www.GossamerSec.com

Assurance Activity Report for Aruba, a Hewlett Packard Enterprise Company 6300M and 8360v2 Switch Series MACsec

Version 0.3
April 13, 2024

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

Document: AAR-VID11422

Revision History

Revision	Date	Authors	Summary
Version 0.1	February 29, 2024	Gossamer Security Solutions	Initial draft
0.2	April 4, 2024	Gossamer	Resolve NIAP comments

The TOE Evaluation was Sponsored by:

Aruba, a Hewlett Packard Enterprise Company
8000 Foothills Blvd.
Roseville, CA 95747

Evaluation Personnel:

- Dip Pudasaini
- Cornelius Haley
- Will Micknick

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 Aruba, a Hewlett Packard Enterprise Company 6300M and 8360v2 Switch Series MACsec 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 evaluation includes two TOE models. Both models were fully tested during the evaluation. Therefore, no equivalence claims are made for this evaluation.

1.1.2 CAVP Certificate Justification

The TOE is the Aruba, a Hewlett Packard Enterprise Company 6300M and 8360v2 Switch Series MACsec which includes the following cryptographic libraries performing the cryptographic operations identified.

- Aruba AOS-CX Cryptographic Module version 1.0
  - TLS connections, SSH connections, Key generation and establishment,
  - Random number generator
  - Trusted updates, Product integrity
  - ECDSA Key Generation & Verification
  - AES CMAC and AES Key Wrap supporting MACsec
- Aruba AOS-CX RSA Engine
  - RSA Key Generation
- AES ECB 128bit & 256bit Encryption/Decryption Engine
  - MACsec AES GCM cryptography

These cryptographic libraries perform all cryptographic operations that provide the cryptographic functions identified in the following table.

Functions	Requirement	Standard	Certificate #
Encryption/Decryption			
AES CBC (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A ISO 19772	A4592
AES-CTR (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A ISO 19772	A4592

AES GCM (128 and 256 bits)	FCS_COP.1/DataEncryption	ISO 19772 FIPS Pub 197 NIST SP 800-38A	A4592	
Cryptographic hashing				
SHA-1, SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	A4592	
Keyed-hash message authentication				
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (digest sizes and block sizes of 160, 256, 384 and 512 bits)	FCS_COP.1/KeyedHash	FIPS Pub 198-1 FIPS Pub 180-4 ISO/IEC 9797-2:2011	A4592	
Cryptographic signature services				
RSA Digital Signature (rDSA) (2048, 3072 bits)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 9796-2	A4592	
ECDSA Digital Signature (P-256, P-384, P-521)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 14888-3	A4592	
Random bit generation				
CTR_DRBG(AES) 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	A4592	
Key generation				
RSA Key Generation (2048-bit)	FCS_CKM.1	FIPS Pub 186-4 ISO/IEC 9796-2	A4590	
ECC Key Generation (P-256, P-384, P-521)	FCS_CKM.1	FIPS PUB 186-4	A4592	
FFC Scheme using key sies of 2048-bit or greater DSA KeyPairGen	FCS_CKM.1	FIPS PUB 186-4	A4592	
FFC Scheme using Diffie-Hellman Group 14	FCS_CKM.1	Per Policy 5: No NIST CAVP, CCTL must perform all assurance/evaluation activities		
Key establishment				
RSA	FCS_CKM.2	RSAES-PKCS1-v1_5	Tested with known good implementation	
KAS ECC P-256, P-384, P-521	FCS_CKM.2	NIST SP 800-56A Rev 3	A4591	
KAS FFC	FCS_CKM.2	NIST SP 800-56A Rev 3	A4591	
FFC Schemes using 'safe-prime' groups	FCS_CKM.2	NIST SP 800-56A Rev 3	Tested with known good implementation	
MACsec				
AES-CMAC 128 & 256 bits	FCS_COP.1/KeyedHashCMAC	SP 800-38B	A4592	
AES Key Wrap 128 & 256 bits	FCS_COP.1/MACSEC	SP 800-38F (wrap)	A4592	
AES GCM 128 & 256 bits	FCS_COP.1/MACSEC	ISO 18033-3 (AES) ISO 19722 (GCM)	BCM 82399	AES 4545
			BCM 54998SM	C1869
			BCM 82756	AES 4550
			BCM 82759	AES 4550
			BCM 82399	AES 4545
			BCM 82398	AES 4545

Table 1-1 TOE Cryptographic Algorithms

1.2 References

The following evidence was used to complete the Assurance Activities:

[ST]	Aruba, a Hewlett Packard Enterprise Company 6300M and 8360v2 Switch Series MACsec Security Target
[CC-Guide]	Common Criteria Admin Guide Network Device collaborative Protection Profile, Target of Evaluation: Aruba 6300 and 8360v2 Switch Series, Version 2.5, April 4, 2024[1]

[1] Note this is the same CC Admin Guide used for the evaluation of these (and other) devices during a recent NDcPP22e evaluation.

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 the ST states for the administrative tasks of generating/importing/deleting cryptographic keys associated with MACsec, the TOE logs the entire command that defines the key, including the key and target address.

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.

The section entitled "List of Auditable Events (as Mandated by the NDcPP22e)" in the [CC-Guide] provides an example of each auditable event required by FAU_GEN.1. The section entitled “Appendix A: MACsec Configuration” contains an “Audit Events” table that includes the audits associated with the MACsec protocol as required by MACSEC10. During testing, the evaluator mapped the entries in the tables in these sections to the TOE generated events, showing that these sections provide 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 (MACsec) (MACSEC10:FAU_GEN.1/MACSEC)

2.1.2.1 MACSEC10:FAU_GEN.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 MACSEC10:FAU_GEN.1.2/MACSEC

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: None Defined

Component Testing Assurance Activities: None Defined

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.

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

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.

See FAU_GEN.1.

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

2.1.4.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.3 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 to an external SYSLOG server. This communication is protected with TLS.

Section 6.1 of [ST] also states that the TOE is a standalone device that is able to generate and store audit records of security relevant events as they occur.

Section 6.1 of [ST] explains that the TOE supports storage of local audit records in two types of logs visible through two CLI commands. The accounting log is visible using the command "show accounting log" while the event log is visible using the command "show logging". The TOE is capable of rotating through a set of compressed files for each log type. The TOE will check periodically to determine whether or not to rotate its logs based upon log size. The TOE fills one file (“full” storage is defined as consumption of the total configured storage space within a log file), it will rotate the contents of the current log into a compressed file, while rotating previously compressed files until finally deleting the oldest compressed file. The event log maintains six (6) total log files with one current file and 5 rotated files. The accounting log maintains two log files with one current and one rotated file.

Section 6.1 of [ST] states that TOE audit records are protected against unauthorized access by only allowing authorized administrators to have access to local audit logs. Once configured to export audit records, the TOE attempts to transmit all logs in real-time, will temporarily maintain unsent records in the event of a disrupted syslog connection, and sends those records when the remote audit server successfully reestablishes the connection. The TOE uses the TLS protocol to protect audit records transmitted to the external syslog server.

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 "Secure Remote Logging" in [CC-Guide] explain how to configure and establish the trusted channel to the audit server. It states that all audit events that are generated are logged locally and also sent to all configured syslog servers.

This section indicates that the TOE will attempt to transmit the log to the syslog server at the same time it is generated locally. It also indicates that the syslog server is expected to be compliant with RFC 5425 (TLS Transport Mapping for Syslog).

The section entitled "Audit log rotation" states the possible configuration options for log file size threshold and rotation frequency, and the resulting behavior of these possible configurations.

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 and performed some auditable administrative actions. The evaluator then captured network traffic between the TOE and the external audit server. The evaluator verified that the packet capture showed the audit data was not clear text on the network.

Test 2: The TSS describes the TOE as creating two types of audit data: accounting and event messages. The CLI provides commands to view both types of records. Audit overflow is provided by the TOE through a log file rotation mechanism. When this log file becomes "full", then the oldest file is deleted and new records are written into a new file. The evaluator generated audit data for each type of the log and verified that the log files were filled and rotated. When the last log file was filled and rotated, the evaluator observed that the TOE deleted the oldest audits.

