www.GossamerSec.com

Assurance Activity Report
for
Apriva MESA VPN v3.0

Version 0.2
July 18, 2023

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Revision History

Revision	Date	Authors	Summary
Version 0.1	06/27/23	Gossamer	Initial draft
Version 0.2	7/18/2023	Gossamer	Validation comments.

The TOE Evaluation was Sponsored by:

Apriva ISS, LLC.
7600 N. 16th St, Suite 230
Phoenix, AZ 85020

Evaluation Personnel:

- Cornelius Haley
- Ryan Hagedorn

Common Criteria Versions:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

Common Evaluation Methodology Versions:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017

Table of Contents

1. Introduction

This document presents evaluations results of the Apriva Mesa VPN v3.0 NDcPP22e/VPNGW12 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

This section provides equivalence arguments for the Common Criteria testing as well as Cryptographic CAVP testing.

1.1.1 Evaluated Platform Equivalence

The Target of Evaluation (TOE) is the Apriva MESA VPN version 3.0. The TOE is a hardware and software product on a Dell PowerEdge R750 2U Rackmount Server or Dell PowerEdge R650 1U Rackmount Server. These hardware servers may contain any of the following CPU.

- Intel® Xeon® Silver 4309Y
- Intel® Xeon® Silver 4316
- Intel® Xeon® Silver 4314
- Intel® Xeon® Gold 5315Y
- Intel® Xeon® Gold 5317
- Intel® Xeon® Gold 5318Y
- Intel® Xeon® Gold 5318N
- Intel® Xeon® Gold 5320
- Intel® Xeon® Gold 6326
- Intel® Xeon® Gold 6330
- Intel® Xeon® Gold 6330N
- Intel® Xeon® Gold 6336Y
- Intel® Xeon® Gold 6338N

These servers may also contain either of the following Network Interfaces.

- Intel X710-T4L Quad Port 10GbE BASE-T Adapter
- Broadcom 5720 Dual Port 1GbE BASE-T Adapter

The TOE was tested at Gossamer's test facility in Columbia Maryland. The evaluation performed full NDcPP22e testing on the Dell PowerEdge R750 2U Rackmount Server with the Intel® Xeon® Silver 4310 processor. This device included both network adapters identified above thus ensuring that both interfaces were tested. Both PowerEdge devices and all identified processors support the same Intel® micro-architecture (Ice Lake) and use the same executable, Apriva MESA VPN software image. The TOE implements all security features in software and does not rely upon hardware specific features, thus testing one device is sufficient.

1.1.2 CAVP Certificate Justification

The TOE is the Apriva MESA VPN version 3.0 which uses the following three cryptographic libraries.

- Apriva ISS OpenSSL implementation
  - TLS connection, SSH connection, Key generation and establishment,
  - IPsec IKE cryptography,
  - Random number generator

- Apriva ISS Kernel Crypto API implementation
  - IPsec ESP data authentication and encryption
  - IPsec ESP HMAC
- Apriva ISS libgcrypt implementation
  - Trusted updates, Product integrity

The following tables show the CAVP certificate associated with the cryptographic SFR from the Security Target.

Functions	Requirement	Standard	Certificate #
Encryption/Decryption			
AES CBC, CTR (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A ISO 19772	A3908
AES GCM (128 and 256 bits)			A3913
Cryptographic hashing			
SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	A3921
Keyed-hash message authentication			
HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	FCS_COP.1/KeyedHash	FIPS Pub 198-1	A3921
Cryptographic signature services			
RSA Digital Signature Algorithm (rDSA) (2048, 3072 bits)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 9796-2	A3921
Elliptic Curve Digital Signature Algorithm (P-256, P-384, P-521)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 14888-3	A3921
Random bit generation			
CTR_DRBG with sw based noise sources with a minimum of 256 bits of non-determinism	FCS_RBG_EXT.1	FIPS SP 800-90A ISO/IEC 18031:2011	A3908
Key generation			
RSA Key Generation (2048, 3072 bits)	FCS_CKM.1 FCS_CKM.1/IKE	FIPS Pub 186-4	A3921
ECC Key Generation (P-256, P-384, P-521)	FCS_CKM.1 FCS_CKM.1/IKE	FIPS Pub 186-4	A3921
Key establishment			
RSA	FCS_CKM.2	RSAES-PKCS1-v1_5	Verification by known good impl.
KAS ECC	FCS_CKM.2	NIST SP 800-56A Rev 3	A3921
FFC Schemes using safe-prime groups	FCS_CKM.2	NIST SP 800-56A Rev 3	Verification by known good impl.

Functions	Requirement	Standard	Certificate #
Encryption/Decryption			
AES CBC, GCM (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116	A3903

		NIST SP 800-38A ISO 19772	
Cryptographic hashing			
SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	A3907
Keyed-hash message authentication			
HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	FCS_COP.1/KeyedHash	FIPS Pub 198-1	A3907

Table 1 Apriva ISS Kernel Crypto API CAVP Certificates

Functions	Requirement	Standard	Certificate #
Cryptographic signature services			
RSA Digital Signature Algorithm (rDSA) (2048 bits)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 9796-2	A3911
Cryptographic hashing			
SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	A3911
Keyed-hash message authentication			
HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	FCS_COP.1/KeyedHash	FIPS Pub 198-1	A3911

Table 2 Apriva ISS libgcrypt CAVP Certificates

1.2 Evaluation Evidence

1.3 References

The following evidence was used to complete the Assurance Activities:

[ST] Apriva MESA VPN v3.0 Security Target

[CC-Guide] Apriva® MESA VPN Common Criteria Guidance

2. Protection Profile SFR Assurance Activities

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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

Section 6.1 of [ST] explains that for cryptographic keys, the act of generating, importing, exporting or deleting a key is audited. Within these audits the key is identified by name and the associated administrator account is identified.

The TOE is not distributed.

Component Guidance Assurance Activities: 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).

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.

Section "Auditable Event Messages" in the [CC-Guide] provides an example of each auditable event required by FAU_GEN.1. During testing, the evaluator mapped the entries in the tables in this section to the TOE generated events, showing that the section provides examples/descriptions of all required audit events.

The evaluator verified the administrative commands when performing all other guidance AAs. Specific references to commands can be found throughout this AAR.

Component Testing Assurance Activities: 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.

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.

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

The evaluator created a list of the required audit events. The evaluator then collected the audit event when running the other security functional tests described by the protection profiles. For example, the required event for FPT_STM_EXT.1 is Changes to Time. The evaluator collected these audit records when modifying the clock using administrative commands. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.2 Audit Data Generation (VPN Gateway) (VPNGW12:FAU_GEN.1/VPN)

2.1.2.1 VPNGW12:FAU_GEN.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 VPNGW12:FAU_GEN.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the audit mechanisms that the TOE uses to generate audit records for VPN gateway behavior. If any audit mechanisms the TSF uses for this are not used to generate audit records for events defined by FAU_GEN.1 in the Base-PP, the evaluator shall ensure that any VPN gateway-specific audit mechanisms also meet the relevant functional claims from the Base-PP.

For example, FAU_STG_EXT.1 requires all audit records to be transmitted to the OE over a trusted channel.

This includes the audit records that are required by FAU_GEN.1/VPN. Therefore, if the TOE has an audit mechanism that is only used for VPN gateway functionality, the evaluator shall ensure that the VPN gateway related audit records meet this requirement, even if the mechanism used to generate these audit records does not apply to any of the auditable events defined in the Base-PP.

Section 6.1 of [ST] indicate that the TSF stores the audit logs it generates as the following discrete local log types: iptables, secure, messages, quicksec, syslog, diag, boot, sysman, common. To prevent unauthorized access, modification, or deletion of the logs the TSF command line interface does not provide functions for users to directly access the /var/log directory. This section also states that all log types are protected local and securely forwarded to an external syslog server in the same manner.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it identifies all security-relevant auditable events claimed in the ST and includes sample records of each event type. If the TOE uses multiple audit mechanisms to generate different sets of records, the evaluator shall verify that the operational guidance identifies the audit records that are associated with each of the mechanisms such that the source of each audit record type is clear.

This activity has been addressed in conjunction with the guidance evaluation activities for VPNGW12:FPF_RUL_EXT.1 where the evaluator verified that the [CC-Guide] describes how to configure the TSF to

result in applicable network traffic logging. In particular, see VPNGW12:FPF_RUL_EXT.1.4 which describes the 'access-list' command and how it is used to specify the actions of deny, permit or log for a rule.

Component Testing Assurance Activities: The evaluator shall test the audit functionality by performing actions that trigger each of the claimed audit events and verifying that the audit records are accurate and that their format is consistent with what is specified in the operational guidance. The evaluator may generate these audit events as a consequence of performing other tests that would cause these events to be generated.

See NDcPP22e:FAU_GEN.1

2.1.3 User identity association (NDcPP22e:FAU_GEN.2)

2.1.3.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See NDcPP22e:FAU_GEN.1 where the activities for FAU_GEN.2 are already covered.

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

See NDcPP22e:FAU_GEN.1 where the activities for FAU_GEN.2 are already covered.

Component Testing Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

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.

Not applicable. The TOE is not a distributed TOE.

2.1.4 Protected audit trail storage (NDcPP22e:FAU_STG.1)

2.1.4.1 NDcPP22e:FAU_STG.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 NDcPP22e:FAU_STG.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally and how these records are protected against unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.

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

Section 6.1 of [ST] states that for all log types, the TOE restricts direct access to the audit log. This section explains that the TSF stores local logs in /var/log and to prevent unauthorized access, modification, or deletion of the logs the TSF command line interface does not provide functions for users to directly access the /var/log directory.

Section 6.1 explains that the TOE provides 10GB of dedicated storage for log storage. To manage this space, the TOE performs a log rotation operations for each log type. Section 6.1 describes that log rotation is based on time or file size, whichever comes first. Time based rotation occurs daily at an administrator-configured time. File size rotation occurs when the log files reach an administrator-configured size from 1MB to 1000MB, default 100MB. Each local log type is rotated independently of the other local log types. When rotating logs, the TSF compresses the active log file, creates a new log file, and checks to see if the maximum number of archives has been exceeded. If the maximum number of archives has been exceeded, the TSF deletes the oldest archive. The TSF keeps a default of seven archives. The TSF also performs a log rotate on all logs if the audit storage reaches 90% of capacity.

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it describes any configuration required for protection of the locally stored audit data against unauthorized modification or deletion.

The section entitled "Audit Logs" in [CC-Guide] indicates that the TOE only permits audit files to be viewed. Therefore, no configuration is required to protect locally stored audit data against unauthorized modification or deletion.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: The evaluator shall access the audit trail without authentication as Security Administrator (either by authentication as a non-administrative user, if supported, or without authentication at all) and attempt to modify and delete the audit records. The evaluator shall verify that these attempts 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 access the audit trail 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.

b) Test 2: The evaluator shall access the audit trail as an authorized administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.

For distributed TOEs the evaluator shall perform test 1 and test 2 for each component that is defined by the TSS to be covered by this SFR.

Test 1 & 2: The TOE does not offer commands for an administrator to clear or delete audit records. The evaluator logged in to the TOE and attempted to modify and delete audit records. The evaluator verified that these attempts were unsuccessful.

2.1.5 Protected Audit Event Storage (NDcPP22e:FAU_STG_EXT.1)

2.1.5.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.5.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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.

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

The evaluator shall examine the TSS to ensure that it details the behavior 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.

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 acceptable frequency for the transfer of audit data.

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).

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.

Section 6.1 of [ST] states that the TOE can be configured to export audit records from all log types to an external SYSLOG server. This communication is protected with the use of TLS. This section states that for all log types, the TOE restricts direct access to the audit log. This section explains that the TSF stores local logs in /var/log and to prevent unauthorized access, modification, or deletion of the logs the TSF command line interface does not

provide functions for users to directly access the /var/log directory. It further states that the TOE provides 10GB of dedicated storage for log storage in /var/log.

Section 6.1 explains that to manage this space, the TOE performs a log rotation operations for each log type. Section 6.1 describes that log rotation is based on time or file size, whichever comes first. Time based rotation occurs daily at an administrator-configured time. File size rotation occurs when the log files reach an administrator-configured size from 1MB to 1000MB, default 100MB. Each local log type is rotated independently of the other local log types. When rotating logs, the TSF compresses the active log file, creates a new log file, and checks to see if the maximum number of archives has been exceeded. If the maximum number of archives has been exceeded, the TSF deletes the oldest archive. The TSF keeps a default of seven archives. The TSF also performs a log rotate on all logs if the audit storage reaches 90% of capacity.

Section 6.1 states that the TOE is a single standalone device that stores audit data locally.

The TOE is not distributed.

Component Guidance Assurance Activities: 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.

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.

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.

The section entitled "Syslog" in [CC-Guide] provides the steps necessary to configure a TLS protected channel for use in transferring audit data to an external audit server (syslog server). This section provides the command to configure the IP address and TLS port where the TOE will attempt to find the syslog server. The section entitled "Install Trust Chains and Certificates/Keys" describe the commands to import, display and remove the trust anchor for the syslog server.

The section entitled "Syslog Server Requirements" explains that the transmission of audit logs to the external audit server occurs in real time, with each audit record transferred as it is generated. The section entitled "Syslog Server Requirements" states the Apriva MESA VPN communicate with an external syslog (audit) server by establishing a trusted channel between itself and the audit server that is protected using TLSv1.2. This statement defines a requirement on the audit server as being able to support 'syslog' communication. This also indicates that the trusted channel employs TLSv1.2, which is interpreted as another requirement on the audit server. These statements indicate that a Apriva MESA VPN can communicate with an audit server supporting the syslog and TLSv1.2 protocols.

The section entitled "Configure Logging" includes a discussion of how the audit files are periodically archived.

Component Testing Assurance Activities: 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 test 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.

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).

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.

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.

Test 1: The evaluator configured the system (per guidance) to securely transfer audit data. The evaluator then generated audit data and captured network traffic between the TOE and the external audit server (rsyslog version 8.16.0). The evaluator verified that the packet capture showed the audit data was not cleartext on the network.

Test 2: The evaluator configured the system (per guidance) to securely transfer audit data. The evaluator captured network traffic between the TOE and the external audit server, and continued to generate audit data until the local storage space on the TOE was exceeded. The evaluator verified that when the local audit storage is filled to the maximum, the existing audit data is archived, and the oldest archive in the rotation is dropped.

Test 3: Not applicable. The TOE does not claim FAU_STG_EXT.2/LocSpace.

Test 4: Not applicable. The TOE is not distributed.

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.2.1.1 NDcPP22e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.2 of [ST] indicates that the TOE supports RSA key generation with 2048 and 3072 key sizes, along with ECC key generation using curves P-256, P-384, and P-521. This is indicated by Section 6.2 that states the TOE implements asymmetric key generation supporting RSA key establishment (key size 2048/3072), ECC key establishment (curves P-256, P-384, and P-521), and FFC Safe Primes key establishment for IPsec communication as described in the section above. The TOE implements ECC key generation & Key establishment (with curves P-256, P-384, and P-521) as part of the TOE SSH Server. The TOE acts as a client for TLS, providing key generation as support for ECC Key establishment (with curves P-256, P-384, and P-521).

Component Guidance Assurance Activities: 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.

The “Create the CSR Key" section of the [CC-Guide] describes how to generate keys using the ***crypto key generate*** command to specify either an RSA key with key size 2048 bits, 3072 bits or an ECDSA key type with curve size P-256, P-384 or P-521.

This command associates a name for the key, which is used when generating the CSR (as described by the section entitled "Generate the CSR").

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

Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

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.

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

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

Key Generation for Finite-Field Cryptography (FFC)

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.

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

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

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.

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

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.

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

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

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

(TD0580 applied)

The TOE has been CAVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

2.2.2 Cryptographic Key Generation (for IKE Peer Authentication) - per TD0723 (VPNGW12:FCS_CKM.1/IKE)

2.2.2.1 VPNGW12:FCS_CKM.1.1/IKE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure that the TSS describes how the key-pairs are generated. 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;

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.

