: If transport mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

High-Level Test Description
Configure the outside workstation VPN and TOE to operate the VPN in transport mode. Initiate a connection and show that the VPN is established.
Findings: PASS

FCS_IPSEC_EXT.1.4

364 The evaluator shall configure the TOE as indicated in the guidance documentation configuring the TOE to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the TOE to negotiate phase 2 with the given encryption and integrity algorithm. Send traffic through the VPN and verify that the packets are encapsulated.
Findings: PASS

FCS_IPSEC_EXT.1.5

365 Tests are performed in conjunction with the other IPsec evaluation activities.

a) Test 1: If IKEv1 is selected, the evaluator shall configure the TOE as indicated in the guidance documentation, and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.

High-Level Test Description
Show that the TOE will not permit establishing the VPN IKEv1 in aggressive mode.
Findings: PASS

b) Test 2: If NAT traversal is selected within the IKEv2 selection, the evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

High-Level Test Description
After configuring the peer to operate behind a NAT, initiate an IPSec connection from the peer to the TOE and show that the IPSec connection is established and that it traverses the NAT successfully.
Findings: PASS

FCS_IPSEC_EXT.1.6

366 The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the TOE to negotiate phase 1 with the given ciphersuite. Send traffic through the VPN and verify that the packets are encapsulated.
Findings: PASS

FCS_IPSEC_EXT.1.7

367 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC “A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.”

368 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If ‘number of bytes’ is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

Note:	The TOE does not claim volume-based rekeying for phase 1 SAs.

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime of 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that a new Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the TOE to rekey after 24 hours has elapsed to show it can be done. Then configure the TOE to rekey after *t* minutes have elapsed for phase 1 and show that it actually rekeys after the given amount of time.
Findings: PASS

FCS_IPSEC_EXT.1.8

369 When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC "A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered."

370 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If 'number of bytes' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the TOE to only accept 10 MB of traffic before a new phase 2 SA is negotiated. Configure the peer to only accept 1 GB of traffic before a new phase 2 SA is negotiated. Send traffic from the peer to the TOE and verify the phase 2 SA is negotiated after the configured limit is reached.
Findings: PASS

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime of 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of the TOE. The evaluator shall establish an SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once a new Phase 2 SA is negotiated when or before 8 hours has lapsed. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the TOE to rekey phase 2 after 8 hours has elapsed to show it can be done. Then configure the TOE to rekey after *t* minutes have elapsed for phase 2 and show that it actually rekeys after the given amount of time.
Findings: PASS

FCS_IPSEC_EXT.1.10

371 Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

Note:	This is a misplaced TSS requirement. No test performed. Furthermore, the [ST] does not claim the first selection.

b) Test 2: If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

Note:	This is a misplaced TSS requirement. No test performed. The [ST] claims the second selection. As written in section 6.5 of the [ST], the TOE utilises CTR-DRBG with AES (as specified in FCS_RBG_EXT.1) to generate the exponents used in IKE key exchanges, having the possible lengths of 224, 256 or 384 bits, corresponding to each of the supported DH groups. Nonces used in IKE are generated in this same way for negotiated PRF hashes. Nonce sizes are: a) 128 bits for SHA-1 and SHA-256; (b) 256 bits for SHA-384 and SHA-512.

FCS_IPSEC_EXT.1.11

372 For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the VPN tunnel on the TOE and the peer to negotiate only the supported key exchange ciphersuite under test for both phase 1 and phase 2 and show that the connection can be established.
Findings: PASS

FCS_IPSEC_EXT.1.12

373 The evaluator simply follows the guidance to configure the TOE to perform the following tests.

a) Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

Note:	This test was conducted in full for FCS_IPSEC_EXT.1.4.

b) Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the VPN tunnel on the TOE to be able to select amongst several phase 1 and phase 2 symmetric ciphers. Configure the peer to attempt to negotiate a phase1/phase2 symmetric pair with strength(ike-alg) < strength(esp-alg) and show that it fails.
Findings: PASS

c) Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the VPN tunnel on the TOE to be able to select amongst all supported ciphersuites for phase 1. Configure the peer to use an unsupported ciphersuite for phase 1 and show the connection is not established.
Findings: PASS

d) Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

High-Level Test Description
For each of IKEv1 and IKEv2: Configure the VPN tunnel on the TOE to be able to select amongst all supported ciphersuites for phase 1 and phase 2. Configure the peer to use an unsupported ciphersuite for phase 2 and show the phase 2 SA is not established.
Findings: PASS

FCS_IPSEC_EXT.1.13

374 For efficiency sake, the testing that is performed may be combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1. The following tests shall be repeated for each peer authentication method selected in FCS_IPSEC_EXT.1.13:

a) Test 1: The evaluator shall configure the TOE to use a private key and associated certificate signed by a trusted CA and shall establish an IPsec connection with the peer.

Note:	This test case is performed as part of FIA_X509_EXT.1/Rev Test 1(a).

b) Test 2: If pre-shared keys are selected, the evaluator shall generate a pre-shared key off-TOE and use it, as indicated in the guidance documentation, to establish an IPsec connection with the peer.

Note:	This test case is performed throughout FCS_IPSEC_EXT.1.

FCS_IPSEC_EXT.1.14

Technical Decisions:	The following assurance activities have been modified by TD0343.

375 In the context of the tests below, a valid certificate is a certificate that passes FIA_X509_EXT.1 validation checks but does not necessarily contain an authorized subject.

The evaluator shall perform the following tests:

a) Test 1: (conditional) For each CN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes CN checking over SAN (through explicit configuration of the field when specifying the reference identifier or prioritization rules), the evaluator shall

also configure the SAN so it contains an incorrect identifier of the correct type (e.g. the reference identifier on the TOE is example.com, the CN=example.com, and the SAN:FQDN=otherdomain.com) and verify that IKE authentication succeeds.

Note:	This test case is not applicable to the TOE because it does not claim CN identifiers.

b) Test 2: (conditional) For each SAN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes SAN checking over CN (through explicit specification of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the CN so it contains an incorrect identifier formatted to be the same type (e.g. the reference identifier on the TOE is DNS-ID; identify certificate has an identifier in SAN with correct DNS-ID, CN with incorrect DNS-ID (and not a different type of identifier)) and verify that IKE authentication succeeds.

Note:	This test case is not applicable to the TOE because it does not claim SAN identifiers.

c) Test 3: (conditional) For each CN/identifier type combination selected, the evaluator shall:

Create a valid certificate with the CN so it contains the valid identifier followed by '\0'. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the evaluator shall configure the SAN so it matches the reference identifier.

Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the CN without the '\0' and verify that IKE authentication fails.

Note:	This test case is not applicable to the TOE because it does not claim CN identifiers.

d) Test 4: (conditional) For each SAN/identifier type combination selected, the evaluator shall:

Create a valid certificate with an incorrect identifier in the SAN. The evaluator shall configure a string representation of the correct identifier in the DN. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the addition/modification shall be to any non-CN field of the DN. Otherwise, the addition/modification shall be to the CN.

Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the correct identifier (expected in the SAN) and verify that IKE authentication fails

Note:	This test case is not applicable to the TOE because it does not claim SAN identifiers.

e) Test 5: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Note:	Tests 5 and 6 are combined below.

f) Test 6: (conditional) If the TOE supports DN identifier types, to demonstrate a bit-wise comparison of the DN, the evaluator shall create the following valid certificates and verify that the IKE authentication fails when each certificate is presented to the TOE:

Duplicate the CN field, so the otherwise authorized DN contains two identical CNs.

Append '\0' to a non-CN field of an otherwise authorized DN.

High-Level Test Description
Connect the TOE to an IPSec VPN gateway using certificate-based authentication and verify that when using a fully-specified DN, the connection is successful. Then, configure a certificate to have a duplicate CN and show it is rejected. Then, configure a certificate to have "\0" appended to a non-CN RDN and show it is rejected.
Findings: PASS

4.1.3 FCS_SSHS_EXT.1 SSH Server

4.1.3.1 TSS

FCS_SSHS_EXT.1.2

Technical Decisions:	The following assurance activities have been modified by TD0339.

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

Findings:	Section 6.4 of the ST TSS indicates that RSA public key authentication is permitted along with password-based authentication. The choice of public key algorithm is consistent with the selection made in FCS_SSHS_EXT.1.5 in section 5.3.2 of the ST.

FCS_SSHS_EXT.1.3

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

Findings:	A large packet is defined in section 6.4 of the ST as any data packet in excess of 256KB. Such packets are dropped.

FCS_SSHS_EXT.1.4

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

Findings:	No optional characteristics are defined. The encryption algorithms are described in section 6.4 of the ST TSS as AES-CBC mode with 128-bit and 256-bit keys. This is consistent with FCS_SSHS_EXT.1.4 in section 5.3.2 of the ST.

FCS_SSHS_EXT.1.5

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

Findings:	No optional characteristics are defined. The public key algorithms are described in section 6.4 of the ST TSS as RSA. This is consistent with FCS_SSHS_EXT.1.5 in section 5.3.2 of the ST.

FCS_SSHS_EXT.1.6

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

Findings:	The integrity algorithms are described in section 6.4 of the ST TSS as HMAC-SHA1, HMAC-SHA2-256 and HMAC-SHA2-512. This is consistent with FCS_SSHS_EXT.1.6 in section 5.3.2 of the ST.

FCS_SSHS_EXT.1.7

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

Findings:	The key exchange algorithms are described in section 6.4 of the ST TSS as diffie-hellman-group14-sha1. This is consistent with FCS_SSHS_EXT.1.7 in section 5.3.2 of the ST.

FCS_SSHS_EXT.1.8

382 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.

383 The intention of FCS_SSHC_EXT.1.8 and FCS_SSHS_EXT.1.8 SFRs is to ensure that the TOE implements both thresholds. The NIT also acknowledges that it is possible that hardware limitation may prevent reaching data transfer threshold in less

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

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

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

Findings:	In section 6.4 of the ST, the TSS indicates that the TOE will rekey after 1 hour or after an aggregate of 1GB of data has been exchanged, whichever comes first. The TSS does not claim that there are hardware limitations on meeting the data threshold and therefore both can and will be tested.

4.1.3.2 Guidance Documentation

FCS_SSHS_EXT.1.4

384 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 further configuration is needed to ensure the SSH server conforms with the description in the TSS.

FCS_SSHS_EXT.1.5

385 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 further configuration is needed to ensure the SSH server conforms with the description in the TSS.

FCS_SSHS_EXT.1.6

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

Findings:	No further configuration is needed to ensure the SSH server conforms with the description in the TSS.

FCS_SSHS_EXT.1.7

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

Findings:	No further configuration is needed to ensure the SSH server conforms with the description in the TSS.

FCS_SSHS_EXT.1.8

388 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:	The thresholds are not configurable.

4.1.3.3 Tests

FCS_SSHS_EXT.1.2

Technical Decisions:	The following assurance activities have been modified by TD0339.

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

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

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

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

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

FCS_SSHS_EXT.1.3

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

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

FCS_SSHS_EXT.1.4

392 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a SSH connection from a remote client (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from

the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

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

FCS_SSHS_EXT.1.5

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

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

Technical Decisions: The following assurance activities have been modified by TD0412.

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

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

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

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

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

FCS_SSHS_EXT.1.6

Technical Decisions: The following assurance activities have been modified by TD0337.

397 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.

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

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

398 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.

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

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

FCS_SSHS_EXT.1.7

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

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

High-Level Test Description
Findings: PASS

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

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

FCS_SSHS_EXT.1.8

Technical Decisions: The following assurance activities have been modified by TD0336.

401 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.

402 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).

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

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

404 For testing of the traffic-based threshold the evaluator shall use an SSH client to connect to the TOE, and shall transmit data from and to the TOE within the active SSH session until the threshold for transmitted traffic is reached. The transmitted traffic is the total traffic comprising incoming and outgoing traffic.

405 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).

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

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

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

Note	These limits are not configurable for this TOE.

4.1.4 FCS_TLSC_EXT.2 TLS Client Protocol with authentication

4.1.4.1 TSS

FCS_TLSC_EXT.2.1

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

Findings:	In section 6.3.3 of the ST TSS, the TLS client ciphersuites are defined. These ciphersuites are consistent with the permissible set defined in the SFR. These ciphersuites are consistent with the claimed cryptographic components from FCS_COP.1.

FCS_TLSC_EXT.2.2

409 The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies if certificate pinning is supported or used by the TOE and how it is implemented.

Findings:	References identifiers for the FAZ audit log server are defined in section 6.3.3 of the ST TSS as being supplied by the admin using the Web GUI and CLI and can be an IP or a hostname. Both CN and SANs of the stated types are supported. Wildcards are supported in both the CN and SAN DNS type. Certificate pinning is claimed as not being supported.