Test 3 & 4 are not applicable.

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 can generate ECDSA SSH host keys of size P-256, P-384 and P-521. It can also generate RSA, DH and ECDHE asymmetric keys as part of TLS key establishment during TLS negotiations.

For asymmetric key pairs used for key exchange, the TOE supports generating ephemeral ECDH keys and DH keys for the SSHv2 key exchange methods selected in FCS_SSHS_EXT.1.7. This implies that the TOE generates ephemeral 256/384-bit ECDH keys using ECC schemes for P-256/384 curves and 2048/3072-bit keys using FFC schemes for DH keys for prime group DH14. The TOE supports DH group 14 key establishment scheme that meets standard RFC 3526, section 3 for interoperability.

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 section entitled "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. Section 6.2 of [ST] identifies the TLS ciphersuites supported by the TOE and indicates they are not configurable.

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.

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.

2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.2.1 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)

Table 6-1 within section 6.2 of [ST] indicates that the TOE supports RSA key generation using 2048-bit keys, and ECC key generation using curves P-256, P-384 and P-521. These can be used to generate keys for use with a Certificate Signing Request, as well as in support of key establishment methods identified by Table 6-2.

Table 6-2 indicates that the TOE supports key establishment using ECDHE, FFC Schemes using 'safe-primes' and FFC schemes, which is consistent with the NDcPP22e:FCS_CKM.2 selections.

Table 6-4 indicates the mapping of these schemes to the SFR and services they support.

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 section entitled "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes key exchange algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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 for this testing.

The FFC Schemes using RSA RSAES-PKCS1-v1_5 was tested against the public implementation of these schemes refer to FTP_TRP.1/Admin for this testing.

2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.3.1 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, Table 6-2, in the ST identifies the Key or CSP, where it is stored, when it is zeroized and how the zeroization is performed.

Section 6.2 of the ST states keys are zeroized when they are no longer needed by the TOE, and additionally, the TOE saves keys to persistent storage. Whether saving or destroying keys, the TOE delays the operation at the physical layer until the administrator issues the “write memory” command, which saves the running configuration to the startup configuration.

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).

Key destruction for SSH host keys and X.509 certificate private keys is performed using the 'erase all zeroize' command which is described in the section entitled "SSH host key generation" and "Certificate Installation and Validation" of [CC-Guide].

Component Testing Assurance Activities: None Defined

2.2.4 Cryptographic Operation (AES-CMAC Keyed Hash Algorithm) (MACSEC10:FCS_COP.1/CMAC)

2.2.4.1 MACSEC10:FCS_COP.1.1/CMAC

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 AES-CMAC function: key length, hash function used, block size, and output MAC length used.

Section 6.2 of [ST] states that the TOE supports keyed-hash message authentication in accordance with AES-CMAC algorithm with key sizes 128 bits and 256 bits, the message digest size (output size) of 128 bits and block size of 128-bits. The algorithm conforms to NIST SP 800-38B. Using this statement the evaluator confirmed the values required were present as follows:

- Hash Function: AES-CMAC
- Key length: 128 bits and 256 bits
- Block size: 128-bits
- Output MAC length: 128-bits

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 1: CMAC Generation Test

To test the generation capability of AES-CMAC, the evaluator shall provide to the TSF, for each key length-message length-CMAC length tuple (in bytes), a set of eight arbitrary key-plaintext tuples that will result in the generation of a known MAC value when encrypted. The evaluator shall then verify that the correct MAC was generated in each case.

Test2: CMAC Verification Test

To test the generation capability of AES-CMAC, the evaluator shall provide to the TSF, for each key length-message length-CMAC length tuple (in bytes), a set of 20 arbitrary key-MAC tuples that will result in the generation of

known messages when verified. The evaluator shall then verify that the correct message was generated in each case.

The following information should be used by the evaluator to determine the key length-message length-CMAC length tuples that should be tested:

- Key length: values will include the following:

o 16

o 32

- Message length: values will include the following:

o 0 (optional)

o Largest value supported by the implementation (no greater than 65536)

o Two values divisible by 16

o Two values not divisible by 16

- CMAC length

o Smallest value supported by the implementation (no less than 1)

o 16

o Any supported CMAC length between the minimum and maximum values

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

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 the ST states the TOE supports the CBC and CTR modes of AES as available ciphers for SSH and GCM mode for TLS ciphersuites (all with both 128 and 256-bit keys).

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 "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH cipher algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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 (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash)

2.2.6.1 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 the ST states the TOE uses the SHA-1, 256, 384, and 512 hashing algorithms as part of SSHv2 integrity algorithms (see FCS_SSHS_EXT.1.6) and TLS ciphersuites. The TOE also uses SHA-256 during verification of a new image (trusted updates).

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 "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH HMAC algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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

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

2.2.7.1 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.

Table 6 in Section 6.2 of the ST states the TOE uses HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA512 for SSHv2 and TLS and provides the hash algorithm, key size, block size, and output MAC length used for each.

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 "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH HMAC algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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

2.2.8 Cryptographic Operation (MACsec AES Data Encryption and Decryption) - per TD0728 (MACSEC10:FCS_COP.1/MACSEC)

2.2.8.1 MACSEC10:FCS_COP.1.1/MACSEC

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 describes the supported AES modes that are required for this PP-Module in addition to the ones already required by the NDcPP in FCS_COP.1/DataEncryption.

Section 6.2 of [ST] states that the TOE performs AES key wrap with AES-GCM. AES is specified in ISO 18033-3, AES Key Wrap is specified in NIST SP 800-38F, GCM is specified in ISO 19772.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform testing for AES-GCM as required by the NDcPP in FCS_COP.1/DataEncryption.

In addition to the tests specified in the NDcPP for other iterations of FCS_COP.1, the evaluator shall perform the following tests:

Test 3: KW-AE Test: To test the authenticated encryption capability of AES key warp (KW), the evaluator shall provide five sets of 100 messages and keys to the TOE for each key length supported by the TSF. Each set of messages and keys shall correspond to one of five plaintext message lengths (detailed below). The evaluator shall have the TSF encrypt the messages with the associated key. The evaluator shall verify that the correct ciphertext was generated in each case.

Test 4: KW-AD Test: To test the authenticated decryption capability of AES KW, the evaluator shall provide five sets of 100 ciphertext values and keys to the TOE for each key length supported by the TSF. Each set of ciphertexts and keys shall correspond to one of five plaintext message lengths (detailed below). For each set of 100 cyphertext values, 20 shall not be authentic (i.e. fail authentication). The evaluator shall have the TSF decrypt the ciphertext messages with the associated key. The evaluator will then verify the correct plaintext was generated or the failure to authenticate was correctly detected.

The messages in each set for both tests shall be the following lengths:

- two lengths that are non-zero multiples of 128 bits (two semiblock lengths)

- two that are odd multiples of the semiblock length (64 bits)

- the largest supported plaintext length less than or equal to 4096 bits

(TD0728 applied)

The TOE has been CAVP 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 the ST explains the TOE supports both RSA (2048 and 3072-bit) and ECDSA (P-256, P-384, P-521) signing and verification. It also indicates that the TOE supports ECDSA (P-256, P-384, P-521) authentication during SSH.

Section 6.5 of the [ST] indicates that the TOE verifies RSA signatures (using RSA 3072) on firmware updates.

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 "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This includes SSH host-key and public-key algorithms. No configuration is needed for the TLS ciphersuites as stated in the ST.

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

2.2.10 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.2.10.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.3 NDcPP22e:FCS_HTTPS_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 and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Section 6.2 of [ST] states that an HTTPS/TLS connection is available which presents Web GUI and Rest API administrative interfaces. The TOE implements HTTPS per RFC 2818. A connection can be established only if the peer initiates the connection.

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

The sections entitled "Web Interface" and "Secure Remote Logging" contain instructions to configure the TOE to act as a TLS web server and TLS syslog client.

Component Testing Assurance Activities: This test is now performed as part of FIA_X509_EXT.1/Rev testing.

Tests are performed in conjunction with the TLS evaluation activities.