Section 6.2 of [ST] indicates that the TSF can generate keys in creation of a Certificate Signing Request (CSR). The TOE supports RSA key generation with 2048 and 3072 key sizes, along with ECC key generation using curves P-256, P-384, and P-521. Section 6.2 of [ST] indicates that the TOE implements all “shall” and “should” statements and does not implement any “shall not” or “should not” statements.

Component Guidance Assurance Activities: 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.

The "Create the CSR Key" section of the [CC-Guide] describes how to generate keys using the ***crypto key generate*** command to specify either an RSA key with key size 2048 bits, 3072 bits or an ECDSA key type with curve size P-256, P-384 or P-521.

This command associates a name for the key, which is used when generating the CSR (as described by the section entitled "Generate the CSR").

Component Testing Assurance Activities: For FFC Schemes using 'safe-prime' groups:

Testing for FFC Schemes using safe-prime groups is done as part of testing in FCS_CKM.2.

For all other selections:

The evaluator shall perform the corresponding tests for FCS_CKM.1 specified in the NDcPP SD, based on the selections chosen for this SFR. If IKE key generation is implemented by a different algorithm than the NDcPP key generation function, the evaluator shall ensure this testing is performed using the correct implementation.

The TOE has been CAVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

The FFC Schemes using safe-primes was tested against the public implementation of these schemes refer to FTP_TRP.1/Admin and FTP_ITC.1 for this testing.

2.2.3 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.3.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

(TD0580 applied)

The evaluator confirmed that the TOE Key Establishment schemes identified by FCS_CKM.2 correspond to the key generation schemes identified by FCS_CKM.1.1. Section 6.2 of [ST] includes Table 9, "Service, Protocol and Key Establishment Scheme Mapping" that maps the RSA, ECC and FFC 'safe-prime' key establishment schemes supported by the TOE to the communication channels (services) that can make use of the scheme. The table show that Remote administration and TLS protected syslog traffic can make use of ECC key establishment. It also shows that VPN communication using IPsec can make use of RSA, ECC and FFC 'safe-prime' key establishment schemes.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

The “Create the CSR Key" section of the [CC-Guide] describes how to generate keys using the ***crypto key generate*** command to specify either an RSA key with key size 2048 bits, 3072 bits or an ECDSA key type with curve size P-256, P-384 or P-521.

This command associates a name for the key, which is used when generating the CSR (as described by the section entitled "Generate the CSR").

The section entitled "Configure the SSH Service" in [CC-Guide] identifies the commands that are used by an administrator to generate SSH host keys used by the TOE.

Component Testing Assurance Activities: Key Establishment Schemes

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

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

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.

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.

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.

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.

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

Validity Test

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.

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).

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

RSA-based key establishment

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

FFC Schemes using 'safe-prime' groups

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

(TD0580 applied)

The TOE has been CAVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

The FFC Schemes using ‘safe-primes’ was tested against the public implementation of these schemes refer to FTP_TRP.1/Admin and FTP_ITC.1 for this testing.

2.2.4 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.4.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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 for2). 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.

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).

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.

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.

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.

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.

Section 6.2 of [ST] identifies the secret keys, private keys and Critical Security Parameters (CSP) used by the TOE and their storage location (either FLASH or RAM) in Table 9, "Service, Protocol and Key Establishment Scheme Mapping".

This section also states that the keys and CSP shown in Table 9 are either zeroized or overwritten with a new value when they are no longer needed by the TOE. Zeroization occurs as follows: 1) when deleted from FLASH, the previous value is overwritten once with zeroes; 2) when added or changed in FLASH, any old value is overwritten completely with the new value; and, 3) the zeroization of values in RAM is achieved by overwriting once with zeroes. FLASH and RAM are accessible only through the restricted CLI command set, which can be used only after successful login to the TOE. Since this restricted CLI command set does not offer any commands to view raw FLASH or RAM, the CSP identified in Section 6.2 cannot be viewed even by administrative users.

Component Guidance Assurance Activities: 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.

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

The TOE does not perform any delayed key destruction.

Component Testing Assurance Activities: None Defined

2.2.5 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS_COP.1/DataEncryption)

2.2.5.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 of [ST] states that the TOE performs AES encryption and decryption using CBC, CTR, and GCM mode with key sizes of either 128 or 256.

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

The section entitled "Configure the SSH Service" in [CC-Guide] describes how an administrator can choose the encryption algorithm used for SSH encryption. This section indicates the product supports aes128-ctr and aes256-ctr. This section further states that the ciphers command specifies the only cipher algorithms available for use as per NIAP certification (aes128-ctr and aes256-ctr).

The section entitled "Syslog" states Syslog enables the following two cipher suites only. The list of TLS ciphersuites is not configurable.

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289.

The section entitled "Syslog" in [CC-Guide] describes how an administrator can define the ECDSA curve used to negotiate TLS syslog sessions.

The section entitled "Configure Global VPN Settings" in [CC-Guide] describes how an administrator can specify the encryption used by the IKE SA and IPsec SA during VPN sessions and how to select the RSA or ECDSA ciphersuite that will be used to negotiate VPN sessions. The section "Create VPN Authentication Suites" of the [CC-Guide], describe how to configure the claimed algorithms. This includes AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 as well as the required hash sizes SHA-256, SHA-384, and SHA-512 in the IPsec proposal using the 'algorithm' command.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

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.

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.

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.

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.

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.

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].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of keys 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.

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].

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

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.

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

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]

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.

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

AES-GCM Test

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.

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.

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.

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

AES-CTR Known Answer Tests

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):

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.

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.

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.

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].

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

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

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]

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.

There is no need to test the decryption engine.

The TOE has been CAVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

2.2.6 Cryptographic Operation (AES Data Encryption/Decryption) (VPNGW12:FCS_COP.1/DataEncryption)

2.2.6.1 VPNGW12:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to require the ST author to make certain selections, but these selections are all part of the original definition of the SFR so no new behavior is defined by the PP-Module.

Section 6.2 of [ST] states that the TOE performs AES encryption and decryption using CBC, CTR, and GCM mode with key sizes of either 128 or 256 bits.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.7 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash)

2.2.7.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.2 of [ST] states that the TOE supports cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512 with message digest sizes 160, 256, 384, and 512. SHS hashing supports keyed-hashing and is used within several services including, VPN Communication, TLS-protected syslog, and SSH-protected remote administration. SHA-256 is used in conjunction with RSA signatures for verification of software image integrity.

Section 6.6 of [ST] explains that the libgcrypt cryptographic module is used for hashing in support of HMAC checks for trusted update and product integrity verification. Table 6 in [ST] shows the algorithms supported by libgcrypt.

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated. This section also states that several cryptographic parameters for SSH are not configurable, and identifies the default values for these parameters as:

- MAC algorithms - HMAC-SHA2-256 and HMAC-SHA2-512.

The section entitled "Syslog" states Syslog enables the following two cipher suites only. The list of TLS ciphersuites is not configurable.

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289.

The section entitled "Create VPN Authentication Suites" states that The NIAP evaluated configuration only allows the use of the following algorithms: AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 as well as the required hash sizes SHA-256, SHA-384, and SHA-512.

No configuration is necessary or possible in relation to the hashing for HMAC integrity checks performed by libgcrypt in support of trusted update and product integrity verification.

Component Testing Assurance Activities: 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.

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

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

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

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

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

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.

The TOE has been ACVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

2.2.8 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS_COP.1/KeyedHash)

2.2.8.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.2 of [ST] indicates that the TOE supports keyed-hash message authentication using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 using SHA-256/384/512 with 256/384/512 bit keys and blocks to produce a 256/384/512 output MAC.

Section 6.6 of [ST] explains that the libgcrypt cryptographic module is used for hashing in support of HMAC checks for trusted update and product integrity verification. Table 6 in [ST] shows the algorithms supported by libgcrypt.

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

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated. This section also states that several cryptographic parameters for SSH are not configurable, and identifies the default values for these parameters as:

- MAC algorithms - HMAC-SHA2-256 and HMAC-SHA2-512.

The section entitled "Syslog" states Syslog enables the following two cipher suites only. The list of TLS ciphersuites is not configurable.

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289.

The section entitled "Create VPN Authentication Suites" states that The NIAP evaluated configuration only allows the use of the following algorithms: AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 as well as the required hash sizes SHA-256, SHA-384, and SHA-512.

No configuration is necessary or possible in relation to the HMAC integrity checks performed by libgcrypt in support of trusted update and product integrity verification.

Component Testing Assurance Activities: 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.

The TOE has been ACVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

2.2.9 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS_COP.1/SigGen)

2.2.9.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 of [ST] indicates that the TOE supports the use of RSA with 2048 and 3072 bit key sizes, and ECDSA with a key size of 256 bits or greater for cryptographic signatures (specifically NIST curves P-256, P-384, or P-521).

Section 6.6 of [ST] explains that the libgcrypt cryptographic library is used to verify RSA digital signatures during TOE trusted updates and product integrity verification operations.

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

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated. This section also states that several cryptographic parameters for SSH are not configurable, and identifies the default values for these parameters as:

- Key exchanges are hard coded to ECDH-SHA2-NISTP256, ECDH-SHA2-NISTP384, and ECDH-SHA2-NISTP521.
- Public key algorithms - ECDSA-SHA2-NISTP256, ECDSA-SHA2-NISTP384, and ECDSA-SHA2-NISTP521.

The section entitled "Syslog" states Syslog enables the following two cipher suites only. The list of TLS ciphersuites is not configurable.

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289.

No configuration is necessary or possible in relation to the RSA Signature verification performed by libgcrypt in support of trusted update and product integrity verification.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

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.

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

Signature Verification Test

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.

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

The TOE has been ACVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

2.2.10 IPsec Protocol - per TD0633 (NDcPP22e:FCS_IPSEC_EXT.1)

2.2.10.1 NDcPP22e:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: 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.

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.

Section 6.2 of [ST] indicates that the TSF uses the Linux iptables service to perform IPsec Security Policy Database (SPD) as described in Section 6.5. Packets not processed by any rules are dropped by a hard-coded final rule that may also log the dropped packet, if configured.

Section 6.5 of [ST] explains that the TSF implements three different rule chains that can be applied to network traffic on the VPN Ingress, VPN Egress, or Management. Each chain is applied to a different traffic type; traffic addressed to the TOE (INPUT), traffic sent by the TOE (OUTPUT), and traffic passing through the TOE (FORWARD). The rules are applied in the order they appear. Each rule can be ACCEPT, DROP, or LOG. Traffic can be filtered by interface (based on the name of the interface), IP protocol (TCP, UDP), port range, and IP address range. FORWARD rules are applied to all traffic not addressed to the TOE, including decrypted VPN traffic.

Section 6.5 indicates the TSF implements support for IPv4, IPv6, TCP, and UDP traffic. Correct implementation of these protocols has been established via third-party interoperability testing with known-good implementations of these common networking protocols.

Section 6.5 indicates the TSF implements SPD BYPASS rules using FORWARD rules from one physical interface to another physical interface.

Section 6.5 indicates the TSF implements SPD PROTECT rules using FORWARD rules from a physical interface to the VPN. Since the FORWARD rules specify the VPN, a failure of the VPN results in the packets becoming undeliverable. If the VPN is functional but an SA has not been established, the TSF attempts to establish the SA by sending an IKE_SA_INIT message to the peer.

Section 6.5 indicates the TSF implements SPD DISCARD rules using a DROP rule on any of the rule chains. The TSF implements one hard-coded iptables rule that cannot be modified:

- DROP (and optionally LOG) any packets that are not matched by previous administrator- configured rules.

Guidance Assurance Activities: 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.

The section entitled "Configure Global VPN Settings" in [CC-Guide] provides instructions to configure vpn-settings that define the peers, restrictions and cryptographic parameters applied to VPN communications. The vpn-settings along with an access-list provide all of the functionality of the SPD. An access-list defines the packets that are permitted or dropped. For packets that are permitted, those that match the vpn-settings traffic selector are encrypted, otherwise they bypass encryption.

This section entitled “Set Access Policies” in [CC-Guide] explains that access policies allow the Apriva MESA VPN server to permit or deny packet traffic based on traffic type, source, or destination. This section refers the reader

to section entitled “Access Policy Management”. The section entitled “Access Policy Management” contains many subsections, which explain that Access policy management is accomplished using four CLI commands:

1. “ruleset” to create a rule set and enter the scope of the rule set,
2. “access-list” to create/modify a named ACL and enter the scope of that ACL,
3. “match” to create an ordered rule within a rule set or ACL,
4. “access-list” to assign an ACL to an interface (executed within the scope of the target interface).

Subsections contain instructions on the use of these commands.

The ordering of rules is implicitly described in this section as enforcement of access-lists followed by the enforcement of vpn-settings. However, the section entitled “Match Rules” states that match rules are ordered by an index value. Later material in the “Match Rules” section states that rules are evaluated in index order.

Testing Assurance Activities: 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.

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.

The evaluator created packet processing rules for PROTECT, BYPASS, and DISCARD on the TOE. The evaluator then caused network traffic through the TOE and verified that the TOE could process the network traffic accordingly.

2.2.10.2 NDcPP22e:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

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

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. 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 create'' final entry that discards packets that do not match any previous entries). The evaluator sends the packet and observes that the packet was dropped.

The evaluator configured the TOE with rules to drop, encrypt, and allow packets to flow in plaintext. The evaluator then attempted to establish a connection and confirmed that it was successful. The evaluator sent a packet matching the rules and observed the packet was successfully sent. The evaluator sent packets that did not match the rules as a part of FPF_RUL_EXT.1.5 (rules ordering).

2.2.10.3 NDcPP22e:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: 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).

Section 6.2 of [ST] states that the TOE only supports IKEv2 and ESP connections operating in tunnel mode.

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

In the "Configure Global VPN Settings" section of [CC-Guide], the subsection “Set VPN MODE” describes how to select the mode for the IPsec channel using the mode <headend|site2site|both> command.

Testing Assurance Activities: The evaluator shall perform the following test(s) based on the selections chosen:

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.

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.

Test 1: For this test, the evaluator alternately configured a test peer to require only tunnel mode. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful in tunnel mode.

Test 2: This test is not applicable as Transport mode was not claimed.

2.2.10.4 NDcPP22e:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4)/KeyedHash Cryptographic Operations (for keyed-hash message authentication) and if the SHA-based HMAC function truncated output is utilized it must also be described.

Section 6.2 of [ST] states that the TOE can be configured to use the AES-GCM-128, AES-GCM-256, AES-CBC-128, and AES- CBC-256 algorithms from the Red Hat Kernel Cryptographic Module for encryption and message authentication for IPsec ESP. When AES-CBC is negotiated as the symmetric cipher, the TOE supports HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512.

The evaluator found that the algorithms claimed in the ST selection were those listed in the TSS, and further found that these algorithms were included in the claims for FCS_COP.1/KeyedHash and FCS_COP.1/DataEncryption.

Guidance Assurance Activities: The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

The section "Create VPN Authentication Suites" of the [CC-Guide], describe how to configure the claimed algorithms. This includes AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 as well as the required hash sizes SHA-256, SHA-384, and SHA-512 in the IPsec proposal using the 'algorithm' command.

Testing Assurance Activities: 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.

For this test, the evaluator alternately configured a test peer to accept each of the algorithms claimed and supported by the TOE. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful with each of the supported algorithms.

2.2.10.5 NDcPP22e:FCS_IPSEC_EXT.1.5

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

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.

Section 6.2 of [ST] states that the TSF only supports IKEv2 and ESP connections operating in tunnel mode.

Guidance Assurance Activities: 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 for the following test (if selected).

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.

The Security Target indicates that the TOE only supports IKEv2. The section entitled "Create VPN Authentication Suites" in [CC-Guide] states that the MESA VPN utilizes Internet Key Exchange version 2 (IKEv2) which is the protocol used to set up a security association (SA) in the IPsec protocol suite. IKEv2 and Network Address Traversal (NAT) are automatically enabled and there are no other configurations necessary.

Testing Assurance Activities: 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.

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.

Test 1: Not applicable. The TOE does not support IKEv1.