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

Findings:	The TOE is not a distributed TOE.

FCS_TLSC_EXT.2.4

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

Findings:	Section 6.3.3 of the ST TSS claims the Supported Elliptic Curves extension is supported. Curve NIST P-256 is transmitted as the only supported curve ID.

FCS_TLSC_EXT.2.5

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

Findings:	In section 6.3.3 of the [ST], the author claims the "…TOE supports presentation of an X.509v3 client certificate for authentication as required by the FAZ Audit Server."

4.1.4.2 Guidance Documentation

FCS_TLSC_EXT.2.1

413 The evaluator shall check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

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

FCS_TLSC_EXT.2.2

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

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

FCS_TLSC_EXT.2.4

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

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

FCS_TLSC_EXT.2.5

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

Findings:	The client certificate is set using the "certificate" option in the "log fortianalyzer" configuration tree. The process for generating or loading this certificate can be found in the [ADMIN] guide Chapter 3.

4.1.4.3 Tests

FCS_TLSC_EXT.2.1

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

High-Level Test Description
Using a Lightship developed TLS server, force the TOE client to negotiate a specifically claimed ciphersuite.
Findings: PASS

Technical Decisions: The following assurance activities have been modified by TD0396.

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

419 Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

High-Level Test Description
Construct two X.509 certificates: one with an extendedKeyUsage with ‘serverAuth’ and another without. Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server and show that the X.509 certificate without the EKU fails.
Findings: PASS

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

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

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

High-Level Test Description
Using a TLS server, force the TOE client to attempt a handshake with a test server using the TLS_NULL_WITH_NULL_NULL (cipher ID 0x0000).
Findings: PASS

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

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

High-Level Test Description
Using a Lightship developed TLS server, force the TOE client to attempt a handshake with a test server advertising an incorrect TLS version.
Findings: PASS

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

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

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

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

d) If using DHE or ECDH, modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

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

Technical Decisions:	The following assurance activities have been modified by TD0289.

e) Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and ACK message. This is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.

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

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

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

FCS_TLSC_EXT.2.2

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

Test Not Applicable	The TOE is not a distributed TOE.

Technical Decisions:	The following assurance activities have been modified by TD0257.

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

a) Test 1: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.

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

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

b) Test 2: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.

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

c) Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

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

d) Test 4: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.

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

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

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

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

2) The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.

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

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

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

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

Note	The TOE does not support pinned certificates.

FCS_TLSC_EXT.2.3

425 Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function and demonstrate that the function succeeds. If the certificate is validated and a trusted channel is established, the test passes. The evaluator then

shall delete one of the certificates and show that the certificate is not validated and the trusted channel is not established.

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

FCS_TLSC_EXT.2.4

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

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

FCS_TLSC_EXT.2.5

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

427 *The purpose of these tests is to confirm that the TOE appropriately handles connection to peer servers that support and do not support mutual authentication.*

428 Test 1: The evaluator shall establish a connection to a peer server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.

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

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

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

High-Level Test Description
Findings: PASS

4.1.5 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

4.1.5.1 TSS

FCS_TLSS_EXT.1.1

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

Findings:	In section 6.3.2 of the ST TSS, the TLS server ciphersuites are defined. These ciphersuites are consistent with the permissible set defined in the SFR. These ciphersuites are consistent with the claimed cryptographic components from FCS_COP.1.

FCS_TLSS_EXT.1.2

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

Findings:	ST TSS section 6.3.2 explicitly states that the TOE will reject any protocol version other than TLS 1.1 and TLS 1.2.

FCS_TLSS_EXT.1.3

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

Findings:	Section 6.3.2 describes the key agreement parameters for DHE and ECDHE ciphersuites.

4.1.5.2 Guidance Documentation

FCS_TLSS_EXT.1.1

433 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 description in the TSS.

FCS_TLSS_EXT.1.2

434 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 description in the TSS.

FCS_TLSS_EXT.1.3

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

Findings:	The TOE can alter its Diffie-Hellman parameter size through the use of the "system global" table and modifying the "dh-params" parameter to the desired (claimed and permitted) bit-size as described in [SUPP] section "Enabling administrative access". The RSA certificate modulus size can be "configured" by installing a new certificate with the given RSA modulus. No other parameters are used to configure elliptic curves.

4.1.5.3 Tests

FCS_TLSS_EXT.1.1

436 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

437 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

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

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

High-Level Test Description
Findings: PASS

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

a) withdrawn

b) withdrawn

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

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

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

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

Technical Decisions: The following assurance activities have been modified by TD0342.

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

a) Correctly encrypt (D)TLS Finished message

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

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

The evaluator shall verify that the Finished message (handshake type hexadecimal 16) is sent immediately after the server's ChangeCipherSpec (handshake type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data

(unencrypted example in hexadecimal, 16 03 03 00 40 14 00 00 0c...), where '14' is the hexidecimal message type code in the verify_data header and '00 00 0c' is the verify_data field length. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

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

FCS_TLSS_EXT.1.2

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

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

FCS_TLSS_EXT.1.3

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

442 The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_TLSS_EXT.1.3. For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.

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

High-Level Test Description
Using a Lightship developed TLS client, connect to the TOE using a valid ECDHE ciphersuite and curve combination and verify that the public key size that comes back in the Server Key Exchange message matches the expected bit size for the chosen curve.

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

4.2 Identification and Authentication (FIA)

4.2.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

4.2.1.1 TSS

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

Findings:	In the ST TSS section 6.8, X.509 certificates are claimed to be processed during the handshaking process for TLS and IPSec. HTTPS is HTTP over TLS and is therefore included in the description implicitly. Regarding the check for extendedKeyUsage OIDs, the TOE will check for the OIDs it supports and expects. If additional OIDs are contained in the certificate, they are not checked.

4.2.1.2 Tests

445 The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev:

a) Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds.

Test 1b: The evaluator shall then delete one of the certificates in the presented chain (i.e. the root CA certificate or other intermediate certificate,

but not the end-entity certificate), and show that an attempt to validate an incomplete chain fails.

High-Level Test Description
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. 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 (if testing the TLS channel) or VPN peer (for IPSec) that sends back a certificate chain as part of the authentication process and show that the connection is accepted. Remove the root CA from the TOE trust store. Force the TOE to connect to a TLS server (if testing the TLS channel) or VPN peer (for IPSec) 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
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. Create an X.509 certificate with a 'notAfter' date in the past. Force the TOE to connect to a TLS server (if testing the TLS channel) or VPN peer (for IPSec) 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.