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

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

2.2.11 MACsec (MACSEC10:FCS_MACSEC_EXT.1)

2.2.11.1 MACSEC10:FCS_MACSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 MACSEC10:FCS_MACSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.3 MACSEC10:FCS_MACSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.4 MACSEC10:FCS_MACSEC_EXT.1.4

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 ability of the TSF to implement MACsec in accordance with IEEE 802.1AE-2018. The evaluator shall also determine that the TSS describes the ability of the TSF to derive SCI values from peer MAC address and port data and to reject traffic that does not have a valid SCI. Finally, the evaluator shall check the TSS for an assertion that only EAPOL and MACsec Ethernet frames, and MAC control frames are accepted by the MACsec interface.

Section 6.2 of [ST] states that the TOE implements MACsec in accordance with IEEE 802.1AE-2018. The TOE derives a Secure Channel Identifier (SCI) from a peer's MAC address and port data to uniquely identify the originator of a MACsec Protocol Data Unit (MPDU) and rejects any MPDUs that do not contain the identifier. Once configured on an interface, only EAPOL (PAE EtherType 88-8E), MACsec Ethernet frames (EtherType 88-E5) and MAC control frames are permitted and others are rejected.

Component Guidance Assurance Activities: None Defined

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

Test 5: The evaluator shall successfully establish a MACsec channel between the TOE and a MACsec-capable peer in the operational environment and verify that the TSF logs the communications. The evaluator shall capture the traffic between the TOE and the operational environment to determine the SCI that the TOE uses to identify the peer. The evaluator shall then configure a test system to capture traffic between the peer and the TOE to modify the SCI that is used to identify the peer. The evaluator then verifies that the TOE does not reply to this traffic and logs that the traffic was discarded.

Test 6: The evaluator shall send Ethernet traffic to the TOE's MAC address that iterates through the full range of supported EtherType values (refer to List of Documented EtherTypes) and observes that traffic for all EtherType values is discarded by the TOE except for the traffic which has an EtherType value of 88-8E, 88-E5, or 8808. Note that there are a large number of EtherType values so the evaluator is encouraged to execute a script that automatically iterates through each value.

Test 5: The evaluator configured MACsec between the TOE and a peer. The evaluator captured the SCI value that is used to negotiate the successful connection. The evaluator sent a MACsec encrypted ping packet from the peer to the TOE. The TOE responded with a MACsec reply, indicating that it accepted the peer’s MACsec packet. The

evaluator then sent a packet with an incorrect SCI value. The evaluator observed that the TOE rejected the incorrect packet and did not send a reply.

Test 6: The evaluator configured MACsec between the TOE and a peer. The evaluator then configured the peer test device to send Ethernet traffic that iterates through the range of supported EtherType values. The evaluator confirmed from packet captures that the TOE does not respond to any EtherTypes except for 88-8E or 88-E5. MACsec and EAPOL EtherTypes can be seen throughout other MACsec tests.

2.2.12 MACsec Integrity and Confidentiality (MACSEC10:FCS_MACSEC_EXT.2)

2.2.12.1 MACSEC10:FCS_MACSEC_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 MACSEC10:FCS_MACSEC_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.3 MACSEC10:FCS_MACSEC_EXT.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 verify that it describes the methods that the TOE implements to provide assurance of MACsec integrity. This should include any confidentiality offsets used, the use of an ICV (including the supported length), and ICV generation with the SAK, using the SCI as the most significant bits of the initialization vector (IV) and the 32 least significant bits of the PN as the IV.

Section 6.2 of [ST] states that the TOE implements MACsec with support for integrity protection with a confidentiality offset of 0, 30, and 50. The TSF provides assurance of the integrity of protocol data units (MPDUs) using an Integrity Check Value (ICV) of 16 bytes derived with the Secure Association Key (SAK). The TOE provides the ability to derive an Integrity Check Value Key (ICK) from a CAK using a KDF, using the SCI as the most significant bits of the Initialization Vector (IV) and the 32 least significant bits of the PN as the IV.

Component Guidance Assurance Activities: If any integrity verifications are configurable, such as the confidentiality offsets used or the mechanism used to derive an ICK, the evaluator shall verify that instructions for performing these functions are documented.

The section entitled "Configuration of Confidentiality" in [CC-Guide] provides the command that instructs the administrator how to configure and apply the confidentiality offset to a macsec-policy. This section also explains how to remove this configuration.

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

Test 7: The evaluator shall transmit MACsec traffic to the TOE from a MACsec-capable peer in the operational environment. The evaluator shall verify via packet captures, audit logs, or both that the frame bytes after the MACsec Tag values in the received traffic is not obviously predictable.

Test 8: The evaluator shall transmit valid MACsec traffic to the TOE from a MACsec-capable peer in the operational environment that is routed through a test system set up as a man-in-the-middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure.

Test 7: A valid test MACsec connection was performed in MACsecEP12:FCS_MACSEC_EXT.1-t1. The evaluator viewed that none of the frame bytes after MACsec Tag value were obviously predictable, indicating that the data was successfully encrypted.

Test 8: The evaluator configured MACsec between the TOE and a peer. The evaluator attempted to send a MACsec encrypted ICMP packet in which the encrypted packet contained a modified byte in the end of the data payload. This effectively creates an invalid packet in which the ICV does not match the entire packet. The evaluator observed that the TOE rejected the incorrect packet and did not send a reply.

2.2.13 MACsec Randomness (MACSEC10:FCS_MACSEC_EXT.3)

2.2.13.1 MACSEC10:FCS_MACSEC_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.2 MACSEC10:FCS_MACSEC_EXT.3.2

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 verify that it describes the method used to generate SAKs and nonces and that the strength of the CAK and the size of the CAK's key space are provided.

Section 6.2 of [ST] states that the TOE generates unique Secure Association Keys (SAKs) using key derivation from Connectivity Association Key (CAK) per section 9.8.1 of IEEE 802.1X-2010 and the TOE's random bit generator as specified by FCS_RBG_EXT.1 such that the likelihood of a repeating SAK is no less than 1 in 2 to the power of the size of the generated key. The TOE generates a unique nonce for the derivation of SAKs also using the TOE's random bit generator as specified by FCS_RBG_EXT.1.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Testing of the TOE's MACsec capabilities and verification of the deterministic random bit generator is sufficient to demonstrate that this SFR has been satisfied.

Refer to other MACsec test cases, which show the TOE's MACsec capabilities.

2.2.14 MACsec Key Usage (MACSEC10:FCS_MACSEC_EXT.4)

2.2.14.1 MACSEC10:FCS_MACSEC_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.2 MACSEC10:FCS_MACSEC_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.3 MACSEC10:FCS_MACSEC_EXT.4.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.4 MACSEC10:FCS_MACSEC_EXT.4.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.5 MACSEC10:FCS_MACSEC_EXT.4.5

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 SAK is wrapped prior to being distributed using the AES implementation specified in this PP-Module.

Section 6.2 of [ST] states that the TOE supports peer authentication using only pre-shared keys. The TOE distributes SAKs between MACsec peers using AES key wrap as specified in FCS_COP.1/MACSEC. The TOE supports specifying a lifetime for CAKs. The TOE associates Connectivity Association Key Names (CKNs) with CAKs that are defined by the key derivation function using the CAK as input data (per 802.1X, section 9.8.1). The TOE associates Connectivity Association Key Names (CKNs) with CAKs. The length of the CKN is an integer number of octets, between 1 and 32 (inclusive).

Component Guidance Assurance Activities: If the method of peer authentication is configurable, the evaluator shall verify that the guidance provides instructions on how to configure this. The evaluator shall also verify that the method of specifying a lifetime for CAKs is described.

The section entitled "MACsec Configuration (using pre-shared keys)" in [CC-Guide] provides instructions to define a MACsec policy using pre-shared keys. This section explains how to create a MACsec policy, create and configure an MKA policy, and apply the MACsec and MKA policy to a port range.

The section entitled "Configuration of CAK lifetime" in [CC-Guide] provides instructions to specify a lifetime for CAKs.