Test 2: The evaluator configured the TOE such that a VPN session from a test server traversed a NAT device. The evaluator initiated an IPsec connection and observed that the TOE correctly negotiated the NAT connection to establish a protected IPsec connection.

2.2.10.6 NDcPP22e:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: 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.

Section 6.2 of [ST] explains that the TOE uses AES-CBC-128, AES-CBC-256, AES-GCM-128 or AES-GCM-256 to encrypt the IKEv2 payload and HMAC-SHA-256, HMAC-SHA-384, or HMAC-SHA-512 to authenticate the IKEv2 payload.

The evaluator found that the algorithms claimed in the ST selection were those listed in the TSS, and further found that these algorithms were included in the claims for FCS_COP.1/KeyedHash and FCS_COP.1/DataEncryption.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement.

The section entitled "Create VPN Authentication Suites" in [CC-Guide] describes how an administrator can configure the claimed algorithms including AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 as well as the required hash sizes SHA-256, SHA-384, and SHA-512 in the IPsec proposal using the 'algorithm' command.

Testing Assurance Activities: 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.

In turn, the evaluator configured the VPN to support each claimed IKEv2 encryption algorithm. For these tests the TOE and a test server were configured as peers and the evaluator configured each to use the same selected algorithms to establish the tunnel (the test was repeated for each algorithm).

2.2.10.7 NDcPP22e:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: 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.

Section 6.2 of [ST] explains that the TSF supports IKE_SA and Child_SA lifetime configurations. Either lifetime can be configured from 5 minutes to 24 hours. The IKE_SA has a default value of 24 hours, while the Child_SA has a default value of 8 hours.

The evaluator confirmed that the IKE selection in FCS_IPSEC_EXT.1.5, FCS_IPSEC_EXT.1.7 and FCS_IPSEC_EXT.1.8 are consistent.

Guidance Assurance Activities: 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, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g. configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours. 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.

The section entitled "Create VPN Authentication Suites" in [CC-Guide] describes how an administrator can configure the IKE SA lifetime limits using the **ike-lifetime seconds** *seconds* command, and IPsec SA lifetime limits using the **ipsec-lifetime kilobytes** *kilobytes* and **ipsec-lifetime seconds** *seconds* command. This section also explains that the IKE lifetime can be disabled by using the "no ike-lifetime seconds" command, but that an administrator must never configure the TOE this way when operating in a NIAP evaluated configuration.

Testing Assurance Activities: 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.'

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 a 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.

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the Phase 1 SA lifetime on the TOE.

Test 1: Not applicable. Number of bytes is not selected as an SA lifetime measure.

Test 2: The evaluator configured the TOE to have a 24 hour IKE and 8 hour ESP limits and the test peer was configured to have 25 hour IKE and 9 hour ESP limits. The evaluator then connected the IPsec VPN between the test peer and all instances of the TOE being tested and then waited for over 24 hours before terminating the test. The evaluator observed that the IKEv2 SA was renegotiated before the 24 hour mark and that the IKEv2 Child SA renegotiation occurred approximately every 8 hours before the 8 hours elapsed.

2.2.10.8 NDcPP22e:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: 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.

Section 6.2 of [ST] explains that the TSF supports IKE_SA and Child_SA lifetime configurations. Either lifetime can be configured from 5 minutes to 24 hours. The IKE_SA has a default value of 24 hours, while the Child_SA has a default value of 8 hours.

This section also indicate that by default, if no packets are processed by the Child_SA within the configured SA lifetime, the SA is closed. In this mode, the IKEv2 SA lifetime like acts as an idle timeout value. If any packets were processed through the IKEv2 SA tunnel, the tunnel is re-keyed instead. The administrator can also configure the IKEv2 tunnel to always rekey rather than drop the tunnel when no packets are processed.

The evaluator confirmed that the IKE selection in FCS_IPSEC_EXT.1.5, FCS_IPSEC_EXT.1.7 and FCS_IPSEC_EXT.1.8 are consistent.

Guidance Assurance Activities: 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, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g. configure a time value of 7h 45min to ensure the actual rekey is performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 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.

The section entitled "Create VPN Authentication Suites" in [CC-Guide] describes how an administrator can configure the IKE SA lifetime limits using the **ike-lifetime seconds** *seconds* command, and IPsec SA lifetime limits using the **ipsec-lifetime kilobytes** *kilobytes* and **ipsec-lifetime seconds** *seconds* command.

Testing Assurance Activities: 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.'

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

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 a 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.

Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE.

Test 1: The evaluator configured a maximum lifetime in number of bytes on the TOE and then while the VPN test peer was configured without a limit. The evaluator then established a connection with the VPN peer. The IPsec SA timed out after the packet size was exceeded and the connection reset. A new Phase 2 SA negotiation was required.

Test 2: The testing of 'length of time' as the SA lifetime measure was performed during the NDcPP21:FCS_IPSEC_EXT.1.7-t2 test activity which showed ESP rekeying before 8 hours elapsed.

2.2.10.9 NDcPP22e:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: 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.

Section 6.2 of [ST] explains that the TSF supports DH groups 14, 15, 19, 20 and 24 for use in IKEv2. One or more of these groups may be enabled by the administrator. The TSF will negotiate the algorithms in the following order if multiple are enabled: 20, 19, 15, 24, 14. The TSF generates the ephemeral private key (x) sizes used in Diffie-Hellman based on the negotiated group.

The evaluator determined that the set of Diffie-Hellman groups claimed by the requirement matched the set described in Section 6.2, and that the key sizes were correct for each group.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.10 NDcPP22e:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: 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.

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.

Section 6.2 of [ST] explains that the TSF generates and proposes nonces that are 256 bits long. The nonces are used in the IKEv2 key exchange for all cipher suites and are generated by the DRBG as defined in FCS_RBG_EXT.1. A 256-bit nonce is at least 128 bits and half the strength of the negotiated PRF hash.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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.

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.

See TSS assurance activity above.

2.2.10.11 NDcPP22e:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: 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.

The evaluator determined that the set of Diffie-Hellman groups claimed by the requirement matched the set described in Section 6.2, and that the key sizes were correct for each group.

Section 6.2 of [ST] states the TSF negotiates the allowed groups with the client in the IKEv2 exchange. The TSF will not allow the client to use a group that was not previously configured by the administrator.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement.

The section entitled "Create VPN Authentication Suites" in [CC-Guide] describes how an administrator can configure the DH groups claimed in the ST which are: DH groups 14, 15, 19, 20, and 24.

Testing Assurance Activities: 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.

Test 1: The evaluator made an IPsec connection to an IPsec peer using each of the claimed DH groups and confirmed that each resulted in successful connections.

2.2.10.12 NDcPP22e:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: 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.

Section 6.2 of [ST] explains that the security management interface ensures that the Key Size(s) configurable for a CHILD_SA are less than or equal to the Key Size(s) configured for the IKEv2 SA. If a client attempts to negotiate a CHILD_SA with a key size that has not been configured on the TSF, the connection will fail with a cipher-suite mismatch.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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.

b) Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a 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.

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.

d) Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a 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.

Test 1: The evaluator alternately configured a test peer to accept each of the claimed IKE hash algorithm. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connections were successful

Test 2: The evaluator configured a test peer to use a 128-bit key size for IKE and attempted to configure a 256-bit key size for ESP. The evaluator confirmed that the configuration was rejected by the TOE.

Test 3: The evaluator attempted to establish a connection with an unsupported algorithm/hash combination. The connection attempt failed.

Test 4: The evaluator attempted to establish a connection with an unsupported ESP algorithm. The connection attempt failed.

2.2.10.13 NDcPP22e:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: 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(2)/SigGen Cryptographic Operations (for cryptographic signature).

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.

Section 6.2 of [ST] indicates that the TOE authenticates peers using X.509 certificates. It also indicates that certificates can be RSA or ECDSA.

The [ST] does not include the selection to support pre-shared keys.

Guidance Assurance Activities: 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.

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.

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'.

The "Create the CSR Key" section of the [CC-Guide] describes how to generate keys using the ***crypto key generate*** command to specify either an RSA key with key size 2048 bits, 3072 bits or an ECDSA key type with curve size P-256, P-384 or P-521.

This command associates a name for the key, which is used when generating the CSR (as described by the section entitled "Generate the CSR").

The section entitled "Install Trust Chains and Certificates/Keys" in [CC-Guide] provide guidance on how the TOE supports X.509v3 certificates as defined by RFC 5280. Subsections within describe how to generate an RSA or ECDSA key pair, create and configure a CA trustpoint, import the CA certificate for the configured trustpoint, generate the CSR and import the certificate to the TOE.

The section entitled "Create VPN Authentication Suites" in [CC-Guide] states that the TOE does not support use of pre-shared keys for VPN authentication.

Testing Assurance Activities: For efficiency sake, the testing is combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1.

The TOE does not claim support for pre-shared keys. The majority of tests were conducted using ECDSA certificates. One additional successful connection was established demonstrating that the TOE was capable of performing peer authentication using RSA certificates.

2.2.10.14 NDcPP22e:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: 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.

Section 6.2 of [ST] includes a discussion of how the TOE compares the peer's presented identifier to the reference identifier. It indicates that the TOE supports authenticating a device connection using the Distinguished Name (DN) field of an X509 certificate. The DN contained in an x509 certificate presented during a VPN session negotiation must match the DN field types configured as required by the TOE (e.g., O, OU, C). Also, the common name (CN) field within the DN must be present in the list of approved device ids configured on the TOE. Only if the certificate's CN matches an approved device, and all DN field types match the TOE configuration, will the TOE accept the IPsec connection.

Guidance Assurance Activities: 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.

The section entitled "Add Devices to the Apriva MESA VPN Server" in [CC-Guide] states that the system supports the use of a whitelist of authorized devices. Whenever a whitelist is enabled, a device list must be created to serve as the index of approved devices that are granted access. The whitelist can also be modified to prevent a specific device from accessing the system before its certificate is revoked. A whitelist is used to implement the reference identifier - specifically the common name of the certificate. The ST requirement indicates that the DN is the reference identifier. This section explains how a whitelist can be used to specify the full DN required in a certificate in order to establish a VPN/IPsec connection. This includes a reference to the section entitled "Configure Distinguished Name Match Rules" where the commands to define specific DN matches are provided.

Testing Assurance Activities: 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:

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.

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.

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

a) Create a valid certificate with the CN so it contains the valid identifier followed by ''. 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.

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

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

a) 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.

b) 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.

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.

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:

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

b) Append '' to a non-CN field of an otherwise authorized DN.

Test 1: Not applicable. The TOE does not claim support for CN/identifier type combinations as reference identifier.

Test 2: Not applicable. The TOE does not claim support for SAN/identifier type combinations as reference identifier.

Test 3: Not applicable. The TOE does not claim support for CN/identifier type combinations as reference identifier.

Test 4: Not applicable. The TOE does not claim support for SAN/identifier type combinations as reference identifier.

Test 5: The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN. The evaluator observed that the TOE accepted the connection.

Test 6a: The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN except that the DN contained a duplicate CN field. The evaluator observed that the TOE did not accept the connection.

Test 6b: The evaluator sent a peer certificate signed by a trusted CA with a DN that matches an expected DN except that the DN contained a NULL character within the DN. The evaluator observed that the TOE did not accept the connection.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.11 IPsec Protocol - per TD0657 (VPNGW12:FCS_IPSEC_EXT.1)

2.2.11.1 VPNGW12:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 VPNGW12:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.3 VPNGW12:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.4 VPNGW12:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.5 VPNGW12:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.6 VPNGW12:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.7 VPNGW12:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.8 VPNGW12:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.9 VPNGW12:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.10 VPNGW12:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.11 VPNGW12:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.12 VPNGW12:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.13 VPNGW12:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.14 VPNGW12:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: All existing activities regarding 'Pre-shared keys' apply to all selections including pre-shared keys. If any selection with 'Pre-shared keys' is included, the evaluator shall check to ensure that the TSS describes how the selection works in conjunction with the authentication of IPsec connections.

The TOE does not support Pre-shared keys as an authentication method.

Component Guidance Assurance Activities: If any selection with 'Pre-shared Keys' is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

The TOE does not support Pre-Shared keys for use with IPsec, therefore the guidance documentation does not contain configuration instructions to specify pre-shared keys for authentication of an IPsec security association.

Component Testing Assurance Activities: None Defined

2.2.12 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.12.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

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.

Section 6.2 of [ST] indicates that the TOE uses one hardware-based entropy source to seed a software-based DRBG that complies with Special Publication 800-90 using CTR_DRBG. AES-256 is used in conjunction with a minimum of 256 bits of entropy for the seed. The Entropy description is provided in a separate (non-ST) document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Component Guidance Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

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

The TOE does not offer any configuration for the RNG functionality.

Component Testing Assurance Activities: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

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).

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.

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.

The TOE has been ACVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

2.2.13 SSH Server Protocol - per TD0631 (NDcPP22e:FCS_SSHS_EXT.1)

2.2.13.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.2 NDcPP22e:FCS_SSHS_EXT.1.2

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

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

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

Section 6.2 of [ST] states that the TOE supports both public-key (ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521) and password-based access mechanisms for user authentication. The evaluator confirmed this list corresponds with the FCS_COP.1/SigGen selections in [ST] which include ECDSA using curves P-256, P-384 and P-521.

Section 6.2 also states that administrators must associate a public key with each user account that is authenticated with public-keys. When an SSH session is established, the private key used by the remote user must be appropriate for the login ID which is provided during the SSH login. Only if the remote user possesses the correct private key for the login ID will the SSH login be successful.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test objective: The purpose of these tests is to verify server supports each claimed client authentication method.

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

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

Test 3: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication succeeds when the correct password is provided by the connecting SSH client.

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

(TD0631 applied)

Test 1 & 2: The evaluator configured a user to be able to login using the SSH interface with public-key based authentication, and observed the user login was successful. The evaluator then attempted to login using the SSH interface with public-key authentication without configuring a public key for that user and observed that the login attempt was not successful.

Test 3 & 4: The evaluator configured the TOE for password authentication on the SSH interface. The evaluator logged in using an SSH client and the correct password. The login was successful. The evaluator attempted an SSH connection using an invalid password. The evaluator was not able to log in.

2.2.13.3 NDcPP22e:FCS_SSHS_EXT.1.3

TSS Assurance Activities: The evaluator shall check that the TSS describes how 'large packets' in terms of RFC 4253 are detected and handled.

Section 6.2 of [ST] states that if the TOE receives an “SSH packet” larger than 262127 bytes from the TCP layer of the network stack, the TSF silently drops the packet. The TSF uses the packet length field in the SSH header to determine the packet length.

Guidance Assurance Activities: None Defined

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

The evaluator created and sent a packet to the TOE that was larger than the maximum packet size of 262128 bytes. The TOE rejected the packet and the connection was closed.

2.2.13.4 NDcPP22e:FCS_SSHS_EXT.1.4

TSS Assurance Activities: 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.

Section 6.2 of [ST] indicates that the TOE implements SSHv2 according to RFCs 4251, 4252, 4253, 4254, 4344, 5656, and 6668. No options included in the RFCs have been implemented. This section also indicates that the TOE supports SSHv2 with AES-CTR-128 or AES-CTR-256 for encryption within SSHv2.

The evaluator confirmed that FCS_SSHS_EXT.1.4 included the following selection: aes128-ctr and aes256-ctr.

Guidance Assurance Activities: 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).

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated.

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

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

The evaluator attempted to establish an SSH connection with each of the following SSH algorithms in the SFR to encrypt the session. The evaluator captured packets associated with each of the connection attempts and observed through testing that the TOE supports the following:

- aes128-ctr
- aes256-ctr

2.2.13.5 NDcPP22e:FCS_SSHS_EXT.1.5

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the SSH server's host public key algorithms supported are specified and that they are identical to those listed for this component. (TD0631 applied)

Section 6.2 of [ST] states that the TOE supports the same public key algorithms (ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521) for host authentication through SSHv2 as for user authentication. The evaluator confirmed that FCS_SSHS_EXT.1.5 included the same selections as specified in the TSS (i.e., ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521).

Guidance Assurance Activities: 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).