High-Level Test Description
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. 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.

High-Level Test Description
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
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. Load a CA into the trust store that is missing the CRLSigning purpose. Load the CRL for the corresponding CA. Show that the TOE prevents loading this CRL.
Findings: PASS

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

High-Level Test Description
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. 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 last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

High-Level Test Description
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. 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.

High-Level Test Description
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
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. 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

446 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.

Technical Decisions: The following assurance activities have been modified by TD0228.

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

448 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
Repeat for both TLS connections for the logging channel and the IPSec VPN peer.

High-Level Test Description
Clone the known good CA certificate and remove the basicConstraints extension. Replace the existing known-good CA with the cloned CA. Verify the TOE fails to load the certificate.
Findings: PASS

b) 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
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. Clone the known good CA certificate and set the basicConstraints extension to have the CA flag set to FALSE. Replace the existing known-good CA with the cloned CA. Verify the load fails.
Findings: PASS

c) Test 3: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.

Note:	This was performed in Test 1 to sanity check the setup and need not be performed again.

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

Findings:	This functionality was tested for TLSC and IPSec VPN peer authentication. They are distinct implementations.

4.2.2 FIA_X509_EXT.2 X.509 Certificate Authentication

4.2.2.1 TSS

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

Findings:	Section 6.8 in the ST TSS indicates there is a central certificate store for certificates to be stored and examined as part of the validation process.

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

Findings:	The TOE claims dynamic CRLs. The refreshing behaviour is described in the TSS in section 6.8 of the [ST]. The TOE caches the last status of the certificates in the CRL and uses the last known state if the CRL is unavailable at the next fetch time. There are no distinctions between channels that use the CRLs for revocation information.

4.2.2.2 Tests

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

453 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
Repeat for both TLS connections for the logging channel and the IPSec VPN peer. Show that if the CRL cannot be fetched, the TOE will validate the certificate based on the last cached information.
Findings: PASS

4.2.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests

4.2.3.1 TSS

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

Findings:	The ST claims “device specific information” and outlines the information claimed in section 6.8 of the ST TSS.

4.2.3.2 Guidance Documentation

Technical Decisions:	The following assurance activities have been modified by TD0333.

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

Findings:	In the [ADMIN] document in Chapter 3, there is a description for generating the CSR on the TOE “Generating a certificate signing request”. This section describes the fields that can be used along with any specifics about what the field can hold (eg. Subject Alternative Name).

4.2.3.3 Tests

Technical Decisions:	The following assurance activities have been modified by TD0333.

456 The evaluator shall perform the following tests:

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

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 response message, and demonstrate that the function succeeds.

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

4.3 Security management (FMT)

4.3.1 FMT_MOF.1/Functions Management of security functions behaviour

4.3.1.1 TSS

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

4.3.1.2 Tests

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

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

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

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

460 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.

461 Test 1 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior

authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The term ‘handling of audit data’ refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

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

462 Test 2 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data with prior authentication as security administrator. The effects of the modifications should be confirmed. The term ‘handling of audit data’ refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

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

463 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.

464 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.

465 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.

466 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.

467 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.

468 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.

4.4 IPS: Intrusion Prevention

4.4.1 IPS_ABD_EXT.1 Anomaly-Based IPS Functionality

4.4.1.1 TSS

469 The evaluator shall verify that the TSS describes the composition, construction, and application of baselines or anomaly-based attributes specified in IPS_ABD_EXT.1.1.

Findings:	In section 6.14 of the ST TSS, IPS rules can be composed via a specific format. These rules can be applied against traffic patterns for throughput, time-of-day, frequency and thresholds.

470 The evaluator shall verify that the TSS provides a description of how baselines are defined and implemented by the TOE, or a description of how anomaly-based rules are defined and configured by the administrator.

Findings:	Section 6.14 of the ST TSS indicates that these IPS rules are constructed using a specific format which can contain a series of rules.

471 The evaluator shall verify that each baseline or anomaly-based rule can be associated with a reaction specified in IPS_ABD_EXT.1.3.

Findings:	The ST TSS in section 6.14 confirms that rules can be associated with one of the following claimed reactions: permit, reset, block. These are consistent with the permissible reactions defined in IPS_ABD_EXT.1.3.

472 The evaluator shall verify that the TSS identifies all interface types capable of applying baseline or anomaly-based rules and explains how they are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used) they can be treated collectively as a distinct network interface.

Findings:	The ST TSS claims in section 6.14 that rules can be applied to any defined interface capable of receiving network traffic.

4.4.1.2 Guidance Documentation

473 The evaluator shall verify that the operational guidance provides instructions to manually create baselines or anomaly-based rules according to the selections made in IPS_ABD_EXT.1.1. Note that dynamic "profiling" of a network to establish a baseline is outside the scope of this PP.

Findings:	Anomaly-based rules are implemented partly as DoS policies in the TOE. In [ADMIN] Chapter 9, "Inside FortiOS: Denial of Service (DoS) Protection", DoS policies are described. Anomaly-based rules are also constructed using custom signatures. The signature construction is described in great detail in Chapter 25 "Security Profiles" > "Custom Application and IPS Signatures". Anomaly-based rules often employ frequency information regarding packets per second, bytes per second, etc. which use the "--rate" syntax. Time of day rules are placed on the firewall policy in which IPS sensors are placed. Time of day schedules are described in [ADMIN] Chapter 9 > "Object Configuration" > "Firewall Schedules".

474 The evaluator shall verify that the operational guidance provides instructions to associate reactions specified in IPS_ABD_EXT.1.3 with baselines or anomaly-based rules.

Findings:	Anomaly-based rules are also constructed using custom signatures. The signature construction is described in great detail in Chapter 25 "Security Profiles" > "Custom Application and IPS Signatures". When defining sensors, a variety of actions can be applied. Those that are consistent with the claims include "Pass", "Monitor" (eg. pass with logging), "Block" and "Reset" (for stateful protocols).

475 The evaluator shall verify that the operational guidance provides instructions to associate the different policies with distinct network interfaces.

Findings:	IPS sensors are applied to Firewall policies for inline interfaces as described in [ADMIN] Chapter 25 "Security Policies" > "Intrusion Prevention" > "Enabling IPS Scanning". Firewall policies are applied to network interfaces as per [ADMIN] Chapter 9 "Firewall" > "Firewall Policies". For promiscuous interfaces, one-armed sniffers are defined as per Chapter 21 "Networking" > "Interfaces" > "One-armed sniffer". IPS signatures are applied to a profile from within the network interface list instead of within the firewall policy.