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

Test 9: For each supported method of peer authentication in FCS_MACSEC_EXT.4.1, the evaluator shall follow the operational guidance to configure the supported method (if applicable). The evaluator shall set up a packet sniffer between the TOE and a MACsec-capable peer in the operational environment. The evaluator shall then initiate a connection between the TOE and the peer such that authentication occurs and a secure connection is established. The evaluator shall wait one minute and then disconnect the TOE from the peer and stop the sniffer. The evaluator shall use the packet captures to verify that theSC was established via the selected mechanism and that the non-VLAN EtherType of the first data frame sent between the TOE and the peer is 88-E5.

Test 10: The evaluator shall capture traffic between the TOE and a MACsec-capable peer in the operational environment. The evaluator shall then cause the TOE to distribute a SAK to that peer, capture the MKPDUs from that operation, and verify the key is wrapped in the captured MKPDUs.

Test 9: The evaluator configured MACsec between the TOE and a peer. The evaluator started a packet capture and then stopped the packet capture after 1 minute. The evaluator verified that the first data frame sent between the TOE and the test peer has an EtherType value of x088E5. At the same time, the evaluator ensured that the TOE is the MKA server by ensuring that the test peer's MKA priority is set to the maximum value (255) and that the TOE's MKA priority is set to a higher priority. The evaluator then analyzed the packet capture and verified that the TOE sends an MKPDU packet to the test peer that contains an AES wrapped SAK.

Test 10: This test was performed as part of Test 1

2.2.15 MACsec Key Agreement - per TD0805 (MACSEC10:FCS_MKA_EXT.1)

2.2.15.1 MACSEC10:FCS_MKA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.2 MACSEC10:FCS_MKA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.3 MACSEC10:FCS_MKA_EXT.1.3

TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the methods that the TOE implements to provide assurance of MKA integrity, including the use of an ICV and the ability to use a KDF to derive an ICK.

Section 6.2 of [ST] states that the TOE provides assurance of the integrity of MKA protocol data units (MKPDUs) using an Integrity Check Value (ICV) derived from an Integrity Check Value Key (ICK). The TOE provides the ability to derive an Integrity Check Value Key (ICK) from a CAK using a KDF.

Guidance Assurance Activities: None Defined

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

Test 11: The evaluator shall transmit MKA traffic (MKPDUs) to the TOE from a MKA-capable peer in the operational environment. The evaluator shall verify via packet captures, audit logs, or both that the last 16 octets of the MKPDUs in the received traffic do not appear to be predictable.

Test 12: The evaluator shall transmit valid MKA traffic to the TOE from a MKA-capable peer in the operational environment that is routed through a test system set up as a man-in-the-middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure.

Test 11: A successful MACsec connection was established in MACsecEP12:FCS_MACSEC_EXT.1-t1 above. The evaluator observed the ICV in the MKPDU packets and determined they were not obviously predictable.

Test 12: The evaluator disabled data replay protection in the TOE in order to execute this test. The evaluator configured MACsec between the TOE and a peer. The evaluator captured a previously sent MKPDU packet and modified the last byte in the packet. The evaluator attempted to send the modified packet to the TOE. The evaluator observed the TOE successfully rejecting the packet and reporting an error in the audit log.

2.2.15.4 MACSEC10:FCS_MKA_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The tests below require the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with. Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the key server and principal actor (peer). The evaluator shall then perform the following tests using a traffic sniffer to capture this traffic:

- Test 13a: The evaluator shall configure the TOE to establish a MKA session with a new peer. The evaluator shall verify from packet captures that MKPDUs are sent at least once every two seconds or every half-second in accordance with the SFR selection.

Test 13b: (Conditional - If "EAPTLS with DevIDs" is selected in FCS_MACSEC_EXT.4.1) The evaluator shall use EAP-TLS to derive a CAK and configure the TOE's peer to send "0" in the MKA parameter field for MACsec Capability (Table 11-6 in 802.1X-2020). The evaluator shall observe that the peer is deleted from the connection after MKA Life Time has passed.

- Test 14a: (Conditional - if any "group CAK" selection is made in FCS_MKA_EXT.1.5) The evaluator shall send a fresh SAK that includes both peers as active participants. The evaluator shall start an MKA session between the TOE and the two active participant peers and send MKPDUs. The evaluator shall verify from packet captures that MKPDUs are sent at least once every two seconds or every half-second in accordance with the SFR selection.

-Test 14b: (Conditional - if any "group CAK" selection is made in FCS_MKA_EXT.1.5) Disconnect one of the peers. Arbitrarily introduce an artificial delay in sending a fresh SAK following the change in the Live Peer List. For this delayed fresh SAK, use a man-in-the-middle device to observe that the MKA Life Time of 6.0 seconds is enforced by the TSF.

(TD0787 & TD0805 applied)

Test 13a: The evaluator set up a MACsec peer to connect to the TOE. The evaluator ensured that the TOE is the Key Server by setting an MKA priority that is lower than the MACsec peers. The evaluator then started an MKA session between the TOE and the active participant peer, also taking a packet capture of the session. The evaluator analyzed the packet capture and noted that the TOE sends MKPDUs at least once every half-second.

Test 13b: Not applicable. The TOE does not support EAPTLS with DevIDs.

Test 14a: Not applicable. The TOE does not support Group CAKs

Test 14b: Not applicable. The TOE does not support Group CAKs

2.2.15.5 MACSEC10:FCS_MKA_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.6 MACSEC10:FCS_MKA_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.7 MACSEC10:FCS_MKA_EXT.1.7

TSS Assurance Activities: The evaluator shall verify that the TSS describes the TOE's compliance with IEEE 802.1X-2010 and 802.1Xbx-2014 for MKA, including the values for MKA and Hello timeout limits and support for data delay protection. The evaluator shall also verify that the TSS describes the ability of the PAE of the TOE to establish unique CAs with individual peers and group CAs using a group CAK such that a new group SAK is distributed every time the group's membership changes. The evaluator shall also verify that the TSS describes the invalid MKPDUs that are discarded automatically by the TSF in a manner that is consistent with the SFR, and that valid MKPDUs are decoded in a manner consistent with IEEE 802.1X-2010 section 11.11.4.

Section 6.2 of [ST] states that the TOE implements Key Agreement Protocol (MKA) in accordance with IEEE 802.1X-2010 and 802.1Xbx-2014. The TOE enforces an MKA Lifetime Timeout limit of 6.0 seconds and Hello Timeout limit of 2.0 seconds.

The evaluator observed that section 6.2 also explains that the TOE behaves as a Key Server which refreshes a SAK when it expires. The Key Server distributes a SAK by using a pairwise CAK. The pairwise CAK is derived from MKA or a pre-shared key. The Key Server refreshes a CAK when it expires. The Key Server distributes a fresh SAK whenever a member is added to or removed from the live membership of the CA. The TOE does not support group CAK.

This section also explains that the TOE discards invalid MKPDUs without processing and defines the conditions that indicate an invalid MKPDU. This section goes on to explain that valid MKPDUs are decoded in a manner consistent with 802.1X, section 11.11.4.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation provides instructions on how to configure the TOE to act as the key server in an environment with multiple MACsec-capable devices.

The section entitled "Key Server Priority" in [CC-Guide] explains how to set the priority of the TOE key server functionality. It explains that the value can be between 0 and 255, with 0 being the highest priority. This section states that no other configuration is necessary by the administrator to configure the TOE to behave as a key server.

Testing Assurance Activities: The tests below require the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with. Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the key server and principal actor (peer). The evaluator shall then perform the following tests:

Test 15: (Conditional - if any "group CAK" selection is made in FCS_MKA_EXT.1.5) The evaluator shall perform the following steps:

1. Load one PSK onto the TOE and device B and a second PSK onto the TOE and device C. This defines two pairwise CAs.

2. Generate a group CAK for the group of three devices using ieee8021XKayCreateNewGroup.

3. Observe via packet capture that the TOE distributes the group CAK to the two peers, protected by AES key wrap using their respective PSKs.