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated. This section also states that several cryptographic parameters for SSH are not configurable, and identifies the default values for these parameters as:

- Public key algorithms - ECDSA-SHA2-NISTP256, ECDSA-SHA2-NISTP384, and ECDSA-SHA2-NISTP521.

This section also explains how to generate SSH host keys for use by the TOE.

Testing Assurance Activities: Test objective: This test case is meant to validate that the TOE server will support host public keys of the claimed algorithm types.

Test 1: The evaluator shall configure (only if required by the TOE) the TOE to use each of the claimed host public key algorithms. The evaluator will then use an SSH client to confirm that the client can authenticate the TOE server public key using the claimed algorithm. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Has effectively been moved to FCS_SSHS_EXT.1.2.

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

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

(TD0631 applied)

Test 1: The evaluator attempted to establish an SSH connection with each of the following SSH public key algorithms. The evaluator captured packets associated with each of the connection attempts. The evaluator observed through testing that the TOE supports the following:

- ecdsa-sha2-nistp256,
- ecdsa-sha2-nistp384,
- ecdsa-sha2-nistp521

Test 2: The evaluator attempted to establish an SSH connection using ssh-dsa. The evaluator captured packets and was able to determine the connection attempt failed as expected.

2.2.13.6 NDcPP22e:FCS_SSHS_EXT.1.6

TSS Assurance Activities: 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.

Section 6.2 of [ST] states that the TOE supports SSHv2 with AES-CTR-128 or AES-CTR-256 for encryption; HMAC-SHA-256 or HMAC- SHA-512 for integrity; and ecdh-sha2-nistp256, ecdh-sha2-nistp384, or ecdh-sha2-nistp521 for key exchange. The evaluator confirmed that FCS_SSHS_EXT.1.5 included the same selections for integrity algorithms as those listed in this statement (i.e., HMAC-SHA-256 and HMAC- SHA-512).

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the 'none' MAC algorithm is not allowed).

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated. This section also states that several cryptographic parameters for SSH are not configurable, and identifies the default values for these parameters as:

- MAC algorithms - HMAC-SHA2-256 and HMAC-SHA2-512.

Testing Assurance Activities: 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.

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 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.

Test 1: The evaluator attempted to establish an SSH connection with each of the following SSH transport MAC algorithms. The evaluator captured packets associated with each of these connection attempts. The evaluator observed through testing that the TOE supports the following:

- hmac-sha2-256,
- hmac-sha2-512.

Test 2: The evaluator attempted to connect to the TOE using HMAC-MD5. The TOE rejects the attempt as expected.

2.2.13.7 NDcPP22e:FCS_SSHS_EXT.1.7

TSS Assurance Activities: 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.

Section 6.2 of [ST] states that the TOE supports SSHv2 with AES-CTR-128 or AES-CTR-256 for encryption; HMAC-SHA-256 or HMAC- SHA-512 for integrity; and ecdh-sha2-nistp256, ecdh-sha2-nistp384, or ecdh-sha2-nistp521 for key exchange. The evaluator confirmed that FCS_SSHS_EXT.1.5 included the same selections for supported key exchange algorithms as those listed in this statement (i.e., ecdh-sha2-nistp256, ecdh-sha2-nistp384, and ecdh-sha2-nistp521).

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

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated. This section also states that several cryptographic parameters for SSH are not configurable, and identifies the default values for these parameters as:

- Key exchanges are hard coded to ECDH-SHA2-NISTP256, ECDH-SHA2-NISTP384, and ECDH-SHA2-NISTP521.

Testing Assurance Activities: 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.

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.

Test 1 - The evaluator attempted to connect to the TOE using Diffie-Hellman-Group1. The TOE rejects the attempt as expected.

Test 2 - The evaluator attempted to establish an SSH connection with each of the key exchange methods in the list below. The evaluator captured packets associated with each of these connection attempts.

- ecdh-sha2-nistp256,
- ecdh-sha2-nistp384,
- ecdh-sha2-nistp521

2.2.13.8 NDcPP22e:FCS_SSHS_EXT.1.8

TSS Assurance Activities: The evaluator shall check that the TSS specifies the following:

a) Both thresholds are checked by the TOE.

b) Rekeying is performed upon reaching the threshold that is hit first.

Section 6.2 of [ST] states that the TOE initiates a rekey if 1GB of data is encrypted with a symmetric encryption key or shortly before 1 hour has elapsed, whichever occurs first.

Guidance Assurance Activities: 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.

The section entitled "Configure the SSH Service" in [CC-Guide] section explains how to configure ciphers used by the TOE SSH Sever, indicating that only aes128-ctr and aes256-ctr were evaluated. This section also states that several cryptographic parameters for SSH are not configurable, and identifies the default values for these parameters as:

- Rekey limits are set to 512 megabytes and 59 minutes.

Testing Assurance Activities: 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.

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).

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.

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

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).

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.

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(3)/Functions).

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

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

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

The evaluator connected to the TOE using an SSH client. The evaluator performed the rekey Time Limit test and found that the TOE initiated a rekey event at roughly the configured time (i.e., the TOE rekeyed at less than 1 hour).

The evaluator performed the DATA LIMIT rekey test found that the TOE rekeyed at below the required 1GB.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.14 TLS Client Protocol Without Mutual Authentication - per TD0634 & TD0670 (NDcPP22e:FCS_TLSC_EXT.1)

2.2.14.1 NDcPP22e:FCS_TLSC_EXT.1.1

TSS Assurance Activities: 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.

Section 6.2 of [ST] indicates that the TOE implements a TLSv1.2 client according to RFCs 4492, 5246, 5289, and 6125. The TSF supports the following ciphersuites:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

The evaluator confirmed that this list matches the list found in FCS_TLSC_EXT.1.1.

Guidance Assurance Activities: 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.

The section entitled "Syslog" states Syslog enables the following two cipher suites only. The list of TLS ciphersuites is not configurable.

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289.

Testing Assurance Activities: 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).

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 will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field, and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

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.

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

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

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

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

Test 5: The evaluator shall perform the following modifications to the traffic:

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

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

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

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

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

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

For the following tests the evaluator configured the test server to require mutual authentication and configured the TOE to establish a TLS session with this test server.

Test 1: The evaluator established a TLS session from the TOE to a test server with the test server configured to accept connections with only one of the claimed cipher suites. The evaluator used a network sniffer to capture the TLS session negotiation. The evaluator examined each traffic capture and observed that the expected TLS cipher was negotiated.

Test 2: The evaluator configured the test server to send a certificate with the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is accepted by the TOE. The evaluator reconfigured the test server to retry the TLS session using a cert that is missing the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is rejected by the TOE.

Test 3: The evaluator established a TLS session from the TOE. A modified test server negotiates TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, but returns a ECDSA-RSA Certificate. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is not negotiated successfully.

Test 4a: The evaluator configured a test server to offer only the TLS_NULL_WITH_NUL_NULL ciphersuite. The evaluator then attempted to establish a TLS session from the TOE to that test server. Using a network sniffer, the evaluator captured the TLS session negotiation and observed that the TLS session is rejected by the TOE.

Test 4b: The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to choose a ciphersuite that the TOE did not offer in its Client Hello handshake message.

Test 4c: The evaluator configured the TOE to connect to a test server using TLS with a TOE supported ECDHE key exchange method. The evaluator also configured the test server to accept that same ECHDE key exchange method, but to require a curve that was not supported by the TOE (i.e., P-192). The evaluator then observed the TOE reject negotiation.

Test 5a: The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to use a TLS version in the Server Hello that is a non-supported TLS version (version 1.4 represented by two bytes 0x0305). Please note the distinction between the TLS record layer version (which remained unchanged as version 1.2) and the TLS version within the Server Hello message (which indicates the Server's selected TLS version to govern the remaining handshake messages). The test requires alternation of the TLS version in the Server Hello message, not the TLS version in the TLS record layer. The evaluator verified that the client rejected the negotiation.

Test 5b: The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to modify the signature block in the Server's Key Exchange handshake message. The evaluator verified that the client rejected the negotiation.

Test 6a & 6b: The evaluator obtained a packet capture of the TLS session negotiation between the TOE (client) and a test server. The server implementation of the TLS protocol was modified as stated in the assurance activity 'a' and 'b'. The evaluator verified that the client did not finish the negotiation and no application data was transferred.

Test 6c: The evaluator obtained a packet capture of the TLS session negotiation between the TOE (client) and a test server. The server implementation of the TLS protocol was modified as stated in the assurance activity. The evaluator verified that the client rejected the Server Key Exchange handshake message.

2.2.14.2 NDcPP22e:FCS_TLSC_EXT.1.2

TSS Assurance Activities: 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.

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

If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE's conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC5952 for IPv6, RFC 3986 for IPv4) is enforced.

Section 6.2 of [ST] describes the TOE behavior in matching configured reference identifiers for the case when the identifier is configured as a DNS name and as an IP address. This explains for both cases how the configured names are matched against certificate fields. It also indicates that the TOE does support wildcards when used in the left-most label of a domain name and that IP addresses are not accepted in the CN field of a certificate.

The material in section 6.2 of [ST] states if the Syslog server is specified in the TOE configuration using a DNS name, the TSF automatically generates DNS-ID and CN-ID reference identifiers containing the specified DNS name that will be compared to values within the certificate received from the Syslog server. When the certificate presented by the Syslog server contains the SAN extension the TSF compares the DNS-ID from the TOE configuration to the DNS Name SAN fields. Otherwise, the TSF compares the CN-ID to the CN field(s). In both cases, the TSF performs the comparison as specified in Section 6.4 of RFC 6125, including support for a wildcard in the left-most label of the domain name.

If the Syslog server is specified in the TOE configuration using an IP address, the TSF automatically generates IP-ID and CN-ID reference identifiers containing the specified IP address that will be compared to values within the certificate received from the Syslog server. When the certificate presented by the Syslog server contains the SAN extension, the TSF verifies the IP-ID exactly matches an IP address SAN field. If there is not a SAN extension, the TSF will not establish a connection if the CN contains an IP address.

The TOE is not distributed.

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

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

The section entitled "Syslog" of [CC-Guide] provides instructions to configure a syslog server to accept audit data from the TOE. These instructions include an example of the command to identify the syslog server's: reference identifier. This section states that the value specified in the network destination host is considered the "reference identifier" that is matched against the SAN or CN as discussed in section "Syslog Server Requirements".

The section entitled "Syslog Server Requirements" describes the TOE approach to verifying the identity of the syslog server using the x509 certificate provided during TLS negotiations. This section states that the syslog server's certificate must contain a Subject Alternative Name (SAN), or Subject Common Name (CN) field if the SAN is empty. It also states that this must match the specified host address of the syslog server. The SAN can be used to match either DNS names or numeric IP addresses. The common name supports DNS name matching. The common name does not support using a numeric IP address.

The TOE is not distributed.

Testing Assurance Activities: Note that the following tests are marked conditional and are applicable under the following conditions:

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

or

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

or

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

Note that for some tests additional conditions apply.

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

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

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

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

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

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

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

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

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

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

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

2) [conditional]: The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds if wildcards are supported or fails if wildcards are not

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

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

Test 6 [conditional]: If IP address identifiers supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g.CN=*.168.0.1 when connecting to 192.168.0.1.20, CN=2001:0DB8:0000:0000:0008:0800:200C:*when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6). (TD0634 applied)

This negative test corresponds to the following section of the Application Note 64: 'The exception being, the use of wildcards is not supported when using IP address as the reference identifier.'

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

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

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

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

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

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

The TOE utilizes TLS for FTP_ITC.1 and FTP_TRP communications, therefore tests 1 through 6 are applicable.

The evaluator repeated tests 1 through 4 using a reference identifier that was either a DNS name or an IPv4 address.

Test 1: The TOE was configured to expect these reference identifiers in either the CN-ID or DN-ID. The evaluator then established a TLS session from the TOE targeting a server using a valid certificate with a CN matching the domain name used by the client. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed a successful connection. The evaluator then established a TLS session from the TOE targeting a server using a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 2: The evaluator established a TLS session from the TOE targeting a server using 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. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 3: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 4: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 5: This test was performed using DNS reference identifiers only. The evaluator tested hostname wildcards by configuring an expected DNS on the TOE with the test server's certificates configured with wildcard DNS names. The TOE successfully checked the hostname wildcards and behaved as expected. The evaluator used a network sniffer to capture the TLS session negotiation and observed that the TLS session was negotiated as shown in column 3 of the following table.

Certificate Contents	Host ID	Expected Result
CN=bar.*.example.com	bar.foo.example.com	No Connection
SAN=bar.*.example.com	bar.foo.example.com	No Connection
CN=*.example.com	foo.example.com	Successful Connection
SAN=*.example.com	foo.example.com	Successful Connection

CN=*.example.com	example.com	No Connection
SAN=*.example.com	example.com	No Connection
CN=*.example.com	bar.foo.example.com	No Connection
SAN=*.example.com	bar.foo.example.com	No Connection

Test 6: This test was performed using IPv4 reference identifiers only. The evaluator configured the TOE to connect with the GSS test server using TLS with the test server alternately configured with a certificate identifier as indicated in each test case below. The evaluator observed that the TOE connected when the identifier fulfilled the required rules, and the connection was rejected when the rules were not followed. The TOE does not support IPv6 addresses as a reference identifier.

Test 7: The TOE does not utilize TLS for FTP_ITT communication and therefore this test is not applicable.

2.2.14.3 NDcPP22e:FCS_TLSC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

Test 1: As part of testing FTP_ITC.1 Test 1, the evaluator loaded certificates needed to validate the certificate that was presented by an external entity and demonstrated that the function succeeds and a trusted channel was established.

Test 2: This test has been performed as part of several other test activities namely:

- Match the reference identifier -- Corresponds to FCS_TLSC_EXT.2.2 Tests 1 through 6.
- Validate certificate path -- Corresponds to FIA_X509_EXT.1/REV.1 Test 1.
- Validate expiration date -- Corresponds to FIA_X509_EXT.1/REV.1 Test 2.
- Determine the revocation status -- Corresponds to FIA_X509_EXT.2 Test 1.

Test 3: The TOE does not offer the ability to override certificate validation failures.

2.2.14.4 NDcPP22e:FCS_TLSC_EXT.1.4

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

Section 6.2 of [ST] states that the TSF sends the Supported Elliptic Curves extension with secp384r1 in its Client Hello message. This section also indicates that the TSF allows the configuration of the supported curves.

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

The subsection "Syslog" of [CC-Guide] provides an example of how to specify the ECDHE curve allowed to be used for the TLS connection to a syslog server. This section also indicates that the curve used for an ECDHE key exchange is secp384r1.

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

Test 1: The evaluator attempted to establish a TLS session between the TOE and a test server configured to allow only one key exchange method. The evaluator observed that the TOE was able to connect with the test server using the following key exchange methods.

- ECDHE w/ P-384 curve

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.15 TLS Client Support for Mutual Authentication - per TD0670 (NDcPP22e:FCS_TLSC_EXT.2)

2.2.15.1 NDcPP22e:FCS_TLSC_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.2 of [ST] indicates that the TOE sends its X.509 certificate in a Client Certificate message and signs a Certificate Verify message using the private key associated with the X.509 certificate to authenticate itself to the server. The evaluator confirmed that FIA_X509_EXT.2.1 indicated the TOE supports certificate authentication for TLS.

Component Guidance Assurance Activities: 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.

The section entitled "Certificate Signing Request Commands" in [CC-Guide] provides guidance for the creation of a Certificate Signing Request (CSR). This section indicates that the administrator can include device-specific information, Common Name, Organization, Organizational Unit, and Country values in the CSR, which will become part of the certificate issued by an external Certificate Authority (CA).

Component Testing Assurance Activities: For all tests in this chapter the TLS server used for testing of the TOE shall be configured to require mutual authentication.

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

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

The evaluator configured the Test server used during testing of FCS_TLSC_EXT.1 such that the server sent a Certificate Request message. The evaluator observed that the TOE responded to these requests by sending the configure Certificate to the server.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.3.1.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 NDcPP22e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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).