4.4.1.3 Test

476 The evaluator shall perform the following tests:

477 Test 1: The evaluator shall use the instructions in the operational guidance to configure baselines or anomaly-based rules for each attributes specified in IPS_ABD_EXT.1.1. The evaluator shall send traffic that does not match the baseline or matches the anomaly-based rule and verify the TOE applies the configured reaction. This shall be performed for each attribute in IPS_ABD_EXT.1.1.

High-Level Test Description
Create a new signature that tests each of the claimed attributes. Transmit data at the TOE that should match the attribute and ensure that the action is performed.
Findings: PASS

478 Test 2: Repeat the test assurance activity above to ensure that baselines or anomaly-based rules can be defined for each distinct network interface type supported by the TOE.

High-Level Test Description
Execute the previous test case on an interface that has been configured as a VPN interface.
Findings: PASS

4.4.2 IPS_IPB_EXT.1 IP Blocking

4.4.2.1 TSS

479 The evaluator shall verify how good/bad lists affect the way in which traffic is analyzed with respect to processing packets.

Findings:	According to ST section 6.14, when good/bad IP lists are attached to policies, this list can be used to dictate how further processing is handled.

480 The TSS should also provide detail with the attributes that create a known good list, a known bad list, their associated rules, including how to define the source or destination IP address (e.g. a single IP address or a range of IP addresses).

Findings:	Section 6.14 indicates that single IP addresses, address groups or subnets can be used to create such lists. Note that the TOE does not associate other attributes with IP lists. IP addresses, groups and netblocks are completely distinct from other rules which can be associated with those IP address objects. IP objects can be combined with any other packet filtering and IPS rule characteristic.

481 The evaluator shall also verify that the TSS identifies all the roles and level of access for each of those roles that have been specified in the requirement.

Findings:	Section 6.14 in the ST identifies the “Administrator” profile as having full privileges to manage and configure IPS policies.

4.4.2.2 Guidance Documentation

482 The evaluator shall verify that the administrative guidance provides instructions with how each role specified in the requirement can create, modify and delete the attributes of a known good and known bad lists.

Findings:	The [ADMIN] guide in Chapter 9 “Firewall” > “Object Configuration” > “Addresses” describes the process for managing lists of IP address lists. Whether they are known-good or known-bad is dependent on the firewall policy action (eg. block bad, allow good).

4.4.2.3 Test

483 The evaluator shall perform the following tests:

484 Test 1: The evaluator shall use the instructions in the operational guidance to create a known-bad address list. Using a single IP address, a list of addresses or a range of addresses from that list, the evaluator shall attempt to send traffic through the TOE that would otherwise be allowed by the TOE and observe the TOE automatically drops that traffic.

Note:	Addressing lists are constructed using the same TSFI as for those that are used for firewall and VPN policies. This test has been conducted as part of the FWcPP testing (FW_RUL_EXT.1.2/FFW_RUL_EXT.1.3/FFW_RUL_EXT.1.4) as well as the VPN GW EP testing (see FPF_RUL_EXT.1.7 test cases). Furthermore, these test cases only make sense when the interface is being treated as an inline sensor (rather than a promiscuous interface) since the requirement is that the traffic be dropped.

High-Level Test Description
Using the WAN interface, attach a policy to scan for outbound connections to known malicious IP addresses.
Findings: PASS

485 Test 2: The evaluator shall use the instructions in the operational guidance to create a known-good address list. Using a single IP address, a list of addresses or a range of addresses from that list, the evaluator shall attempt to send traffic that would otherwise be denied by the TOE and observe the TOE automatically allowing traffic.

Note:	Refer to the previous test case for rationale. This functionality was previously tested as part of the FWcPP testing (FW_RUL_EXT.1.2/FFW_RUL_EXT.1.3/FFW_RUL_EXT.1.4) and the VPN GW EP testing (FPF_RUL_EXT.1.7). These test cases only make sense when the interface is being treated as an inline sensor (rather than a promiscuous interface) since the implied requirement is that the traffic would normally have been denied.

486 Test 3: The evaluator shall add conflicting IP addresses to each list and ensure that the TOE handles conflicting traffic in a manner consistent with the precedence in IPS_NTA_EXT.1.1.

High-Level Test Description
Create two address lists (one considered 'good' and one considered 'bad') with an overlapping subnet and send traffic from the outside workstation to the inside workstation using addresses that are contained in the overlapping segment. Verify that the action ordered first takes precedence.
Findings: PASS

4.4.3 IPS_NTA_EXT.1 Network Traffic Analysis

4.4.3.1 TSS

487 The evaluator shall verify that the TSS explains the TOE's capability of analyzing IP traffic in terms of the TOE's policy hierarchy (precedence).

Findings:	According to section 6.14 of the ST TSS, policies are applied in sequential order based on the sequence number associated with each policy. The administrator may change this sequence to suit their needs.

488 The TSS should identify if the TOE's policy hierarchy order is configurable by the administrator for IPS policy elements (known-good lists, known-bad lists, signature-based rules, and anomaly-based rules).

Findings:	The administrator may change this sequence to suit their needs.

489 Regardless of whether the precedence is configurable, the evaluator shall verify that the TSS describes the default precedence as well as the IP analyzing functions supported by the TOE.

Findings:	According to section 6.14 of the ST, a sequence number is assigned to each rule when created. Rules are enforced in sequential order based on this sequence number. If an administrator does not modify the sequence number, one is still assigned. There is also an implicit default deny rule that is enforced if no other policy is matched.

490 The TSS associated with this requirement is assessed in the subsequent assurance activities.

4.4.3.2 Guidance Documentation

491 The evaluator shall verify that the guidance describes the default precedence.

492 If the precedence is configurable. The evaluator shall verify that the guidance explains how to configure the precedence.

Findings:	IPS Sensors are groups of IPS signatures. The order in which the signatures are added to the sensor defines the precedence. This is explained in [ADMIN] Chapter 25 "Security Profiles" > Intrusion Prevention". "Each filter consists of a number of signatures attributes. All of the signatures with those attributes, and only those attributes, are checked against traffic when the filter is run. If multiple filters are defined in an IPS Sensor, they are checked against the traffic one at a time, from top to bottom. If a match is found, the unit takes the appropriate action and stops further checking."