4. Verify that B can form an SA with C and connect securely.

5. Disable the KaY functionality of device C using ieee8021XPaePortKayMkaEnable.

6. Generate a group CAK for the TOE and B using ieee8021XKayCreateNewGroup and observe they can connect.

7. The evaluator shall have B attempt to connect to C and observe this fails.

8. Re-enable the KaY functionality of device C.

9. Invoke ieee8021XKayCreateNewGroup again.

10. Verify that both the TOE can connect to C and that B can connect to C.

Test 16: The evaluator shall start an MKA session between the TOE and the two environmental MACsec peers and then perform the following steps:

1. Send an MKPDU to the TOE's individual MAC address from a peer. Verify the frame is dropped and logged.

2. Send an MKPDU to the TOE that is less than 32 octets long. Verify the frame is dropped and logged.

3. Send an MKPDU to the TOE whose length in octets is not a multiple of 4. Verify the frame is dropped and logged.

4. Send an MKPDU to the TOE that is one byte short. Verify the frame is dropped and logged.

5. Send an MKPDU to the TOE with unknown Agility Parameter. Verify the frame is dropped and logged.

Test 15: Not applicable. The TOE does not support Group CAKs

Test 16: The evaluator set up two MACsec peers to connect to the TOE. The evaluator ensured that the TOE is the Key Server by setting an MKA priority that is lower than the two MACsec peers. The evaluator then started an MKA session between the TOE and the two active participant peers, also taking a packet capture of the session. The evaluator then sent five modified MKA packets according to the conditions identified in this test case. In each case the TOE detected the failure and rejected the packet.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.16 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.16.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.2 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 instantiates it's AES-256 CTR_DRBG with a 384-bit seed (containing a minimum of 256 bits of entropy) from one software-based noise source.

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 provide the ability to configure 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 CAVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

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

2.2.17.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.17.2 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)

The TOE supports both public key-based and password-based authentication. Section 6.2 of [ST] states that the TOE establishes a user identity when an SSH client presents an identity along with a valid public key or correct

password. The TOE allows use of the ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 algorithms for public key authentication. These algorithms match those listed in the requirement.

Section 6.2 of [ST] includes a footnote indicating that a public key must be previously installed and mapped to a TOE user account before public key authentication for that user can occur. Therefore, an SSH session can be established by a user using either a previously defined public key or the correct password for the user account specified during the login attempt.

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: 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.

Test 2: The evaluator 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: 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.

Test 4: The evaluator attempted an SSH connection using an invalid password. The evaluator was not able to log in.

2.2.17.3 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 the ST explains the TOE's SSHv2 implementation limits SSH packets to a size of 262127 bytes. Anything larger will be dropped by the TOE.

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 262127 bytes. The TOE rejected the packet and the connection was closed.

2.2.17.4 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] explains the TOE supports AES-CBC and AES-CTR (both 128 and 256 keyed variants) ciphers for data encryption and hmac-sha1/sha2-256/sha2-512 for data integrity (and does not allow the "none" MAC algorithm). These algorithms match those claimed in the ST.

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 "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This section indicates ciphers should be set to allow aes128-ctr, aes256-ctr, aes128-cbc, and aes256-cbc.

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-cbc
- aes256-cbc
- aes128-ctr
- aes256-ctr

2.2.17.5 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] explains that ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 are used for public key based authentication. These algorithms match those listed in the requirement. The TOE establishes a user identity when an SSH client presents a public key or correct password.

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 "SSH" in [CC-Guide] identifies the evaluated parameters for SSH. The section entitled "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This indicates that host-key and user public-key algorithms should be set to allow ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521.

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 generated a new RSA key pair on a client and did not configure the TOE to recognize that key pair. The subsequent connection attempt failed.

Test 3: 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.

2.2.17.6 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] explains that hmac-sha1, hmac-sha2-256, hmac-sha2-512 are the data integrity algorithms used. These algorithms match those listed in the requirement.

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 "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This indicates that SSH HMAC algorithms should be configured to allow hmac-sha2-256, hmac-sha2-512, and hmac-sha1.

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-sha1,
- 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.

2.2.17.7 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] explains that diffie-hellman-group14-sha1, ecdh-sha2-nistp256 and ecdh-sha2-nistp384 are used for SSHv2 key exchange. These algorithms match those listed in the requirement.

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 "Disabling Unsupported Algorithms" in [CC-Guide] describes how to disable unsupported cryptographic algorithms for SSH to restrict the TOE to the cryptographic behavior described in the ST. This states that key exchange algorithms should be configured ao allow ecdh-sha2-nistp256, ecdh-sha2-nistp384, and diffie-hellman-group14-sha1.

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.

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- ecdh-sha2-nistp384

2.2.17.8 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 there is a TOE initiated rekey before 1 hour or before 1GB whichever comes 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 TOE does not provide the ability to configure the thresholds. The section entitled "SSH Rekey" in [CC-Guide] states that the SSH server will perform a rekey operation for all open SSH sessions at every hour or before 1 GB of data is transferred, whichever occurs first.

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 attempted to connect to the TOE using an SSH client. The rekey time limit on the TOE is 1 hour. The evaluator performed the rekey Time Limit test and found that the TOE initiated a rekey event before 1 hour had elapsed. The evaluator configured the rekey data limit to 1MB. The evaluator performed the DATA LIMIT rekey test and found that the TOE rekeyed before 1MB of traffic had passed through the connection.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.18 TLS Client Protocol Without Mutual Authentication - per TD0670 & TD0790 (NDcPP22e:FCS_TLSC_EXT.1)

2.2.18.1 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] states that the TOE establishes a secure channel using TLS as a client for communication with an external audit server (syslog) for audit storage. The TOE supports the following cipher suites.

- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256,
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

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 "Secure Remote Logging" in [CC-Guide] includes instructions to configure the TOE to communicate with an external syslog server using a TLS protected channel. This section allows the administrator to specify the reference identifier for the syslog server, the TLS port number, the audit events to be transferred, the network interface over which the traffic shall be transmitted, the certificate to be used by the TOE in this TLS exchange and to ensure the TOE enforces key usage requirements upon certificates received from the syslog server. The TOE does not require additional configuration steps to ensure the TLS ciphersuites are in compliance with the Security Target selections for FCS_TLSC_EXT.1. The “Secure Remote Logging” section in [CC-Guide] identifies the ciphersuites supported by the TOE when acting as a TLS client as the following:

- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_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 without mutual authentication and configured the TOE to establish a TLS session with this test server. In all of the following tests cases the TOE responded in accordance with the AA and Security Target claims.

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_128_GCM_SHA256 but returns an RSA cert. 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.

2.2.18.2 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] states the TOE supports the use of FQDN and IPv4 addresses as reference identifiers within a certificate's CommonName (CN) or Subject Alternate Name (SAN) extension. The TOE checks the SAN/CN when performing certificate validation as described in NDcPP22e:FIA_X509_EXT.1/Rev. Wildcards are allowed in certificates.

IP addresses are converted to binary by parsing decimal delimited by periods. The conversion happens before any comparisons are made. Canonical format is enforced.

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 "Certificate Installation and Validation" in [CC-Guide] discusses X.509 digital certificates that are used for remote logging protection. This section explains how certificates are validated.

The section entitled "Secure Remote Logging" further states that the syslog client shall compare the syslog server's FQDN or IPv4 address against the syslog server's certificate Common Name or Subject Alternative Name. This indicates that the SAN field is not required. This section provides the command for enabling certificate name checking and provides a note that says "While IP address is supported for identity verification, it is recommended that FQDN is used for higher assurance."

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 are 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 192.168.1.20, 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).

This negative test corresponds to the following section of the Application Note 64/105: '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.

(TD0790 applied, supersedes TD0670)

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 performed tests 1 through 5 using a reference identifier that was a DNS name. The evaluator also performed tests 1 through 4 and test 6 using a reference identifier that was a IPv4.

Test 1: The TOE was configured to expect the 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 reference name configured in the TOE 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 1: The evaluator repeated the test and established a TLS session from the TOE targeting a server using a valid certificate with a CN matching the IPv4 address 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.

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

Test 6: This test was performed using IPv4 reference identifiers only. The evaluator tested hostname wildcards by configuring an expected IP4v4 on the TOE with the test server's certificates configured with wildcard IPv4 address. 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=192.168.144.*	192.168.144.254	No Connection

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

2.2.18.3 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.1.2 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.1.1 Test 4 and FIA_X509_EXT.2 Test 1.

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

2.2.18.4 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 Elliptical curves P-256, P-384, and P-521 are supported. These are not configurable.