Section 6.3 of [ST] explains that the TSF maintains a separate failed authentication counter and lock flag for each remote (SSH) administrative user account. When a user attempts to establish an SSH session, the TSF checks if the lock flag has been set once the user has provided their username. If the account has been locked, the TSF does not process the authentication data and terminates SSH connection. Otherwise, the TSF processes the authentication attempt. A failed public key authentication attempt immediately followed by a failed username/password authentication attempt in the same SSH session is counted as a single failure, because most SSH clients automatically attempt public key authentication. A failed public key authentication attempt that is not followed by a username/password attempt is still counted as a failed authentication attempt. Each failed username/password authentication attempt is individually counted, with the exception of the case noted above. For each unsuccessful authentication attempt, the TSF increments the counter, compares the counter to the configured limit, and sets the lock flag if the counter has reached the configured limit. For each successful authentication attempt, the TSF resets the failed authentication counter to zero. Accounts can be unlocked only by a user over the local console connection.

This section also states that the TOE supports a ‘root’ account that is only allowed to login through the local console. Remote (SSH) login attempts to the root account are always rejected and cannot lock the ‘root’ account. Thus, the ‘root’ account can always login at the console and unlock other locked accounts.

Component Guidance Assurance Activities: 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.

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.

The section entitled "Configure the SSH Service" in [CC-Guide] provides an example of commands to specify the max-failed-attempts enforced by SSH. This section explain that this TOE mechanism locks an account such that SSH login for the locked account will fail, as well as a login attempt at the system console. It states that the root account is never locked out and is only available for use at the console.

Since the root account is accessible only through the console, failed SSH login attempts cannot lock the root account, allowing it to always be accessible at the console.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which remote administrators access the TOE (e.g. any passwords entered as part of establishing the connection protocol or the remote administrator application):

a) Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.

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

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

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

Test 1 & 2: The evaluator configured a limit on failed authentication attempts (i.e., 3 failures). The evaluator then performed more login attempts using incorrect credentials than the configured limit. The evaluator observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator then unlocked the account (using procedures from guidance) and observed that the user could login successfully with the correct password.

2.3.2 Password Management (NDcPP22e:FIA_PMG_EXT.1)

2.3.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of [ST] indicates that the TOE enforces an administrator configurable password length. The minimum password length may be set to 8 to 64 characters. The TSF supports passwords containing upper and lower case letters, digits and the following special characters '!', '@', '#', '$', '%', '^', '&', '*', '(', ')', '"', '+', ',', '-', '.', '/', ':', ';', '<', '=', '>', '?' , '[', '_']', '_', '`', '', '|', '~', '<space>'.

Component Guidance Assurance Activities: 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.

The section entitled "Create Superuser Users" in [CC-Guide] identifies the characters that can be used in a password along with instructions on how to change a password. The section entitled "Set Password Policies (optional)" explains that minimum password length can be set to a value between 8 and 64 characters.

Component Testing Assurance Activities: The evaluator shall perform the following tests.

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

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

The evaluator attempted to set/change a password for a user's account using several attempts. Throughout those attempts, every upper case letter, lower case letter, digit, and special character (as specified by the SFR in [ST]) were used in a password. The evaluator also confirmed that a minimum length of 8 was required by attempting to

set passwords with 7 characters (and observing the TOE reject the password) and of 8 characters (and observing that the TOE accepted the password change).

2.3.3 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.3.3.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

There are no preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

Component Testing Assurance Activities: 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.

Test 1- The evaluator observed during testing that passwords are obscured on the console login.

2.3.4 Password-based Authentication Mechanism (NDcPP22e:FIA_UAU_EXT.2)

2.3.4.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

See the evaluation assurance activity for NDcPP22e:FIA_UIA_EXT.1.

Component Guidance Assurance Activities: 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.

Evaluation Activities for this requirement are covered under those for NDcPP22e:FIA_UIA_EXT.1.

Component Testing Assurance Activities: 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.

See FIA_UIA_EXT.1

2.3.5 User Identification and Authentication (NDcPP22e:FIA_UIA_EXT.1)

2.3.5.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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'.

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.

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.

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.

Section 6.3 of [ST] contains a description of the logon process for local console and SSH-protected remote administrative sessions.

For local console logins this section explains that when a user connects to the console interface, the TSF prompts the user for a username and password. The TSF does not echo any characters back to the local console while the user is entering their password. If the username/password match an authorized administrator's credentials, the user is granted access to the command line interface.

For SSH-protected remote administrative sessions, when a user connects to the SSH interface, the TSF checks to see if the user proposed public key authentication. If the client proposed public key authentication, the TSF attempts to authenticate the user using the username and SSH_RSA (RFC 4253). If the SSH_RSA authentication fails or the client did not propose public key authentication, the TSF attempts to authenticate the client using a username/password. If either SSH_RSA authentication or username/password match an authorized administrator's credentials, the user is granted access to the command line interface.

Section 6.3 of [ST] also indicates that the only actions that are allowed before user identification and authentication are the following:

- Display the warning banner in accordance with FTA_TAB.1;
- Respond to ICMP Echo Request with an Echo Reply; and
- Respond with ICMP Destination Unreachable messages.

These actions match the actions identified by FIA_UIA_EXT.1.1.

The TOE is not distributed.

Component Guidance Assurance Activities: 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.

The section entitled "Configuration Steps" in [CC-Guide] contains subsections that lead an administrator to properly configure the TOE. The first "Initial Login" provide instructions to perform an initial log to secure the default root passwords and establish a new superuser account. The administrator is told to use the 'root' account only as an emergency measure. Subsequent subsections provide instructions for setting up changing the system boot password, creating superuser (admin) accounts, setting up session timeouts, login banners, password policies, network configurations and the SSH service.

Component Testing Assurance Activities: 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.

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.

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.

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.

The TOE support logins from the local console and via SSH. Both console and SSH support password-based authentication, while SSH also supports public-key authentication. At the local console and via a remote SSH connection, the evaluator demonstrated authentication using valid passwords were successful, while authentication using invalid passwords were rejected. The evaluator ensured that passwords words were obscured (FIA_UAU.7) and that a 'logout' operation could terminate an interactive login session (FTA_SSL.4). The evaluator also performed the same actions through a remote SSH connection authenticating with public-keys rather than passwords. The evaluator observed that the correct public key was accepted by the TOE and incorrect keys resulted in failed login attempt.

2.3.6 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/Rev)

2.3.6.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

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

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.

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.)

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.)

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.)

h) The following tests are run when a minimum certificate path length of three certificates is implemented.

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

Test 8a: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust

store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a valid chain of EC certificates (terminating in a trusted CA certificate), where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

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

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

(TD0527 12/2020 update applied)

The TOE validates certificates as part of the TLS Session establishment with a syslog server and as part of an IPsec connection to a VPN peer. The evaluator performed each of the following tests on these interfaces.

Test 1 -- The evaluator configured the TOE and a peer with valid certificates. The evaluator then attempted to make a connection between the peer devices using TLS protected syslog and IPsec VPN peer. A successful connection was made in both cases. The evaluator then configured the non-TOE system to present a certificate chain with an invalid certification path by deleting an intermediate CA so that the certificate chain was invalid because of a missing certificate. The connection between the TOE and peer was refused by the TOE.

Test 2 -- The evaluator used the TOE's TLS client and IPsec VPN to attempt connections to a network peer (syslog server or VPN peer). The test server presented a certificate during the TLS negotiation where the certificate was expired. The TOE rejected the connections.

Test 3 -- The evaluator used a test server to accept connection attempts from the TOE TLS client (syslog). The test server then presented a certificate during the TLS negotiation where the certificate was valid. A packet capture was obtained of this TLS negotiation which shows that the connection was successful and that the revocation checking (OCSP) was performed. The evaluator caused the test server to return a individually revoked certificate within the chain, followed by a cert issued by an intermediate CA that is revoked and attempted the same

connection from the TOE. Attempts using revoked certificates were not successful and revocation checking was performed.

The evaluator alternatively configured Strongswan on a test peer to accept connection attempts from the TOE (IPsec VPN). The test peer then presented a certificate during the IPsec negotiation where the certificate was valid. A packet capture was obtained of this IPsec negotiation which shows that the connection was successful and that the revocation checking was performed. The evaluator caused the test peer to return a individually revoked certificate within the chain, followed by a cert issued by an intermediate CA that is revoked and attempted the same connection from the TOE. Attempts using revoked certificates were not successful and revocation checking was performed. This test was repeated ensuring that the TOE performed revocation checking with OCSP and also with CRLs.

Test 4

OCSP The evaluator configured an OCSP responder to present a certificate that does not have the OCSP signing purpose. The evaluator established a session (TLS Syslog AND IPsec VPN) from the TOE client such that the TOE receives OCSP response signed by the invalid certificate and ensured that the session was not negotiated successfully.

CRL The evaluator also configured a CRL distribution point to present a certificate that does not have the CRLsign Key Usage. The evaluator established an IPsec VPN session with the TOE VPN peer such that the TOE receives the CRL signed by the invalid certificate and ensured that the IPsec VPN session was not negotiated successfully.

Test 5 -- The evaluator configured a test server to present a certificate that had a byte in the first eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 6 -- The evaluator configured a test server to present a certificate that had a byte in the last eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 7 -- The evaluator configured a test server to present a certificate that had a byte in the public key of the certificate modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 8a - The evaluator configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where the ECDSA certificates have a supported named curve. The TOE accepted the certificate as valid.

Test 8b - The evaluator then configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where an intermediate CA's ECDSA certificate has an explicit curve. The TOE rejected the certificate as invalid.

Test 8c - The evaluator first attempted to upload a valid subordinate CA certificate with the elliptic curve parameters specified as a named curve. Next, the evaluator attempted to upload a certificate to the TOEs trust store that uses an explicit format version of the elliptic curve parameters. The evaluator observed that the valid certificate with named-curves was successfully imported, while the certificate using an explicitly stated curve was not imported.

2.3.6.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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.

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

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).

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).

Test 1: The evaluator configured a test syslog server and a VPN peer to each present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test syslog server and the TOE IPsec implementation to connect to the VPN peer. The evaluator observed that the TOE rejected both connections.

Test 2: The evaluator configured the test syslog server and VPN peer to present a certificate chain containing a CA certificate having the basicConstraints section but with the cA flag not set (i.e., FALSE). The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test syslog server and the TOE IPsec implementation to connect to the VPN peer. The evaluator observed that the TOE rejected both connections.

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

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

Section 6.3 of [ST] states that the TSF verifies the validity of a certificate when an administrator loads a certificate into the TSF, when the TSF loads its certificates into memory, when IKEv2 receives a client certificate, and when Syslog/TLS receives a server certificate.

This section also explains that the TOE performs certificate validation for certificate import and TLS authentication using OCSP as described by the validation rules earlier in section 6.3. The TOE performs certificate validation for IPsec using OCSP and CRL as specified in RFC 5280 and as stated by the validation rules described earlier in section 6.3.

These validation rules further indicate that checks are performed on extendedKeyUsage and key usage when the certificate is used for specific purposes.

a. OCSP Signing is verified for any certificate used to sign an OCSP response
b. CRLsign Key Usage bit is verified for any certificate used to sign a CRL
c. TLS Server Authentication is verified for the certificate used to authenticate the Syslog server
d. IPsec Tunnel is verified for the certificates used to authenticate VPN peers.

These rules also state that administrators may import a trusted root certificate, only if the certificate is self-signed and the certificates has the basicConstraints flag value of TRUE.

Finally, these rules explain that the certificate's validity period is enforced, the certificate is checked for revocation status, the certificate path is checked, and the basic constraints flag is checked, as well as Key Usages are check.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

The section entitled "Syslog Server Requirements" in [CC-Guide] indicates that the TOE requires the TLS Syslog server to present an x509 certificate during TLS negotiations. This section also indicates that the Apriva MESA VPN server will validate this certificate, ensure it identifies a peer that matches the system's configuration and ensure (using OCSP) that the certificate has not been revoked. This section indicates that the certificate presented by the syslog server must include Server Authentication EKU and must chain to a trusted root certificate installed on the TOE.

This section also indicates that the validation process includes ensuring the certificate chains to a trusted root, the certificate matches the configured reference identifier, the certificate is still valid, the certificate key usage is appropriate and the BasicConstraints flags is set appropriately for the certificate type.

The section entitled "X.509 Certificate Key Usage OIDs" provides information regarding the validation of certificates for a TLS or IPsec VPN peer. It explains that the TOE validates the certificate by checking the expiration, issuer, signature and revocation status. Certificates must chain to a trusted root CA certificate and must contain appropriate Extended Key Usage values.

The section entitled "Syslog" indicates that Revocation checking is performed differently on TLS syslog and VPN IPsec connections. It describes the logic used by the TOE to perform revocation checking for TLS and IPsec. It indicates that TLS uses OCSP revocation checking, while IPsec uses both OCSP and CRL checking.

Component Testing Assurance Activities: None Defined

2.3.7 X.509 Certificate Validation (VPNGW12:FIA_X509_EXT.1/Rev)

2.3.7.1 VPNGW12:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

See the evaluation assurance activity for NDcPP22e:FIA_X509_EXT.1/Rev.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.8 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2)

2.3.8.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 NDcPP22e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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.

Section 6.3 of [ST] indicates that when validating presented certificates (peer certificates in IPsec, server certificates for TLS), if the revocation status of a certificate cannot be verified because the revocation server is unreachable, the TOE will either accept or reject the certificate based upon the TOE configuration. An administrator may configure the TOE behavior for IPsec-protected, VPN communication and for TLS-protected syslog connections.

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

The section entitled "Syslog" in [CC-Guide] provides instructions to configure the syslog destination, including instructions to define the targeted syslog host. The section entitled "Install Trust Chains and Certificate/Keys" discusses the concept of a cert chain between an end-entity certificate and a Root Certificate Authority certificate.

This section includes an example that depicts properly configuring the system for proper syslog usage. This section states that the "cert" command is used to identify the certificate that the Apriva MESA VPN will send to the syslog server during the TLS negotiation. This section states that the value specified in the network destination host is considered the "reference identifier" that is matched against the SAN or CN as discussed in section entitled "Syslog Server Requirements".

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

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

The evaluator established a trusted channel to a TLS audit server. For this channel, the TOE was an initiator of the connection from the TOE the audit server protected by TLS. The evaluator demonstrated that when the revocation server for the certificate presented by the remote test server was available, the TOE was successful in establishing the TLS session.

The evaluator then made the revocation server inaccessible and observed that the TOE was able to successfully establish connections with the audit server. Since this was the behavior claimed in the SFR selection for a TLS connection, this test passed.

The evaluator repeated this test using an IPsec connection to a VPN peer and observed the same passing results.

2.3.9 X.509 Certificate Authentication (VPNGW12:FIA_X509_EXT.2)

2.3.9.1 VPNGW12:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.2 VPNGW12:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to support its use for IPsec at a minimum. The

evaluator shall ensure that all evaluation of this SFR is performed against its use in IPsec communications as well as any other supported usage.

See the evaluation assurance activity for NDcPP22e:FIA_X509_EXT.2.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.10 X.509 Certificate Requests (NDcPP22e:FIA_X509_EXT.3)

2.3.10.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.10.2 NDcPP22e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.4 of [ST] indicates that the TOE allows the administrator to generate CSRs which contain:

- Public Key
- Common Name
- Country
- Email
- Locality
- Organization
- Organization Unit
- Serial Number (the signing CA may override this serial number; this is merely a requested serial, and is not the serial number of the TOE)
- State
- Subject Alternative Name

The administrator may configure the common name and Subject Alternative Name fields, and these fields do not contain any information that is outside the control of the administrator.

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

The section entitled "Certificate Signing Request Commands" in [CC-Guide] provides guidance for the creation of a Certificate Signing Request (CSR). This section contains subsections that provide instructions to create a CSR as well as to include device-specific information, Common Name, Organization, Organizational Unit, and Country values in the CSR, which will become part of the certificate issued by an external Certificate Authority (CA).

Component Testing Assurance Activities: 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.

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.

Test 1- The evaluator generated a certificate signing request by following the instructions in the guidance documentation for generating the request. The request was then exported to an external CA. While the CSR was within the CA, the evaluator examined the CSR and found that it included the fields identified in the Security Target. The CSR was also used to generate certificates on the CA which was returned to the TOE (also through a trusted path).