	Furthermore, inline IPS sensors that are applied to firewall policies inherit the ordering of firewall policy rules that can affect the precedence of traffic analysis. As per Chapter 9 “Firewall”, firewall policy rules can be ordered explicitly by the administrator based on a unique policy ID.

4.4.3.3 Test

493 The testing associated with this requirement is assessed in the subsequent assurance activities.

Note:	No testing is defined.

4.4.4 IPS_NTA_EXT.1.2

4.4.4.1 TSS

494 The evaluator shall verify that the TSS indicates that the following protocols are supported:

- IPv4
- IPv6
- ICMPv4
- ICMPv6
- TCP
- UDP

Findings:	Section 6.14 identifies that the above protocols are supported by the TOE.

495 The evaluator shall verify that the TSS describes how conformance with the identified protocols has been determined by the TOE developer. (e.g., third party interoperability testing, protocol compliance testing)

Findings:	The ST TSS in section 6.14 claims that conformance is determined through compliance testing during development with changes being made to ensure conformance with the requirements.

4.4.4.2 Guidance Documentation

496 The Guidance associated with this requirement is assessed in the subsequent assurance activities.

Note:	No guidance activities are defined.

4.4.4.3 Test

497 The testing associated with this requirement is addressed in the subsequent test assurance activities.

Note:	No testing is defined.

4.4.5 IPS_NTA_EXT.1.3

4.4.5.1 TSS

498 The evaluator shall verify that the TSS identifies all interface types capable of being deployed in the modes of promiscuous, and or inline mode as well as the interfaces necessary to facilitate each deployment mode (at a minimum, the interfaces need to support inline mode). The TSS should also provide descriptions how the management interface is distinct from sensor interfaces.

Findings:	The ST claims in section 6.14 that all interfaces can be deployed as promiscuous or inline modes. A management interface is defined as any interface which does not have an IPS policy tied to it.

4.4.5.2 Guidance Documentation

499 The evaluator shall verify that the operational guidance provides instructions on how to deploy each of the deployment methods outlined in the TSS. The evaluator shall also verify that the operational guidance provides instructions of applying IPS policies to interfaces for each deployment mode. If the management interface is configurable the evaluator shall verify operational guidance explains how to configure the interface into a management interface.

Findings:	Promiscuous interfaces can be configured as per [ADMIN] Chapter 21 “Networking” > “Interfaces” > “One-armed sniffer”. Inline interfaces are the default stance. Management interfaces are a logical construct only even though devices have silkscreening identifying specific interfaces as “mgmt.” or “mgmt1” or “mgmt2”. Management interfaces are logically defined as having no IPS policy defined on them.

500 The evaluator shall verify that the operational guidance explains how the TOE sends commands to remote traffic filtering devices.

501 Note: the secure channel configurations between the TOE and the remote device would be discussed as per FTP_ITC.1 (if the ST author selects other interface types) and/or FTP_TRP.1 (for interfaces in management mode) in the base PP.

Findings:	The TOE does not transmit commands to a remote traffic filtering device. Rather, it receives traffic from a filtering device via a network mirroring port. See Chapter 21 in [ADMIN] under “Interfaces” > “One-armed sniffer”.

4.4.5.3 Test

502 The tests associated for this requirement have been completed in subsequent assurance activities in which promiscuous and inline interfaces are tested (e.g. tests for IPS_SBD_EXT.1.7) and in the requirement of FTP_ITC.1 (if the ST author selects

other interface types) and/or FTP_TRP.1 (for interfaces in management mode) in the base PP.

Note:	No testing is defined.

4.4.6 IPS_SBD_EXT.1.1 Signature-Based IPS Functionality

4.4.6.1 TSS

503 The evaluator shall verify that the TSS describes what is comprised within a signature rule.

Findings:	A signature rule follows a specific format according to section 6.14 of the ST TSS. This format contains a series of parameters that define characteristics of the rule.

504 The evaluator shall verify that each signature can be associated with a reaction specified in IPS_SBD_EXT.1.5.

Findings:	Each signature can be associated with a reaction (allow, drop, reset) as described section 6.14 of the ST.

505 The evaluator shall verify that the TSS identifies all interface types capable of applying signatures and explains how rules are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used) they can be treated collectively as a distinct network interface.

Findings:	The ST TSS claims in section 6.14 that rules can be applied to any defined interface capable of receiving network traffic.

4.4.6.2 Guidance Documentation

506 The evaluator shall verify that the operational guidance provides instructions with how to create and/or configure rules using the following protocols and header inspection fields:

- IPv4: Version; Header Length; Packet Length; ID; IP Flags; Fragment Offset; Time to Live (TTL); Protocol; Header Checksum; Source Address; Destination Address; and IP Options.
- IPv6: Version; traffic class; flow label; payload length; next header; hop limit; source address; destination address; routing header; home address options.
- ICMP: Type; Code; Header Checksum; and Rest of Header (varies based on the ICMP type and code).
- ICMPv6: Type; Code; and Header Checksum.
- TCP: Source port; destination port; sequence number; acknowledgement number; offset; reserved; TCP flags; window; checksum; urgent pointer; and TCP options.
- UDP: Source port; destination port; length; and UDP checksum.

507 The evaluator shall verify that the operational guidance provides instructions with how to select and/or configure reactions specified in IPS_SBD_EXT.1.5 in the signature rules.

Findings:	The [IPS] guide contains a syntax guide for all of the above attributes in section "Protocol Related Options". Of all of the attributes, only the IPv4 header length, the IPv4 flags, IPv4 fragment offset, IPv6 flow label, IPv6 traffic class, and ICMP "rest of header" fields are not addressable via the direct named fields or "dotted" notation syntax. Instead, the syntax provides the user the ability to directly access the header information fields using a byte-addressable array syntax. For example, the IPv4 flags are addressable as ip[6] & 0xe0 which gets the 6th byte in the IP header and masks the byte with 0xe0. Reactions are associated with custom rules by configuring a default action in the rule itself (using the 'action' setting in the rule) or by configuring a reaction at the sensor level when the rule is added to the sensor. See [ADMIN] Chapter 25 "Security Profiles" > Intrusion Prevention".