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.

No configuration is needed to meet the Supported Elliptic Curves/Supported Groups Extension claims.

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: For ECDHE key exchanges, 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-256 curve
- ECDHE w/ P-384 curve
- ECDHE w/ P-521 curve

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.19.1 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] states that the TOE can be configured with an X509 certificate which it will send to a TLS server in response to a certificate request message sent by the TLS server.

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 Installation and Validation" in [CC-Guide] explains how to generate a CSR and import the X.509 digital certificate that is used for remote logging protection.

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 performed all TLS client tests such that the test server used for testing of FCS_TLSC_EXT.1.1 Test 1 required mutual authentication. In all of the following test cases the TOE responded in accordance with the AA and Security Target claims.

2.2.20 TLS Server Protocol Without Mutual Authentication - per TD0635 (NDcPP22e:FCS_TLSS_EXT.1)

2.2.20.1 NDcPP22e:FCS_TLSS_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 are identical to those listed for this component.

Section 6.2 of [ST] states that the TOE supports the following ciphersuites:

- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, and
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384.

The evaluator confirmed that the set of ciphersuites selected in FCS_TLSS_EXT.1.1 and the set identified in the TSS was identical.

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 (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

The section entitled "Web Interface" in [CC-Guide] contains a subsection entitled "Cipher Suites" which lists the cipher suites are permitted in the evaluated configuration. The four (4) cipher suites listed here for the Aruba CX switch acting as a TLS server match those claimed by the FCS_TLSS_EXT.1.1 selection in the [ST]. No configuration is needed to support only these cipher suites.

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

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

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

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

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

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

due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

Test 1: The evaluator attempted to connect to the TOE using each of the claimed ciphersuites. A packet capture was obtained for each connection attempt. The evaluator confirmed that each connection was successful.

Test 2: The evaluator sent a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verified that the server denied all such connection attempts. Additionally, the evaluator sent a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verified that the server denied the connection.

Test 3: (a, b): The evaluator made connection attempts from a client to the TOE. The client implementation of the TLS protocol was modified as stated in parts a and b of the assurance activity. The evaluator observed in packet captures that the connections are rejected for each scenario. Also, the evaluator established a TLS connection from a test server to the TOE, while capturing packets associated with that connection. The evaluator located the ChangeCipherSpec message (Content type hexadecimal 14) in the packet capture and found it to be followed by an EncryptedHandshakeMessage (Content type hexadecimal 16). The evaluator examined the payload of the EncryptedHandshakeMessage to determine if it contained a cleartext or encrypted version of the required Finished message. A Cleartext version would begin with a Content type hexadecimal value of 14, while an encrypted version would (for at least one of three test messages) not contain a Content type hexadecimal value of 14.

2.2.20.2 NDcPP22e:FCS_TLSS_EXT.1.2

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

Section 6.2 of [ST] states that the TOE acts as a TLS server supporting TLSv1.2 only. No older versions of TLS, and no version of SSL are supported.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

The section entitled "Cipher Suites" explains that the TOE only supports the 6 claimed cipher suites using the secp384r1 curve and with TLSv1.2.

The section entitled "Web Server Certificate" in [CC-Guide] provides the command to configure the TOE web server certificate offered during a TLS negotiation. No configuration is necessary to specify the TLS cryptographic functions offered by the TOE.

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

The evaluator attempted to establish a TLS session on the TOE with each of the unsupported versions of SSL and TLS. The evaluator used a network sniffer to capture the session negotiation and observed that the expected protocol and version were offered and rejected during negotiation.

2.2.20.3 NDcPP22e:FCS_TLSS_EXT.1.3

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

Section 6.2 of [ST] indicates that the TOE supports key exchanges using secp384r1. It also states that key exchanges are not configurable.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

The section entitled "Cipher Suites" explains that the TOE only supports the 6 claimed cipher suites using the secp384r1 curve and with TLSv1.2.

The section entitled "Web Server Certificate" in [CC-Guide] provides the command to configure the TOE web server certificate offered during a TLS negotiation.

Testing Assurance Activities: Test 1: [conditional] If ECDHE ciphersuites are supported:

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

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

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

Test 3: [conditional] If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to

perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size.

Test 1a: The evaluator attempted to establish a TLS session with the TOE when the evaluator's TLS client specified only one key exchange method in the Client Hello. The evaluator observed that the TOE only negotiated to use the ECDHE w/ P-384 curve.

Test 1b: The evaluator also attempted a connection using ECDHE w/ P-192 curve and observed the TOE rejected the connection attempt.

Test 2: Not applicable. The TOE does not support ciphersuites that use DHE key exchange.

Test 3: Not applicable. The TOE does not support ciphersuites that use RSA key exchange.

2.2.20.4 NDcPP22e:FCS_TLSS_EXT.1.4

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

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

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

Section 6.2 of [ST] states that the TOE does support session resumption using session IDs.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance. (TD0569 applied)

No configuration is needed to meet the requirement.

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

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

a) The client sends a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

b) The client verifies the server does not send a NewSessionTicket handshake message (at any point in the handshake).

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps: Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

d) The client completes the TLS handshake and captures the SessionID from the ServerHello.

e) The client sends a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).

f) The client verifies the TOE (1) implicitly rejects the SessionID by sending a ServerHello containing a different SessionID and by performing a full handshake (as shown in Figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

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

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

a) The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246).

b) The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and

performing a full handshake (as shown in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

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

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

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with a ServerHello with an empty SessionTicket extension, NewSessionTicket, ChangeCipherSpec and Finished messages (as seen in figure 2 of RFC 5077).

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

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

Test 1: Not applicable. The TOE claims to support session resumption using session ID values.

Test 2a: The evaluator first attempted a successful TLS negotiation capturing the TOE-generated session ID value from the server hello message. The evaluator initiated a new TLS connection sending the session ID value obtained during the successful negotiation. The evaluator observed that the TOE resumed the first TLS session because the TOE responded with a ServerHello message containing the original Session ID value followed by ChangeCipherSpec and Finished messages.

Test 2b: The evaluator attempted to open a TLS connection to the TOE where the evaluator's client recorded the session_id from a failed TLS negotiation. The evaluator observed the TOE reject the connection.

Test 3: Not applicable. The TOE claims to support session resumption using session ID values.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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] states that the administrator can set a lockout failure count for login attempts as the TOE's default configuration does not enforce a failed login limit. If the count is exceeded, the targeted account is locked (preventing remote administrators from logging in through SSH under the locked account/username) for an

administrator-configurable time limit. The TOE does not lock administrative access through local console, only remote/SSHv2 administrator access.

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 "Login Management" in [CC-Guide] provides instruction on configuring the number of login attempts and the lockout time, and states that the functionality does not apply for local administrators on a console session, thus guaranteeing administrator access.

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.

The evaluator performed 3 failed login attempts using a known bad password, all of which were rejected by the TOE. The evaluator immediately attempted to login using good credentials and the TOE rejected the login attempt,

showing the account had been locked. The evaluator then waited a short time less than the configured lockout duration and showed that the login attempt (using good credentials) was unsuccessful. The evaluator then waited more than the configured lockout duration and showed that the login attempt (also using good credentials) was successful.

2.3.2 Password Management - per TD0792 (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 check that the TSS lists the supported special character(s) for the composition of administrator passwords.

The evaluator shall check the TSS to ensure that the minimum_password_length parameter is configurable by a Security Administrator.

The evaluator shall check that the TSS lists the range of values supported for the minimum_password_length parameter. The listed range shall include the value of 15.

(TD0792 applied)

Section 6.3 of [ST] states that passwords can be composed of any alphabetic, numeric, and a wide range of special characters identified in the requirement. Password length can be configured by an administrator to be between 1-32 characters. The evaluator observed that the claimed range of length values does include the value 15.

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 "General password rules" in [CC-Guide] identifies possible characters that can be used in a password, guidance on how to construct a strong password, and how to set the minimum password length. It states that the minimum password length ranges from 1 to 32 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 password length was enforced by attempting to set passwords with 7 characters while a minimum length of 8 was configured (and observing the TOE reject the password).