Test 2 - The evaluator tested that a certificate without an intermediate CA cannot be imported into the TOE manually.

2.3.11 X.509 Certificate Requests (VPNGW12:FIA_X509_EXT.3)

2.3.11.1 VPNGW12:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.2 VPNGW12:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

See the evaluation assurance activity for NDcPP22e:FIA_X509_EXT.3.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4 Security management (FMT)

2.4.1 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate)

2.4.1.1 NDcPP22e:FMT_MOF.1.1/ManualUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

There are no specific requirements for non-distributed TOEs.

This TOE is not distributed and the assurance activity states that there are no specific requirements for non-distributed TOEs.

Component Guidance Assurance Activities: 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).

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).

The section entitled "Software Update" in [CC-Guide] indicates that Software Updates to the system are performed manually. This section indicates that the system should be considered unusable during an update. The subsection

entitled "Initiating Update Process" explains that the software update process is initiated by logging into the system as administrator and executing the “update system repo” command.

The TOE is not distributed.

Component Testing Assurance Activities: 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 should fail.

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.

This test case is covered by the tests for FPT_TUD_EXT.1 and FIA_UIA_EXT.1 testing.

2.4.2 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData)

2.4.2.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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

Section 6.4 of [ST] states that the TOE does not allow any administrative actions to be performed prior to authentication of the administrative user. Once the administrative user is authenticated, the TSF grants the user access to a restricted command-line shell. This shell restricts the administrative users to commands required for administering the TSF while preventing users from running general-purpose Linux commands.

Section 6.4 of [ST] lists the administrative functions available to an authorize administrator. As indicated by section 6.3 of [ST] the only actions that are allowed before user identification and authentication are the following:

- Display the warning banner in accordance with FTA_TAB.1;
- Respond to ICMP Echo Request with an Echo Reply; and

- Respond with ICMP Destination Unreachable messages.
- Therefore, none of the administrative functions listed in Section 6.4 are available prior to login.

Section 6.8 of [ST] states that the TOE provides assured identification of non-TSF endpoints (for both TLS and IPsec) by validating X.509 certificates. The TOE implements a trust store containing trust anchors which it uses to verify identities of those non-TSF certificates. Since the list of administrative functions includes the capabilities to manage trusted CAs, the TOE is restricting management of the trust store using the administrative login mechanism.

Component Guidance Assurance Activities: 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.

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

Specific sections of the [CC-Guide] along with specific admin commands are identified or referenced throughout this AAR with the requirement to which they apply.

All accounts with superuser privilege are administrative accounts. The section entitled "Create Superuser Users" in [CC-Guide] states that at least one Superuser (superuser) user must be created for system configuration.

See NDcPP22e:FIA_X509_EXT.2.2 which identifies the sections in the Guide which describe how the administrator can configure and maintain the trust store including loading of CA certificates.

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

No separate testing for FMT_MTD.1/CoreData is required.

2.4.3 Management of TSF Data (NDcPP22e:FMT_MTD.1/CryptoKeys)

2.4.3.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see chapter 2.4.1.1.

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

Section 6.4 of [ST] states that only the authorized administrator can configure cryptographic keys. The keys an authorized administrator can manage consist of importing trusted Root CA certs, generating SSH host keys, importing SSH public keys, and loading X.509 certificates. All of these keys can be also be deleted.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

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

Section " Install Trust Chains and Certificates/Keys" describes trust chains and Certificate as used for both VPN (IPsec) and TLS communication. This section indicates that the administrator can generate and delete keys that can be used during the generation of a CSR. This section states that the CSR can be used to obtain and import a certificate that is signed by an external CA. This section explains that once a certificate is imported into the TOE, the administrator can delete but not modify the certificate.

Two subsections are provided present information and instructions to create CSR for use with both VPN (IPsec) and TLS communication. These subsections identify how the keys associated with the Certificates and Trust chains used with IPsec and TLS connections can be managed by an administrator. These subsections provide instructions on how to perform import, delete and display actions.

The section entitled "Configure the SSH Service" in [CC-Guide] states that the Apriva MESA VPN HostKey is automatically generated during system generation. There are, however, two commands that can be used to manage the SSH HostKey. The HostKey Show command displays the current HostKey value, whereas the HostKey Generate command replaces the existing HostKey with a new generated key. The section also contains instructions to use these commands.

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

Testing in FIA_UIA_EXT.1_t3 demonstrated that no actions other than viewing the login banner were available prior to authentication by the Security Administrator. Any attempts to modify, delete, generate, or import a

cryptographic key before being authenticated as an administrator failed. The evaluator then completed a login and demonstrated the cryptographic action of deleting a saved TOE private key.

2.4.4 Management of TSF Data (VPNGW12:FMT_MTD.1/CryptoKeys)

2.4.4.1 VPNGW12:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

See the evaluation assurance activity for NDcPP22e:FMT_MTD.1/CryptoKeys.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4.5 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1)

2.4.5.1 NDcPP22e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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(1)/ManualUpdate, FMT_MOF.1(4)/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(2)/Services, and FMT_MOF.1(3)/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.

(containing also requirements on Guidance Documentation and Tests)

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).

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.

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.

Several sections of [ST] including (section 6.4, 6.8 and 6.3) indicates the TOE provides a command line interface (CLI) interface via a local console or remotely through an SSH-protected session. Section 6.4 states that when the TSF grants access to an administrative user using SSH protocol or the console, the user has read only access to non-sensitive data. Authorized administrators must run a separate “enable” command, enter an additional password, and be assigned the authorized administrator (exec) privilege to gain access to privileged mode. Section 6.4 also provides the following list of functions to that are available to authorized administrators through the CLI, all of which encompass the functions identified by NDcPP22e:FMT_SMF.1 and VPNGW12:FMT_SMF.1.

- Configure the access banner
- Configure the remote administrator inactivity timeout
- Configure the local administrator inactivity timeout
- Initiate an update to the software
- Manage the failed authentication lockout threshold
- Start and Stop Services
- Configure audit trail archiving
- Configure Syslog server connectivity
- Manage cryptographic keys
    - Generate CSR (and RSA or ECDSA key pair)
    - Load a private key (associated with an X.509 certificate)
    - Generate SSH Host Key (ECDSA key pair)
- • Manage IPsec security parameters
    - Configure IKEv2 SA lifetimes
    - Configure IKEv2 algorithms
    - Mange IKEv2 Session Establishment restrictions
    - Configure IPsec ESP algorithms
- Unlock account (local console only)
- Manually set the time
- Ability to configure the reference identifier for the peer
    - Configure IP address assignment to VPN clients
    - Configure the Syslog server's reference identifier

- Manage trusted CAs
  - Import X.509 Certificates into the TOE's trust store and designate X509.v3 certificates as trust anchors
  - Delete trust anchor certificates from the TOE's trust store
- Import an X.509 Certificate for use by the TOE
- Load and assign SSH public key for user authentication
- Manage administrator accounts
- Manage minimum password length
- Configure Packet filtering rules
- Defining Packet filtering rules
- Ordering Packet filtering rules
- Assigning Packet filtering rules to interfaces

The evaluator found that all of these functions were present in the TOE interface.

The evaluator determined that both the TSS and guidance document describe the local administrative interface as well as the remote SSH-protected interface as providing the CLI used by administrators.

The TOE is not distributed.

Component Guidance Assurance Activities: See TSS Assurance Activities

The TOE is compliant with all requirements in the ST as identified in this report.

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

All TOE security functions are identified in the guidance documentation and have been tested as documented throughout this AAR.

2.4.6 Specification of Management Functions (VPNGW12:FMT_SMF.1/VPN)

2.4.6.1 VPNGW12:FMT_SMF.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the TSS identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

Section 6.4 also provides a list of functions to that are available to authorized administrators through the CLI, the following entries in that list correspond to the functions identified by VPNGW12:FMT_SMF.1.

- Configure Packet filtering rules
- Defining Packet filtering rules
- Ordering Packet filtering rules
- Assigning Packet filtering rules to interfaces

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the operational guidance identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

The section entitled "Access Policy Management" in [CC-Guide] describes how an administrator can define packet filter rules using rule sets, access lists and match rules. This section indicates how match rules can be ordered to establish their priority. The command to associate an access-list with an interface is provided in the sections entitled "Configure the Management Network Interface" and "Configure the Management Network Interface".

Component Testing Assurance Activities: The evaluator tests management functions as part of performing other test EAs. No separate testing for FMT_SMF.1/VPN is required unless one of the management functions in FMT_SMF.1.1/VPN has not already been exercised under any other SFR.

Testing of VPN management functions was performed while testing VPNGW12:FPF_RUL_EXT.1.

2.4.7 Restrictions on Security Roles (NDcPP22e:FMT_SMR.2)

2.4.7.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.7.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.7.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.4 of [ST] indicates the TOE maintains a single role for all administrative users. When the TSF grants access to an administrative user using SSH protocol or the console, the user has read only access to non-sensitive data. Authorized administrators must run a separate “enable” command, enter an additional password, and be assigned the authorized administrator (exec) privilege to gain access to privileged mode.

Component Guidance Assurance Activities: 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.

The section entitled "Introduction" in [CC-Guide] indicates that the TOE offers an administration CLI interface at a local console and remotely via an SSH. The section entitled "Initial Login" instructs the administrator to use the system console, attach a VGA monitor and a USB keyboard to the ports on the back of the Apriva MESA VPN server and use them to login to the system. The section entitled "Configure the SSH Service" describes the ssh-server command that can be used to configure the SSH interface offered by the TOE. This section indicates that the ssh subcommand "ciphers" specifies the only cipher algorithms available for use as per NIAP certification (aes128-ctr and aes256-ctr). Later in this same section a list of SSH configuration items that supported by default and which are not configurable within the MESA VPN is provided. These include key exchanges, MAC algorithms, public key algorithms, and rekey limits.

Component Testing Assurance Activities: 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.

The TOE is administered through either an SSH protected CLI, or the console CLI. These two were used for varying configuration operations. The CLI (both console and SSH) were used for all testing.

2.5 Packet Filtering (FPF)

2.5.1 Packet Filtering Rules - per TD0683 (VPNGW12:FPF_RUL_EXT.1)

2.5.1.1 VPNGW12:FPF_RUL_EXT.1.1

TSS Assurance Activities: 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.

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.

Section 6.5 of [ST] indicates that while the TSF is powering up, the TSF does not enable any network interfaces prior to completion of the power-up self-tests. This ensures that the TSF is operating properly and that the packet filtering rules have been initialized before the TSF processes any network data. Once the network interfaces have been enabled, the TSF verifies that all packets which are destined to the TOE are processed using the packet filtering rulesets.

This section also states that packet filtering rulesets are integral to the correct functioning of the network packet routing functionality; any failure of the packet filtering component will also cause networking to fail. At any time that networking functionality is enabled, the packet filtering rules will be applied.

Guidance Assurance Activities: The operational guidance associated with this requirement is assessed in the subsequent test EAs.

Please refer to the subsequent test assurance activities below.

Testing Assurance Activities: 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 directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization.

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 directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization and is only permitted once initialization is complete.

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test Evaluation Activities.

These tests were performed and described as part of the component testing assurance activities for FFW_RUL_EXT.1.1 where the evaluator determined that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

2.5.1.2 VPNGW12:FPF_RUL_EXT.1.2

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

See FPF_RUL_EXT.1.4

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.3 VPNGW12:FPF_RUL_EXT.1.3

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

See FPF_RUL_EXT.1.4

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.4 VPNGW12:FPF_RUL_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes a packet filtering policy that can use the following fields for each identified protocol, and that the RFCs identified for each protocol are supported:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

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).

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

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.

Section 6.5 or [ST] stats that packets entering the TSF's network stack are filtered by a TOE filtering mechanism known as iptables. iptables examines the following fields within the header of each packet: Source Address (IPv4 or IPv6), Destination address (IPv4 or IPv6), Protocol (IPv4 or IPv6), Source Port (TCP or UDP), and destination port (TCP or UDP). The IPsec engine then performs its own filtering and processing as necessary to encrypt and decrypt packets. Finally, the resulting packets are processed via iptables once more.

The TSF initially sets iptables to block traffic on the ingress and egress ports. When the VPN service starts, it initializes and performs various self-tests. Once complete, the TOE loads iptables with the active VPN configuration to allow VPN traffic to commence.

This section states that the TOE supports all of the required protocols, IPv4 (RFC 791), IPv6 (RFC 2460), TCP (RFC 793), AND UDP (RFC 768) as well as source and destination address as determined by testing with known good implementations.

This section states that the TOE implements three different rule chains that can be applied to network traffic on the VPN Ingress, VPN Egress, or Management. Each chain is applied to a different traffic type; traffic addressed to the TOE (INPUT), traffic sent by the TOE (OUTPUT), and traffic passing through the TOE (FORWARD). The rules are applied in the order they appear. Each rule can be ACCEPT, DROP, or LOG. Traffic can be filtered by interface (based on the name of the interface), IP protocol (TCP, UDP), port range, and IP address range. FORWARD rules are applied to all traffic not addressed to the TOE, including decrypted VPN traffic.

This section states that the TOE supports Ethernet interfaces using the same packet filtering policy mechanism. Administrators can configure rules and apply them to individual interfaces, but all use the same underlying mechanism.

Guidance Assurance Activities: The evaluators shall verify that the operational guidance identifies the following protocols as being supported and the following attributes as being configurable within packet filtering rules for the associated protocols:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

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

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

The guidance may 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 shall ensure that it is made clear what protocols were not considered as part of the TOE evaluation.

The section entitled "Access Policy Management" in [CC-Guide] and the subsections within this section provide a description of the packet filtering function supported by the TOE. These sections describe the components of an access-policy as a 'ruleset', 'access-list' and 'match rules'. An access-list can be assigned to an interface to specify the packet filtering policy that is to be enforced on that interface.

Access lists are the collections of rule sets and match rules that can be attached to an interface. Rule sets are collections of match rules. Match rules are individual rules that are applied to each packet to determine what action should be applied to the packet.

Actions consist of things to be done when a match is made, such as to allow the packet to traverse the server (permit), silently drop the packet (deny), drop the packet, and send a notification to the sender (reject), log the packet was seen (log), send a copy of the packet to a defined destination (monitor), or stop processing (return).

The section entitled "Match Rules" provides the format for a match command. This command's format indicates that the match can be based on source address, source port, destination address, destination port, and protocol.

The subsection entitled "Match Rules" contains a list of protocols which the TOE is able to match in an access rule and this list does include IPv4, IPv6, TCP and UDP.

The section entitled "Access Policy Management" indicates that rule sets and access lists are assigned to network interfaces. The command to associate an access-list with an interface is provided in the sections entitled "Configure the Management Network Interface" and "Configure the Management Network Interface".

Testing Assurance Activities: The evaluator shall perform the following tests:

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

- IPv4

o Source address

o Destination Address

o Protocol

- IPv6

o Source Address

o Destination Address

o Next Header (Protocol)

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

Test 2: The evaluator shall repeat Test 1 above for each distinct network interface type supported by the TOE to ensure that Packet filtering rules can be defined for each all supported types.

Note that these test activities should be performed in conjunction with those of FPF_RUL_EXT.1.6 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.1.6 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.

Test 1 & 2: The evaluator configured firewall rules for testing of the other VPNGW12:FPF_RUL_EXT.1 (including VPNGW12:FPF_RUL_EXT.1.6) tests using instructions provided within the administrative guidance and found all necessary instructions were provided accurately. The tests performed for FPF_RUL_EXT.1.6 incorporate numerous variations of packet filtering rules that demonstrate proper enforcement of the packet filtering ruleset (e.g., permit, deny, and log rules, ICMPv*, IPv4 and IPv6, TCP and UDP, numerous ports, source & destination differences, and transport protocols).

2.5.1.5 VPNGW12:FPF_RUL_EXT.1.5

TSS Assurance Activities: 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.