4.4.6.3 Test

508 The evaluator shall perform the following tests:

509 Test 1: The evaluator shall use the instructions in the operational guidance to test that packet header signatures can be created and/or configured with the selected and/or configured reactions specified in IPS_SBD_EXT.1.5 for each of the attributes listed below. Each attribute shall be individually assigned to its own unique signature:

- IPv4: Version; Header Length; Packet Length; ID; IP Flags; Fragment Offset; Time to Live (TTL); Protocol; Header Checksum; Source Address; Destination Address; and IP Options.
- IPv6: Version; traffic class; flow label; payload length; next header; hop limit; source address; destination address; routing header; home address options.
- ICMP: Type; Code; Header Checksum; and Rest of Header (varies based on the ICMP type and code).
- ICMPv6: Type; Code; and Header Checksum.
- TCP: Source port; destination port; sequence number; acknowledgement number; offset; reserved; TCP flags; window; checksum; urgent pointer; and TCP options.
- UDP: Source port; destination port; length; and UDP checksum.

510 Using packet sniffers, the evaluator will generate traffic to trigger a signature and using packet captures will ensure that the reactions of each rule are performed as expected.

High-Level Test Description
Create a series of rules, one per attribute for each of the given protocols along with a configured reaction: • IPv4: Version; Header Length; Packet Length; ID; IP Flags; Fragment Offset; Time to Live (TTL); Protocol; Header Checksum; Source Address; Destination Address; and IP Options.

High-Level Test Description
• IPv6: Version; traffic class; flow label; payload length; next header; hop limit; source address; destination address; routing header; home address options. • ICMP: Type; Code; Header Checksum; and Rest of Header (varies based on the ICMP type and code). • ICMPv6: Type; Code; and Header Checksum;. • TCP: Source port; destination port; sequence number; acknowledgement number; offset; reserved; TCP flags; window; checksum; urgent pointer; and TCP options. • UDP: Source port; destination port; length; and UDP checksum. Generate network traffic which will trigger the rules and show that the given reaction occurs.
Findings: PASS

511 Test 2: Repeat the test assurance activity above to ensure that signature-based IPS policies can be defined for each distinct network interface type capable of applying signatures as supported by the TOE.

High-Level Test Description
Execute the previous test case on an interface that has been configured as a promiscuous interface as well as a VPN interface.
Findings: PASS

4.4.7 IPS_SBD_EXT.1.2

4.4.7.1 TSS

512 The evaluator shall verify that the TSS describes what is comprised within a string-based detection signature.

Findings:	Section 6.14 of the TSS supports inspection of packet payload data and can inspect the data elements described in IPS_SBD_EXT.1.2. String-based detection patterns have certain placement restrictions as indicated in the TSS.

513 The evaluator shall verify that each packet payload string-based detection signature can be associated with a reaction specified in IPS_SBD_EXT.1.5.

Findings:	Each signature can be associated with a reaction (allow, drop, reset back to traffic source) as described section 6.14 of the ST.

4.4.7.2 Guidance Documentation

514 The evaluator shall verify that the operational guidance provides instructions with how to configure rules using the packet payload string-based detection fields defined in IPS_SBD_EXT.1.2. The operational guidance shall provide configuration instructions, if needed, to detect payload across multiple packets.

Findings:	The [IPS] guide contains a syntax guide to perform string-matching in payloads. This information is found in section “Payload related Options”.

	The TOE automatically detects payloads smuggled within multiple packets without additional configuration.

515 The evaluator shall verify that the operational guidance provides instructions with how to configure reactions specified in IPS_SBD_EXT.1.5 for each string-based detection signature.

Findings:	Reactions are associated with custom rules by configuring a default action in the rule itself (using the 'action' setting in the rule) or by configuring a reaction at the sensor level when the rule is added to the sensor. See [ADMIN] Chapter 25 "Security Profiles" > Intrusion Prevention".

516 The evaluator shall verify that the operational guidance provides instructions with how rules are associated with distinct network interfaces that are capable of being associated with signatures.

Findings:	IPS sensors are applied to Firewall policies as described in [ADMIN] Chapter 25 "Security Policies" > "Intrusion Prevention" > "Enabling IPS Scanning". Firewall policies are applied to network interfaces as per [ADMIN] Chapter 9 "Firewall" > "Firewall Policies".

4.4.7.3 Test

517 The evaluator shall perform the following tests:

518 Test 1: The evaluator shall use the instructions in the operational guidance to test that packet payload string-based detection rules can be assigned to the reactions specified in IPS_SBD_EXT.1.5 using the attributes specified in IPS_SBD_EXT.1.2. However it is not required (nor is it feasible) to test all possible strings of protocol data, the evaluator shall ensure that a selection of strings in the requirement is selected to be tested. At a minimum at least one string using each of the following attributes from IPS_SBD_EXT.1.2 should be tested for each protocol. The evaluator shall generate packets that match the string in the rule and observe the corresponding reaction is as configured.

- Test at least one string of characters for ICMPv4 data: beyond the first 4 bytes of the ICMP header.
- Test at least one string of characters for ICMPv6 data: beyond the first 4 bytes of the ICMP header.
- TCP data (characters beyond the 20 byte TCP header)
    i. Test at least one FTP (file transfer) command: help, noop, stat, syst, user, abort, acct, allo, appe, cdup, cwd, dele, list, mkd, mode, nlst, pass, pasv, port, pass, quit, rein, rest, retr, rmd, rnfr, rnto, site, smnt, stor, stou, stru, and type.
    ii. HTTP (web) commands and content:
        1. Test both GET and POST commands
        2. Test at least one administrator-defined strings to match URLs/URIs, and web page content.
    iii. Test at least one SMTP (email) state: start state, SMTP commands state, mail header state, mail body state, abort state.

iv. Test at least one string in any additional attribute type defined within [selection: [assignment: other types of TCP payload inspection];

- Test at least one string of UDP data: characters beyond the first 8 bytes of the UDP header;

- Test at least one string for each additional attribute type defined in [assignment: other types of packet payload inspection]]

High-Level Test Description

Create a series of rules, one per requirement, for each of the given protocols along with a configured reaction:
• ICMPv4: data beyond the first 4 characters of the ICMP header.
• ICMPv6: data beyond the first 4 characters of the ICMP header.
• TCP: data beyond the 20 byte TCP header:
  ○ At least one FTP command
  ○ HTTP GET command
  ○ HTTP POST command
  ○ URLs
  ○ Web page content
  ○ SMTP commands
• UDP: data beyond the first 8 bytes of the UDP header
 	
Generate network traffic which will trigger the rules and show that the given reaction occurs.

Findings: PASS

519 Test 2: The evaluator shall repeat one of the tests in Test 1 but generate multiple nonfragmented packets that contain the string in the rule defined.