2.3.3 Pre-Shared Key Composition (MACSEC10:FIA_PSK_EXT.1)

2.3.3.1 MACSEC10:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.3.2 MACSEC10:FIA_PSK_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 it describes the process by which the bit-based pre-shared keys are generated (if the TOE supports this functionality), and confirm that this process uses the RBG specified in FCS_RBG_EXT.1.

Section 6.3 of [ST] states that the TOE supports the use of pre-shared keys for MKA as defined by IEEE 802.1X-2010. The pre-shared keys are not generated by the TOE but rather the TOE will accept bit based pre-shared keys as a string of hexadecimal characters.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on the composition of strong PSKs, and (if the selection indicates keys of various lengths can be entered) that it provides information on the range of lengths supported.

The evaluator shall confirm the operational guidance contains instructions for either entering bit-based PSKs for each protocol identified in the requirement, or generating a bit-based pre-shared key, or both.

The section entitled "MACsec Configuration (using pre-shared keys)" in [CC-Guide] indicates that CKN and CAK are not auto-generated, are hexadecimal character values between 1 and 64 characters in length, and longer values are stronger. This section provided instructions to enter these hexadecimal keys.

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

Test 17: (conditional, the TOE supports PSKs of multiple lengths) The evaluator shall use the minimum length, the maximum length, a length inside the allowable range, and invalid lengths beyond the supported range (both higher and lower). The minimum, maximum, and included length tests should be successful, and the invalid lengths must be rejected by the TOE.

Test 18: (conditional, the TOE does not generate bit-based PSKs) The evaluator shall obtain a bit-based PSK of the appropriate length and enter it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

Test 19: (conditional, the TOE can generate bit-based PSKs) The evaluator shall generate a bit-based PSK of the appropriate length and use it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

Test 17: The evaluator configured the MACsec functionality by specifying a CAK that is used by the PSKs for MKA. The TOE accepted the correct length of the values for the CAK which is demonstrated by the Part 1a (2 hexadecimal), 1b (64 hexadecimal), 1c (32 hexadecimal) for the key sizes of 128 bits. Similarly, the TOE accepted the correct length of the values for the CAK which is demonstrated by the Part 2a (2 hexadecimal), 2b (64 hexadecimal) and 2c (32 hexadecimal) for the key sizes of 256 bits.

Test 18: This test has been performed in MACsec10:FCS_MACSEC_EXT.1-t1. The evaluator reviewed that a successful protocol negotiation can be performed with the CKN value of 1000 and CAK value of 12345678901234567890123456789012.

Test 19: The TOE does not generate bit-based PSKs, therefore this test is not applicable.

2.3.4 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.3.4.1 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.

The password is obscured by default. No configuration is necessary.

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.

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

2.3.5.1 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 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.

See 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

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

2.3.6.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.6.2 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] states that in the evaluated configuration, users can connect to the TOE via a local console or remotely using SSHv2, WebUI or RestAPI. It explains that passwords can be composed of any alphabetic, numeric, and a wide range of special characters (identified in FIA_PMG_EXT.1). Passwords can be between 1-32 characters.

Section 6.2 of [ST] states that remote SSHv2 connections can be established via both public key-based and password-based authentication, and that the TOE allows use of the ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 algorithms for public key authentication.

Section 6.3 of [ST] explains that the TOE requires users to be identified and authenticated before they can access any of the TOE functions except to display a warning banner and to permit network switching services without identification or authentication.

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 sections entitled "User, Password, and Session Management" and "Management Interfaces" in [CC-Guide] describe how to configure and log in via both the local console and remote SSH.

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 offers the several user interfaces where authentication is provided and the evaluator tested each interface (local and remote) as specified by the Security Target.

Test 1 - Using each interface the evaluator performed an unsuccessful and successful logon of each type using bad and good credentials respectively.

Test 2 - Using each interface the evaluator was able to observe the TOE displayed a banner to the user before login.

Test 3 - Using each interface the evaluator found that no functions were available to the administrator accessing the console with the exception of acknowledging the banner.

Test 4 - The TOE is not distributed, thus tests 1 through 3 above test the only TOE component.

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

2.3.7.1 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. The evaluator performed each of the following tests on this interface.

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. A successful connection was made. The evaluator then configured a syslog server that presented 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 the syslog server was refused by the TOE.

Test 2 -- The evaluator used the TOE's TLS client (syslog) to attempt connections to a test server. The test server then presented a certificate during the TLS negotiation where the certificate was expired. The TOE rejected the connection.

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. The evaluator revoked certificates in the chain (individually) and attempted the same connection from the syslog client. The attempt after revoking the certificate was not successful.

Test 4 -- The evaluator configured an OCSP responder to present a certificate that does not have the OCSP signing purpose. The evaluator established a TLS session from the TOE TLS client such that the TOE receives OCSP response signed by the invalid certificate and ensured that the TLS 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.

2.3.7.2 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 to 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 server. The evaluator observed that the TOE rejected the connections.

Test 2: The evaluator configured a test server 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 server. The evaluator observed that the TOE rejected the connection.

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 OCSP is supported for X509v3 certificate validation. Certificates are validated as part of the authentication process when they are presented to the TOE and when they are loaded into the TOE. The following fields are verified:

- Chain length

- Certificate revocation check with OCSP
- Certificate validity
- CA validity check
- Key Usage verification
- Signature verification
- SAN/CN check with wild card support

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 "Certificate Installation and Validation" in [CC-Guide] discusses X.509 digital certificates that are used for remote logging protection. The section explains how certificates are validated and provides a list of checks made. This section indicates that the TOE verifies the validity dates of all certificates within a certificate chain, uses OCSP to verify that certificate have not been revoked, and verifies the Server Authentication, Client Authentication and OCSP signing EKU bits within certificates provided by a peer.

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] explain that certificates are checked and if found not valid are not accepted or if the OCSP server cannot be contacted for validity checks, then the connection is rejected.

This behavior is consistent with the selection of "not accept the certificate" in FIA_X509_EXT.2.2.

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 "Certificate Installation and Validation" in [CC-Guide] states that X.509 digital certificates are used by the Secure Remote Logging feature. This section then provides instructions to identify the certificate used for Secure Remote Logging.

This section also explains that the TOE treats all OCSP-related failures as a failure to authenticate the peer device's certificate.

This section also states that the TOE must be configured to have the proper network access to reach the OSCP responder, and configured with an accurate time in order to ensure the OCSP responses are valid.

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 syslog server. For this channel, the TOE was an initiator of the connection from the TOE the syslog 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 syslog server. Since this was the behavior claimed in the SFR selection for a TLS connection, this test passed.

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

2.3.9.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.2 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.

Not Applicable. The ST does not select ‘device-specific information’.

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 Installation and Validation" in [CC-Guide] provides instructions for generating and installing a certificate on the TOE. This section include instructions for specifying fields for Common Name, Organization, Organizational Unit, or Country within the CSR. This section also indicates the instructions apply for the TOE syslog client certificate, but the section entitled "Web Server Certificate" explains that these same instructions apply for a certificate used by the TOE https server.

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 chaining to a CA that is not trusted cannot be imported into the TOE manually. The evaluator also tested that a certificate that chained to a trusted CA and be imported into the TOE.

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.

The TOE is not distributed so no description required.

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 Signing and Verification" in [CC-Guide] explains how to verify the version of currently active software through the “show version” command, and how to query the loaded but inactive version of the software through the “show images” command in the case of delayed activation. The section entitled "Copying the software and rebooting the switch" describes how firmware validation is performed through signature verification.

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.

The evaluator tested to determine that two commands used during a TOE update are not accepted by the TOE prior to a successful login. Any user that can login, is considered an administrator and can perform TOE updates.

The TOE is not distributed.

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] explains that the TOE offers command line functions which are accessible via the CLI. The CLI is a text-based interface which can be accessed from a directly connected terminal or via a remote terminal using SSHv2. These command line functions can be used to manage every security policy, as well as the non-security relevant aspects of the TOE. The TOE also permits administrators to perform administrative tasks using an

HTTPS/TLS protected communication channel offering a Web-based GUI and upload certificates through a RESTAPI interface.