Section 6.5 of [ST] indicates that the TOE implements three different rule chains that can be applied to network traffic on the VPN Ingress, VPN Egress, or Management. Each chain is applied to a different traffic type; traffic addressed to the TOE (INPUT), traffic sent by the TOE (OUTPUT), and traffic passing through the TOE (FORWARD). The rules are applied in the order they appear. Each rule can be ACCEPT, DROP, or LOG. Traffic can be filtered by interface (based on the name of the interface), IP protocol (TCP, UDP), port range, and IP address range. FORWARD rules are applied to all traffic not addressed to the TOE, including decrypted VPN traffic.

This section also states that the TOE initially sets iptables to block traffic on the ingress and egress ports.

Guidance Assurance Activities: 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.

The section entitled "Access Policy Management" in [CC-Guide] describe the packet filtering functionality offered by the TOE. This includes a description of rules that are contained by an access-list. The subsection entitled "Match Rules" in [CC-Guide] explains that match rules are ordered by an index value, which may be updated to rearrange the match rules within a rule set or ACL. There is a limit of 255 match rules per rule set or ACL, and the index is limited to a value between 1 and 999999 inclusive.

This subsection also explains that the rules are evaluated in index order. If no index is specified, the first rule created is set to an index of 10. The index is then incremented by 10 for each additional rule created without a specified index.

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall devise two equal Packet Filtering rules with alternate operations -- permit and discard. 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.

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.

Test 1: The evaluator configured the TOE (according to the admin guide) with two packet filtering rules using the same matching criteria, where one rule would permit while the other would deny traffic. Packets matching the ACL entry rule were sent through the TOE and the evaluator observed that the action taken by the TOE matched the action specified by the first ACL entry.

Test 2: Continuing test 1, the evaluator repeated the procedure above, except the evaluator changed the rules to make one a subset of the other, and then tested both orders. The evaluator confirmed that the first rule is enforced regardless of the specificity of the rule.

2.5.1.6 VPNGW12:FPF_RUL_EXT.1.6

TSS Assurance Activities: 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 discard packets when there is no rule match. The evaluator shall verify the TSS describes when the IPv4/IPv6 protocols supported by the TOE differ from the full list provided in the RFC Values for IPv4 and IPv6 table.

Section 6.5 of [ST] states that the TOE supports all of the required protocols, IPv4 (RFC 791), IPv6 (RFC 2460), TCP (RFC 793), and UDP (RFC 768). It also states that the TOE does not impose limitations upon the protocols supported by IPv4 and IPv6.

Guidance Assurance Activities: 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 discard packets with no matching rules. The evaluator shall verify that the operational guidance describes the range of IPv4 and IPv6 protocols supported by the TOE.

The section entitled "Access Policy Management" in [CC-Guide] states that the system starts with a default policy that but drops all packets (inbound, outbound, and forwarded) packets aside from providing SSH access. The subsection entitled "Match Rules" contains a list of protocols which the TOE is able to match in an access rule and this list does include ipv4 and ipv6 (among many other protocols).

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 supported IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 2: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 3: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 supported IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 4: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 5: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 6: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) 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 the supported protocols are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

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.

Test 8: The evaluator shall configure the TOE to discard 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.

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.

Test 10: The evaluator shall configure the TOE to discard 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.

The following table identifies the RFC defined values for the protocol fields for IPv4 and IPv6 to be used in configuring and otherwise testing Packet Filtering rule definition and enforcement:

Test 1: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured. The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv4 Protocol permitted and logged based on specific source and destination addresses.

- IPv4 Protocol permitted and logged based on specific destination addresses.
- IPv4 Protocol permitted and logged based on specific source addresses.
- IPv4 Protocol permitted and logged based on wildcard addresses.

Test 2: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv4 Protocol all permitted with some denied and logged based on specific source and destination addresses.
- IPv4 Protocol all permitted with some denied and logged based on specific destination addresses.
- IPv4 Protocol all permitted with some denied and logged based on specific source addresses.
- IPv4 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 3: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

- IPv4 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on specific destination addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on specific source addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on wildcard addresses.
- IPv4 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 4: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv6 Protocol permitted and logged based on specific source and destination addresses.
- IPv6 Protocol permitted and logged based on specific destination addresses.
- IPv6 Protocol permitted and logged based on specific source addresses.
- IPv6 Protocol permitted and logged based on wildcard addresses.

Test 5: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv6 Protocol all permitted with some denied and logged based on specific source and destination addresses.

- IPv6 Protocol all permitted with some denied and logged based on specific destination addresses.
- IPv6 Protocol all permitted with some denied and logged based on specific source addresses.
- IPv6 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 6: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- IPv6 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on specific destination addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on specific source addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on wildcard addresses.
- IPv6 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 7: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- TCP (IPv4) permitted and logged based on source port.
- TCP (IPv4) permitted and logged based on destination port.
- TCP (IPv4) permitted and logged based on source and destination port.
- TCP (IPv6) permitted and logged based on source port.
- TCP (IPv6) permitted and logged based on destination port.
- TCP (IPv6) permitted and logged based on source and destination port.

Test 8: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- TCP (IPv4) denied and logged based on source port.
- TCP (IPv4) denied and logged based on destination port.
- TCP (IPv4) denied and logged based on source and destination port.
- TCP (IPv6) denied and logged based on source port.
- TCP (IPv6) denied and logged based on destination port.
- TCP (IPv6) denied and logged based on source and destination port.

Test 9: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured

rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- UDP (IPv4) permitted and logged based on source port.
- UDP (IPv4) permitted and logged based on destination port.
- UDP (IPv4) permitted and logged based on source and destination port.
- UDP (IPv6) permitted and logged based on source port.
- UDP (IPv6) permitted and logged based on destination port.
- UDP (IPv6) permitted and logged based on source and destination port.

Test 10: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied. Additionally, supported protocols are filtered properly and the rules are enforced.

- UDP (IPv4) denied and logged based on source port.
- UDP (IPv4) denied and logged based on destination port.
- UDP (IPv4) denied and logged based on source and destination port.
- UDP (IPv6) denied and logged based on source port.
- UDP (IPv6) denied and logged based on destination port.
- UDP (IPv6) denied and logged based on source and destination port.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6 Protection of the TSF (FPT)

2.6.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1)

2.6.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.6 of [ST] states that administrator passwords are stored salted and hashed 5000 times with SHA-512. The CLI does not provide the user with any commands that allow for the reading of the hashed passwords.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.2 Failure with Preservation of Secure State (Self-Test Failures) (VPNGW12:FPT_FLS.1/SelfTest)

2.6.2.1 VPNGW12:FPT_FLS.1.1/SelfTest

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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 non- security relevant), the evaluator shall ensure that those cases are identified and a rationale is provided that supports the classification and justifies why the TOE's ability to enforce its security policies is not affected in any such instance.

Section 6.6 of [ST] states that if any self-testing generates a failure, the TOE immediately fails-secure by powering off and shutting down.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides information on the self-test failures that can cause the TOE to shut down and how to diagnose the specific failure that has occurred, including possible remediation steps if available.

The section entitled "Self-Test" indicate that if the self-test fails, the system enters fail secure. The following section entitled "User Actions" describes a variety of possible fail-secure conditions and the implications of each. Some fail-secure conditions can be recovered through a reboot, while others are more serious and may prevent operation.

Component Testing Assurance Activities: None Defined

2.6.3 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1)

2.6.3.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.6 of [ST] explains that keys stored in the filesystem are stored in an encrypted partition. The key to unlock this partition is stored in a Root only accessible partition and is automatically read by the system during the boot up process.

This section also states that the CLI does not provide the user with any commands that allow for the reading of the secret and private keys.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.4 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1)

2.6.4.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.4.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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

Section 6.5 of [ST] indicates that the TOE includes its own hardware clock. The TOE allows the administrator to set time manually. The date and time are used as the time stamp that is applied to TOE generated audit records, used to track inactivity of administrative sessions, and perform certificate expiration checks.

The TOE does not support NTP and does not obtain time from an underlying virtualization system.

Component Guidance Assurance Activities: 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.

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

The section entitled "Set System Date/Time" in [CC-Guide] provides instructions for an administrator to manually set the time on the TOE.

The TOE does not support NTP to set time and does not obtain time from an underlying Virtual System.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

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

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

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.

c) Test 3: [conditional] If the TOE obtains time from the underlying VS, the evaluator shall record the time on the TOE, modify the time on the underlying VS, and verify the modified time is reflected by the TOE. If there is a delay

between the setting the time on the VS and when the time is reflected on the TOE, the evaluator shall ensure this delay is consistent with the TSS and Guidance.

Test 1: The evaluator followed the guidance instructions to configure the time on the TOE. The evaluator read the time from the TOE using a date command and also found audit records confirming that the time was successfully changed.

Test 2: The TOE does not support the use of NTP to set time.

Test 3: The TOE does not obtain time from an underlying VS system; thus this test is not applicable.

2.6.5 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.6.5.1 NDCPP22E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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.

Section 6.6 of [ST] indicates that when the TSF starts-up, it runs the following self-tests:

- FIPS Self-Test for the Kernel, OpenSSL, and libgcrypt Cryptographic Modules
  - HMAC Integrity Check
  - Known Answer Tests of cryptographic algorithms
- Entropy Health Check
  - Verification that RDSEED does not report a failure
- Digital Signature verification of all software components

This section goes on to describe these tests further in subsequent paragraphs. It concludes by justifying that these tests are sufficient using the following argument.

Because all cryptographic operations are tested, the TSF is known to be performing cryptographic operations correctly. Because the underlying hardware is tested, the TSF is known to be correctly executing the firmware. Together, when the TOE is operating, the TSF is known to be operating as expected to enforce the SFRs.

The TOE is not distributed.

Component Guidance Assurance Activities: 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.

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.

The section entitled "Self-Test" indicate that if the self-test fails, the system enters fail secure. The following section entitled "User Actions" describes a variety of possible fail-secure conditions and the implications of each. Some fail-secure conditions can be recovered through a reboot, while others are more serious and may prevent operation.

Component Testing Assurance Activities: 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 fulfill any of the SFRs.

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.

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.

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.

During a reboot of the TOE, the evaluator confirmed that the TOE performed self tests to verify the firmware integrity and the cryptographic functions as identified in the screenshot below. The output of these tests indicate that they were successful. The firmware integrity test passed and all other tests were successfully completed with no errors.

2.6.6 TSF Testing (VPNGW12:FPT_TST_EXT.1)

2.6.6.1 VPNGW12:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module requires a particular self-test to be performed, but this self-test is still evaluated using the same methods specified in the Supporting Document.

See the evaluation assurance activity for NDcPP22e:FPT_TST_EXT.1

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.7 Self-Test with Defined Methods (VPNGW12:FPT_TST_EXT.3)

2.6.7.1 VPNGW12:FPT_TST_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.7.2 VPNGW12:FPT_TST_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator verifies that the TSS describes the method used to perform self-testing on the TSF executable code, and that this method is consistent with what is described in the SFR.

Section 6.6 of [ST] describes the method used to perform self-testing on the TSF executable code with the statements that, "The TSF verifies the integrity of all TSF components by verifying a digital signature (RSA 2048) of the when each software component is loaded or reloaded into memory for execution. The TSF verifies the integrity of configuration files by comparing a hash of each configuration file to a hash generated when the configuration file was last updated."

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.8 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.6.8.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.3 NDcPP22e:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

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.

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.

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.

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.

The TOE does not support delayed activation during an update, does not support automatic checking for updates and is not distributed.

Section 6.6 of [ST] states that the “show system version” command allows the user to query the current overall software version of the TOE.

Section 6.6 also states that when installing updated software, the TOE uses digital signatures to authenticate the update, as well as to ensure it is the update intended and originated by the vendor.

This section also indicates that the TSF utilizes the Red Hat RPM package management system for software updates. The TSF is configured to trust updates that are digitally signed by Red Hat’s private key and Apriva’s private key. The TSF disallows the user from performing an update using a solely Red Hat signed RPM. The update package must be signed by Apriva, but sub-packages can be signed by Red Hat only. This ensures that users do not load arbitrary Red Hat RPMs on the TSF. Both the Red Hat and Apriva private keys are RSA 2048-bit keys. Red Hat-supplied RPM files are digitally signed with Red Hat’s private key. Apriva-created packages are only signed with an Apriva private key. The TSF automatically verifies the signature of any package that is updated. If the signature verification fails, the TSF logs the failure, aborts the update, and deletes the invalid package. If the signature verification succeeds, the TSF installs the update. All updates are “atomic” once the update process has started, so the TOE automatically restarts and activates the update if it was successful, or the TOE rolls back to the previous version if the update failed for any reason.

Component Guidance Assurance Activities: 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.

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.

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.

For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT_TUD_EXT.1, how all TOE components are updated, and the error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. The guidance documentation only has

to describe the procedures relevant for the Security Administrator; it does not need to give information about the internal communication that takes place when applying updates.

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.

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.

The section entitled "Software Identification" explains that the CLI command "show system version" identifies the running system version.

The section entitled "Verifying Digital Signature" states that all software updates are signed using a digital signature. The process for validating signatures is automated by the system and is automatically performed as part of a software update. When the system update command is executed, the system automatically validates the signatures of all the files in the update prior to beginning the installation of the updates and will only perform the update if all signatures are successfully updated needs to be change to validated.

The TOE does not use published hash or certificates to authenticate an update. Also, the TOE is distributed.

Component Testing Assurance Activities: 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.

b) Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE.

The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

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

2) An image that has not been signed

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

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

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

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

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

to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE.

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

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.

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).

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

Test 1: Prior to performing an update, the evaluator verified the TOE version using TOE commands. The evaluator then followed guidance to install a valid update to the TOE. Upon successful installation, the evaluator verified the TOE version once again and confirmed that the version after the successful update was changed as expected.

Test 2: The evaluator attempted to perform a TOE update using a legitimate update where the executable code was modified and the update contained the original signature. The TOE rejected the update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with the digital signature removed. The TOE rejected the unsigned update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with the digital signature modified. The TOE rejected the update and the product version did not change.

Test 3: This was not applicable as the TOE does not claim Published Hashes are used for product updates.

2.6.9 Trusted Update (VPNGW12:FPT_TUD_EXT.1)

2.6.9.1 VPNGW12:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.9.2 VPNGW12:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.9.3 VPNGW12:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to mandate that a particular selection be chosen, but this selection is part of the original definition of the SFR so no new behavior is defined by the PP-Module.

See the evaluation assurance activity for NDcPP22e:FPT_TUD_EXT.1.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7 TOE ACCESS (FTA)

2.7.1 TSF-INITIATED TERMINATION (NDCPP22E:FTA_SSL.3)

2.7.1.1 NDCPP22E:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 of [ST] explains that the TOE terminates remote sessions that have been inactive for an administrator-configured period of time. After termination, administrative authentication is required to access any of the administrative functionality of the TOE.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

The section entitled "Configure the SSH Service" in [CC-Guide] includes an example of how to set the inactivity timeout value for remote administrative sessions. This section explains that the timeout value defines the inactivity limit (in seconds). Once the SSH session has been inactive for this time period, the user is logged out of the sessions automatically.

Component Testing Assurance Activities: 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.

The evaluator followed the guidance to configure the session timeout periods for SSH CLI remote sessions. The evaluator confirmed that the session was terminated after the configured time period. The inactivity time period was configured for periods of 1 minute, 3 minutes and 5 minutes.

2.7.2 User-initiated Termination (NDcPP22e:FTA_SSL.4)

2.7.2.1 NDcPP22e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 of [ST] explains that the TOE provides the function to logout (or terminate) both local and remote user sessions as directed by the user.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

The section entitled "Re-login as the Superuser User" in [CC-Guide] explains that a user with a CLI session (either using the local console or using a remote SSH connection) can terminate their interactive session using the “logout” command.

Component Testing Assurance Activities: 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.

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.

Test 1: The evaluator logged in to the local console and then typed in the command "logout". The evaluator observed that the session ended and a login prompt was presented.

Test 2: The evaluator repeated this test using an SSH connection and observed that the session ended and the SSH connections was terminated.