High-Level Test Description
Using the ruleset defined in the previous test case, construct several non-fragmented packets containing matches and show they are all captured by the IPS filter.
Findings: PASS

520 Test 3: Repeat the test assurance activity above to ensure that signature-based IPS policies can be defined for each distinct network interface type capable of applying signatures as supported by the TOE.

High-Level Test Description
Execute the previous test case on an interface that has been configured as a promiscuous interface as well as a VPN interface.
Findings: PASS

4.4.8 IPS_SBD_EXT.1.3

4.4.8.1 TSS

521 The evaluator shall verify that the TSS describes how the attacks defined in IPS_SBD_EXT.1.3 are processed by the TOE and what reaction is triggered when these attacks are identified.

Findings:	According to section 6.14 of the ST TSS, the TOE is capable of detecting the attacks mandated by IPS_SBD_EXT.1.3 (among many others). The reaction is configurable by the administrator and can be set to Permit, Deny or Reset (for TCP).

4.4.8.2 Guidance Documentation

522 The evaluator shall verify that the operational guidance provides instructions with configuring rules to identify the attacks defined in IPS_SBD_EXT.1.3 as well as the reactions to these attacks as specified in IPS_SBD_EXT.1.5.

Findings:	The [ADMIN] guide in Chapter 25 “Security Profiles” > “Intrusion Prevention” > “Enabling IPS Scanning” describes the process by which either customized IPS signatures or pre-canned signatures can be applied to identify attacks. Reactions to the attacks are assigned as per any other IPS signature. They can be applied once the signature is assigned to the IPS Sensor.

4.4.8.3 Test

523 Test 1: The evaluator shall create and/or configure rules for each attack signature in IPS_SBD_EXT.1.3. For each attack, the TOE should apply its corresponding signature and enable it to each distinct network interface type capable of applying the signatures. The evaluator shall use packet captures to ensure that the attack traffic is detected by the TOE and a reaction specified in IPS_SBD_EXT.1.5 is triggered and stops the attack. Each attack should be performed one after another so as to ensure that its corresponding signature successfully identified and appropriately reacted to a particular attack.

High-Level Test Description
For each of the attacks named in the SFR, enable the pre-configured rule in the TOE and then perform the attack on all interfaces capable of applying the signatures. The TOE will catch the attack and perform the configured action.
Findings: PASS

4.4.9 IPS_SBD_EXT.1.4

4.4.9.1 TSS

524 The evaluator shall verify that the TSS describes how the attacks defined in IPS_SBD_EXT.1.4 are processed by the TOE and what reaction is triggered when these attacks are identified.

Findings:	According to section 6.14 of the ST TSS, the TOE is capable of detecting the attacks mandated by IPS_SBD_EXT.1.4 (among many others). The reaction is configurable by the administrator and can be set to Permit, Deny or Reset (for TCP).

4.4.9.2 Guidance Documentation

525 The evaluator shall verify that the operational guidance provides instructions with configuring rules to identify the attacks defined in IPS_SBD_EXT.1.4 as well as the reactions to these attacks as specified in IPS_SBD_EXT.1.5.

Findings:	The [ADMIN] guide in Chapter 9 “Firewall” > “inside FortiOS: Denial of Service (DoS) Protection” > “DoS Policies” describes the process by which pre-canned attacks can be applied to identify those listed attacks. Reactions to the attacks are assigned at the DoS policy itself (block or monitor).

4.4.9.3 Test

526 Test 1: The evaluator shall configure individual signatures for each attack in IPS_SBD_EXT.1.4. For each attack, the TOE should apply its corresponding signature and enable it to each distinct network interface type capable of applying signatures. The evaluator shall use packet captures to ensure that the attack traffic is detected by the TOE and a reaction specified in IPS_SBD_EXT.1.5 is triggered and stops the attack. Each attack should be performed one after another so as to ensure that its corresponding signature successfully identified and appropriately reacted to a particular attack.

High-Level Test Description
For each of the attacks named in the SFR, enable the pre-configured rule in the TOE and then perform the attack on all interfaces capable of applying the signatures. The TOE will catch the attack and perform the configured action.
Findings: PASS

5 Evaluation Activities for Security Assurance Requirements

5.1 ASE: Security Target

527 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).

5.2 ADV: Development

528 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.

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

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

531 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.

532 5.2.1.1 Evaluation Activity: The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

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

533 5.2.1.2 Evaluation Activity: The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

Findings:	See comments in the previous work unit.

534 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

535 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.

536 5.3.1.1 Evaluation Activity: The evaluator shall ensure the Operational guidance documentation is distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

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

537 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.

538 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:	The [SUPP] provides wording indicating that the Network Processing Unit (NPU) is not FIPS-validated and it must be turned off in section "Disabling NPU support".

539 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.

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

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

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

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

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

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

Findings:	See work unit [PP] 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]. Additional information regarding the use of claimed digital signatures is provided in Chapter 2 of the [ADMIN] guide. The process for manually upgrading the TOE is provided in [SUPP] and [ADMIN]. See work unit [PP] 5.3.1.4 for details as to what was covered by the EAs.

541 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.

542 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.

543 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.

544 5.3.2.4 Evaluation Activity: The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

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

545 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.

6 Vulnerability Assessment

546 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.

547 The developer shall provide documentation identifying the list of software and hardware components[4] that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

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

548 5.6.1.2 Evaluation Activity: The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

Findings:	The following sources of public vulnerabilities were considered in formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform key-word searches during the evaluation of the TOE. Hypothesis sources for public vulnerabilities were: - Fortinet security advisories (https://fortiguard.com/psirt) - NIST National Vulnerabilities Database (can be used to access CVE and US-CERT databases identified below): https://web.nvd.nist.gov/view/vuln/search - Common Vulnerabilities and Exposures: http://cve.mitre.org/cve/ https://www.cvedetails.com/vulnerability-search.php - Community (Symantec) security community: https://www.securityfocus.com/ - US-CERT: http://www.kb.cert.org/vuls/html/search - Tenable Network Security http://nessus.org/plugins/index.php?view=search

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

- Tipping Point Zero Day Initiative http://www.zerodayinitiative.com/advisories

- Offensive Security Exploit Database: https://www.exploit-db.com/

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

- Google

Type 1 Hypothesis searches were conducted on February 14, 2019 and included the following search terms:

- Fortinet;

- Fortigate;

- Linux kernel;

- TLS;

- OpenSSH;

- Apache;

- TCP

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

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

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

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