2.7.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.7.3.1 NDcPP22e:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 of [ST] states that the TOE terminates local sessions that have been inactive for an administrator-configured period of time. After termination, administrative authentication is required to access any of the administrative functionality of the TOE.

Component Guidance Assurance Activities: 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.

The section entitled "CLI Session Timeout" in [CC-Guide] indicates that the CLI session timeout on the local console can be configured by an administrator using the "**`console-timeout`**" command.

Component Testing Assurance Activities: 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 reauthentication is needed when trying to unlock the session.

The evaluator followed the guidance to configure the idle timeout periods for the Local Console session and confirmed that the session was terminated after the configured time period. The inactivity time period was configured using the "console timeout" command for periods of 1 minute, 3 minutes and 5 minutes.

2.7.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.7.4.1 NDcPP22e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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).

Section 6.7 of [ST] states that the TOE displays a configurable advisory and consent message when administrator accesses the CLI through either the local or remote administrative interface. The TOE can be configured to display administrator-defined advisory banners when administrators establish interactive sessions with the TOE, allowing administrators to terminate their session prior to performing any functions. The banner is displayed after the login ID has been presented, but before the password prompt is shown.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

The section entitled "Set the Login Banner" in [CC-Guide] provides the commands that an administrator can use to configure the message that users see before they log in. The banner command is identified as this command. This section also explains that the banner is displayed on the local console and remote SSH logins.

Component Testing Assurance Activities: 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.

The evaluator configured a banner and verified that the banner was displayed appropriately for console and SSH CLI logins.

2.7.5 TOE Session Establishment - per TD0656 (VPNGW12:FTA_TSE.1)

2.7.5.1 VPNGW12:FTA_TSE.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the methods by which the TSF can deny the establishment of an otherwise valid remote VPN client session (e.g., client credential is valid, not expired, not revoked, etc.), including day, time, and IP address at a minimum.

Section 6.7 of [ST] explains that when acting as a VPN Headend and authenticating VPN users, the TSF can be configured to prevent access based on remote IP address, time of day, and/or day of week.

Component Guidance Assurance Activities: The evaluator shall review the operational guidance to determine that it provides instructions for how to enable an access restriction that will deny VPN client session establishment for each attribute described in the TSS.

The section entitled "Configure Global VPN Settings" in [CC-Guide] explains that vpn-settings can include restrictions on location, day and time. This section contains an example that uses the "authrule" command to restrict VPN peers to a specific IP subnet (location), to specific days of the week, and to a range of time.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 1: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it, noting the IP address from which the client connected. The evaluator shall follow the steps described in the operational guidance to prohibit that IP address from connecting, attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 2: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it. The evaluator shall follow the steps described in the operational guidance to prohibit the VPN client from connecting on a certain day (whether this is a day of the week or specific calendar date), attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 3: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it. The evaluator shall follow the steps described in the operational guidance to prohibit the VPN client during a range of times that includes the time period during which the test occurs, attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 4: [conditional] If any other attributes are identified in FTA_TSE.1, the evaluator shall conduct a test similar to tests 1 through 3 to demonstrate the enforcement of each of these attributes. The evaluator shall demonstrate a successful remote client VPN connection, configure the TSF to deny that connection based on the attribute, and demonstrate that a subsequent connection attempt is unsuccessful.

Test 1: The evaluator connected a remote VPN client to the TOE and recorded the IP address from which the client connected, then disconnected the client. The evaluator configured the TOE to prevent that IP address from connecting, then attempted to reconnect using the same VPN client, and observed that it was not successful.

Test 2: The evaluator connected a remote VPN client to the TOE and recorded the IP address from which the client connected, then disconnected the client. The evaluator configured the TOE to prevent that IP address from connecting on a certain day. The evaluator then attempted to reconnect using the same VPN client on the configured day, and observed that it was not successful.

Test 3: The evaluator connected a remote VPN client to the TOE and recorded the IP address from which the client connected, then disconnected the client. The evaluator configured the TOE to prevent that IP address from connecting during a range of times that includes the current time. The evaluator then attempted to reconnect using the same VPN client, and observed that it is not successful.

Test 4: Not applicable. No other attributes are claimed by the TOE.

2.7.6 VPN Client Management - per TD0656 (VPNGW12:FTA_VCM_EXT.1)

2.7.6.1 VPNGW12:FTA_VCM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to verify that it asserts the ability of the TSF to assign a private IP address to a connected VPN client.

Section 6.7 of [ST] explains that the TOE supports the capability of assigning a private IP address to VPN clients upon successful establishment of a session.

Component Guidance Assurance Activities: There are no operational guidance EAs for this component.

There are no operational guidance EAs for this component.

Component Testing Assurance Activities: The evaluator shall connect a remote VPN client to the TOE and record its IP address as well as the internal IP address of the TOE. The evaluator shall verify that the two IP addresses belong to the same network. The evaluator shall disconnect the remote VPN client and verify that the IP address of its underlying platform is no longer part of the private network identified in the previous step.

The evaluator connected a GSS test server to the VPN and observed that the TOE successfully assigned an IP address to the client. The evaluator than disconnected the VPN client and observed that the platform and the peer were no longer on that same network.

2.8 Trusted path/channels (FTP)

2.8.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1)

2.8.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.3 NDcPP22e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.8 of [ST] states that the TOE uses TLS when exporting audit records to a third-party syslog server. The TOE uses IPsec when communicating with IPsec clients and peers.

Section 6.8 also states that when making connections with remote VPN clients and peers the TOE can act as the initiator or the responder for initiates the IPsec communication between the IPsec peers.

Section 6.8 also states that the TOE provides assured identification of non-TSF endpoints (for both TLS and IPsec) by validating X.509 certificates. The TOE implements a trust store containing trust anchors which it uses to verify identities of those non-TSF certificates. The TOE utilizes TLS and IPsec as described in Section 6.2 of [ST].

Component Guidance Assurance Activities: 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.

Section 1, "Introduction" of [CC-Guide] indicates that communication with an external audit server that can be protected by TLS and communication with VPN clients that are protected by IPsec.

For the TLS communication with an external audit server (syslog server), there are several sections with relevant information.

The section entitled "Syslog Server Requirements", describes a variety of constraints which the syslog server must satisfy in order for the TOE to securely communicate with the syslog server.

The section entitled "Import TLS Trust Chains and Certificates/Keys" provides instructions to load certificates used by the TOE for the TLS connection to the syslog server. A subsection to this includes an example of the commands needed to configure the TOE to communicate with the syslog server using TLS. This includes command that identify certificate chains, provide syslog server reference identifiers, enable the required revocation checking, specify the TOE TLS identity certificate, and specify the syslog server's network address/port. This section also states that if a connection is interrupted, it will automatically attempt to reconnect.

The section entitled "Relationship between local logs and remote audit server logs", indicates that the TOE does not implement queueing of audit data that is generated when a connection to an external audit server is not available.

For the IPsec communication with VPN peer, there are also several sections with relevant information.

The administrator must configure VPN Authentication suites as described by the section entitled "Create VPN Authentication Suites". These suites specify the IPsec security parameters used for VPN communication. This includes encryption algorithms, MAC algorithms, Diffie-Hellman groups, and SA lifetimes.

The section entitled "Import VPN Trust Chains and Certificates/Keys" provides instructions to load certificates used by the TOE. The section entitled "Configure Global VPN settings" allows an administrator to apply constraints on VPN connections for destination, day, and time. The VPN settings can also specify the Authentication Suites that may be used. The VPN settings along with an access-list (Packet Filtering Rules) provide all the functionality of the SPD. An access-list defines the packets that are permitted or dropped. For packets that are permitted, those that match the vpn-settings traffic selector are encrypted, otherwise they bypass encryption.

The section entitled "Add Devices to the Apriva MESA VPN Server" describes the use of a whitelist to specify the authorized devices that may be an IPsec peer. This section indicates that the Apriva MESA VPN server will automatically reconnect an IPsec VPN connection that is interrupted.

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

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.

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.

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

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

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

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

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

Further assurance activities are associated with the specific protocols.

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.

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

The TOE utilizes TLS to protect communications with an external audit server (syslog server) and IPsec for communication with VPN peers.

The evaluator established a successful TLS connection between the TOE and the external audit server (the TOE initiated the connection [Test 2]) and began a packet capture. Within the new TLS session [Test 1], the evaluator observed encrypted application data traffic between the TOE and the test server [Test 3]. Next the evaluator initiated the physical disruption [Test 4] between the TOE and the remote audit server for roughly 90 seconds. Upon reconnection the TOE continued to use the original TLS sessions and did not establish a new TLS tunnel between the TOE and the test server. The evaluator also observed audit records appearing on the syslog server

after the connection was repaired. The packet capture shows that upon reconnection the TOE does not initiate a new TLS handshake.

The evaluator repeated the above steps using a duration of 50 minutes. Upon reconnection, the evaluator observed that the TOE initiated a new TLS negotiation to establish a new TLS session between the TOE and the Gossamer test server. The evaluator also observed audit records appearing on the syslog server. The packet capture shows that upon reconnection the TOE does initiate new TLS handshake.

The evaluator established a successful TOE IPsec connection supporting communication to a VPN peer [Test 1] and began a packet capture. The packet capture shows that the TOE initiated the connection [Test 2]; and Application data transferred is encrypted (i.e., not plaintext) [Test 3]. Next the evaluator initiated the physical network disruption [Test 4] between the TOE and the VPN peer for roughly 30 seconds. Upon reconnection, the evaluator observed audit records appearing on the syslog server. The packet capture shows that upon reconnection the TOE does not initiate ISAKMP negotiation followed by ESP traffic. The evaluator observed that no data was transmitted unprotected.

The evaluator repeated the above steps using a duration of 45 minutes and observed that the TOE initiated a new IKE negotiation to establish a new IPsec tunnel between the TOE and the Gossamer test server. The evaluator also observed audit records appearing on the syslog server. The packet capture shows that upon reconnection the TOE does initiate new ISAKMP negotiation followed by ESP traffic.

2.8.2 Inter-TSF Trusted Channel (VPN Communications) (VPNGW12:FTP_ITC.1/VPN)

2.8.2.1 VPNGW12:FTP_ITC.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

See the evaluation assurance activity for NDcPP22e:FTP_ITC.1.

Component Guidance Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

Refer to NDcPP22e:FTP_ITC.1 where these activities have been performed and applied to IPsec VPN communications.

Component Testing Assurance Activities: TheEAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications. Additional evaluation testing for IPsec is covered in FCS_IPSEC_EXT.1.

Refer to NDcPP22e:FTP_ITC.1 where these activities have been performed and applied to IPsec VPN communications.

2.8.3 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin)

2.8.3.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 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.

Section 6.8 of [ST] states that the TOE provides a trusted path for its remote administrative users accessing the TOE using SSH protected Command Line Interface. The protocols identified in section 6.8 are consistent with those listed in the requirement and the SSH protocol is described in Section 6.2 of [ST].

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Section 1 of [CC-Guide] indicates that the TOE provides administration at the local console, remote administration via SSH.

The section entitled "Initial Login" in [CC-Guide] describes the process to set up and use the system console for management of the TOE. While the section entitled "Configure the Management Network Interface" describes how to configure the TOE to offer an SSH-protected Command Line interface to remote administrators.

Component Testing Assurance Activities: 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.

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

Further assurance activities are associated with the specific protocols.

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.

The TOE offers remote administration via SSHv2 to provide the trusted path (with protection from disclosure and modification) for all remote administration sessions.

Test 1: The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE using SSH and observed that the connection was successful and did use SSHv2.

Test 2: The evaluator examined the packet capture and observed that the remote administration session was protected by SSH and did not contain plaintext data.

3. Protection Profile SAR Assurance Activities

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

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.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

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

The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

It should be noted that there may be some SFRs that do not have an interface that is explicitly 'mapped' to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

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. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps (per TD0536):

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

In addition, the evaluator performs the EAs specified below.

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.

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.

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.

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.

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.

As identified throughout this AAR, the [CC-Guide] provides instructions for configuring the TOE's cryptographic security functions. The [CC-Guide] provides instructions for configuring the cryptographic algorithms and parameters used for the evaluated configuration. The [CC-Guide] is clear that no other cryptographic configuration has been evaluated or tested. There are warnings and notes throughout the [CC-Guide] regarding use of functions that are and are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP22e:FPT_TUD_EXT.1.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

Preparative procedures are 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.

In addition, the evaluator performs the EAs specified below.

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).

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

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.

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

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.

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.

The evaluation team had the following documents to use when configuring the TOE:

- Apriva Mesa VPN Common Criteria Configuration Guide [CC- Guide]

The completeness of this documentation is addressed by its use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a 'hard label' (e.g., stamped into the metal, paper label) or a 'soft label' (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

See section 3.3.1 above for an explanation of how all CM items are addressed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes 'evaluated configuration' instructions). The focus of the testing is to confirm that the requirements specified in Section 5.1.7 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

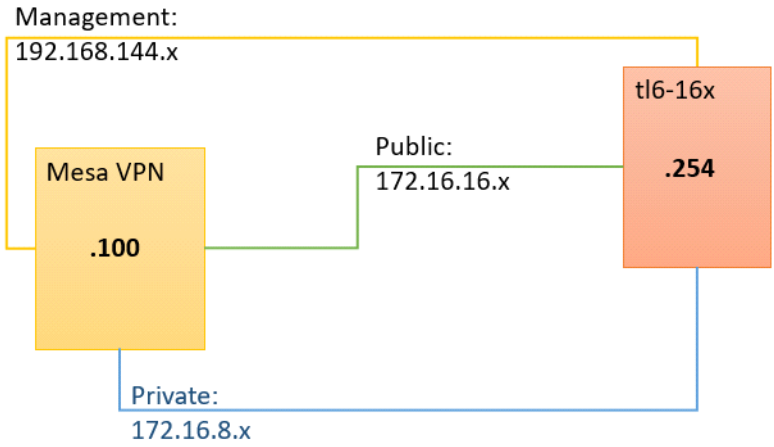
The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

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

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

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section B.4.3.1.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The test configuration consisted of the following TOE platforms along with supporting products.



TOE Platforms:

- Apriva MESA VPN v3.0 on Dell PowerEdge R750 w/ Intel Xeon Silver

Supporting Products:

- None

Supporting Software:

The Gossamer Test servers utilized both a windows and Ubuntu environment. The Windows supporting software included the following.

- Wireshark version 4.0.5
- Windows SSH Client - Putty version 0.73, 0.74 & 0.78 (used to connect to device console and SSH)

The Gossamer Test servers with an Ubuntu environment acted as platforms to initiate testing. The test servers also acted as a syslog server.

- OpenSSL version 1.0.2g
- OpenSSH version 7.2p2
- Big Packet Putty, Openssh-client version 6.8p1,
- Rsyslog version 8.16.0
- TCPdump version 4.9.3
- LibPCAP version 1.7.4
- NMAP version 7.01 (Linux)
- Stunnel version 5.30

3.5 Vulnerability Assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A, while an 'outline' of the assurance activity is provided below.

In addition to the activities specified by the CEM in accordance with Table 2, the evaluator shall perform the following activities.

The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

The developer shall provide documentation identifying the list of software and hardware components7 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.

If the TOE is a distributed TOE then the developer shall provide:

a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]

b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

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.

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuzz testing. None of the public search for vulnerabilities or the fuzz testing uncovered any residual vulnerability.

The evaluation team performed a public search for vulnerabilities in order to ensure there are no publicly known and exploitable vulnerabilities in the TOE from the following sources:

- National Vulnerability Database (https://web.nvd.nist.gov/vuln/search)
- Vulnerability Notes Database (http://www.kb.cert.org/vuls/)
- Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities)
- Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories)
- Exploit / Vulnerability Search Engine (http://www.exploitsearch.net)
- SecurITeam Exploit Search (http://www.securiteam.com)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search)
- Offensive Security Exploit Database (https://www.exploit-db.com/)

The search was performed on July 12, 2023. The search was conducted with the following search terms: "Apriva", "MESA", "Broadcom 5720", "Intel X710-T4L", "ssh", "tls", "ipsec", “Intel+Xeon+Silver”, “Intel+Xeon+Gold” and “Ice+Lake”.