This section provides a list of security functions (which include handling of x.509 certificates and managing the TOE's trust store) that are available to administrator once authenticated. It also states that none of these functions are available to any user before being identified and authenticated.

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.

The evaluator reviewed the guidance documentation while performing the guidance assurance activities in this AAR. The evaluator identified the TSF data manipulating functions and referenced the guidance documentation to demonstrate that it contained the corresponding configuration information. The evaluator documented these Guidance Assurance Activities throughout this AAR with the requirements to which they apply.

The section entitled "User, Password, and Session Management" in [CC-Guide] states that once in secure mode, the switch does not offer any management services or access to its management functions, except for displaying a warning banner, without requiring a user to be identified and authenticated. The evaluator observed that this explains how the TOE ensure that only administrators have access to TSF-data manipulating functions.

The section entitled "Certificate Installation and Validation" in [CC-Guide] provides instructions for managing the TOE trust store. This section explains how to configure the CA certificate of the remote syslog server's PKI, how to generate a certificate signing request (CSR) for the TOE syslog client and how to import a certificate created externally using that CSR. These steps explain how to designate a CA certificate as a trust anchor.

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 the [ST] states that only administrators can perform management operations including the command to generate, import and delete cryptographic keys as defined by Table 6-2 Key Zeroization.

These key manipulation operations are performed using commands available through the Command Line Interface.

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.

The section entitled "Certificate Installation and Validation" in [CC-Guide] provides instructions to generate keys used as part of a certificate signing request (CSR). This section also explains how to load trusted CA and TOE certificates. This sections also describes how the TOE certificate and trusted CA certificates can be listed and deleted.

The section entitled "Web Server Certificate" explains that the same procedures to load a trusted CA, generate a CSR and import a certificate are used for the Web Server Certificate as are describe in the section entitled "Certificate Installation and Validation". The administrator is simply required to issue a command to associate the new certificate with the TOE https-server.

The section entitled "SSH host key generation" describes how to generate SSH host keys. This section also indicates that when a host key is generated, it overwrites (deletes) the current key of the same type.

The section entitled “SSH authorized keys” describes how to import public keys which are used for user SSH public key authentication.

All of the above key manipulation is performed using commands available through the Command Line Interface.

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.

The evaluator attempted to import a cryptographic key (SSH public key) for a user before being authenticated as an administrator. This attempt was unsuccessful. The evaluator logged into the TOE as an authorized administrator and attempted to import a cryptographic key (SSH public key) and observed that the import was successful.

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

2.4.4.1 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.

Section 6.4 of [ST] identifies all the management functions:

- Ability to administer the TOE locally and remotely;
- Ability to configure the access banner;
- Ability to configure the session inactivity time before session termination or locking;
- Ability to update the TOE, and to verify the updates using [digital signature] capability prior to installing those updates;
- Ability to configure the authentication failure parameters for FIA_AFL.1;
- Ability to modify the behavior of the transmission of audit data to an external IT entity;
- Ability to manage the cryptographic keys,
- Ability to configure the cryptographic functionality,
- Ability to set the time which is used for time-stamps;
- Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors,
- Ability to import X.509v3 certificates to the TOE's trust store
- Ability to manage the trusted public keys database

It explains that both console and SSH CLI access provide full administrative capabilities.

Section 6.3 of [ST] explains that the local console must be co-located with the TOE and protected with equivalent physical security measures. The section entitled “Connecting to the console port” in [CC-Guide] provides a warning that the local console must be physically protected.

Component Guidance Assurance Activities: See TSS Assurance Activities

See the TSS assurance activity. The [CC-Guide] was used throughout the evaluation and the evaluator was able to perform all required functions as part of testing.

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.

2.4.5 Specification of Management Functions (MACsec) - per TD0748 (MACSEC10:FMT_SMF.1/MACSEC)

2.4.5.1 MACSEC10:FMT_SMF.1.1/MACSEC

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 describes the ability of the TOE to provide the management functions defined in this SFR.

Section 6.4 of [ST] indicates that the TOE support the following management functions which correspond to those identified by the requirement.

- Manage a PSK-based CAK and install it in the device;
- Manage the Key Server to create, delete, and activate MKA participants as specified in 802.1X-2020, sections 9.13 and 9.16 (cf. MIB object ieee8021XKayMkaParticipantEntry) and section.12.2 (cf. function createMKA());
- Specify a lifetime of a CAK; and
- Enable, disable, or delete a PSK-based CAK using the MIB object ieee8021XKayMkaPartActivateControl.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides instructions on how to perform each of the management functions defined in this SFR.

The section entitled "MACsec Configuration (using pre-shared keys)" in [CC-Guide] describes the ability of the security administrator to manage a PSK-based CAK and install it in the device. The section "Key Server Priority" in [CC-Guide] provides instructions to manage key server to create, delete, and activate MKA participants. The section "Configuration of CAK Lifetime" in [CC-Guide] specifies instructions to configure the lifetime of a CAK.

The section entitled "MACsec Configuration (using pre-shared keys)" in [CC-Guide] describes the ability of the security administrator to enable, disable or delete a pre-shared key.

Component Testing Assurance Activities: The evaluator shall set up an environment where the TOE can connect to two other MACsec devices, identified as devices B and C, with the ability of PSKs to be distributed between them. The evaluator shall configure the devices so that the TOE will be elected key server and principal actor, i.e., has highest key server priority.

The evaluator shall follow the relevant operational guidance to perform the tests listed below. Note that if the TOE claims multiple management interfaces, the tests should be performed for each interface that supports the functions.

Test 20: The evaluator shall connect to the PAE of the TOE and install a PSK. The evaluator shall then specify a CKN and that the PSK is to be used as a CAK.

Repeat this test for both 128-bit and 256-bit key sizes.

Repeat this test for a CKN of valid length (1-32 octets), and observe success.

Repeat this test again for CKN of invalid lengths zero and 33, and observe failure.

Test 21: The evaluator shall test the ability of the TOE to enable and disable MKA participants using the management function specified in the ST. The evaluator shall install PSKs in devices B and C, and take any necessary additional steps to create corresponding MKA participants. The evaluator shall disable the MKA participant on device C, then observe that the TOE can communicate with B but neither the TOE nor B can communicate with device C. The evaluator shall re-enable the MKA participant of device C and observe that the TOE is now able to communicate with devices B and C. (TD0748 applied)

Test 22: For TOEs using only PSKs, the TOE should be the key server in both tests and only one peer (B) needs to be tested. The tests are:

Test 22.1: Switch to unexpired CKN: TOE and Peer B have CKN1(10 minutes) and CKN2. CKN2 can either be configured with a longer overlapping lifetime (20 minutes) or be configured with a lifetime starting period of more than 10 minutes after the CKN1 start. The TOE and Peer B start using CKN1 and after 10 minutes, verify that the TOE expires SAK1. This can be verified by either 1) seeing the TOE immediately distribute a new SAK to the peer if the lifetime of CKN2 overlaps CKN1, or 2) by terminating the connection with CKN1 and distributing a new SAK once the lifetime period of CKN2 begins.

Test 22.2: Reject CA with expired CKN: TOE has CKN1 (10 minutes). Peer B has CKN1 (20 minutes). TOE and Peer B start using CKN1 and after 10 minutes, verify that the TOE rejects (or ignores) peer's request to use (or distribute) a SAK using CKN1.

Test 23: (conditional, 'Cause key server to generate a new group CAK...' is selected) The evaluator shall connect to the PAE of the TOE, set the management function specified in the ST (e.g., set ieee8021XKayCreateNewGroup to true), and observe that the TOE distributes a new group CAK.

Test 20: The evaluator configured the MACsec functionality by specifying a CKN and that the PSK is to be used as a CAK. The TOE accepted the correct length of the octet values for the CKN, which is demonstrated by the Part 1a (2 hexadecimal) and 1b (64 hexadecimal) for the key sizes of 128 bits. Similarly, the TOE accepted the correct length of the values for the CKN, which is demonstrated by the Part 2a (2 hexadecimal) and 2b (64 hexadecimal) for the key sizes of 256 bits.

Test 21: The evaluator configured the TOE to establish MACsec connections with peers B and C. The evaluator ensured that the TOE could establish a MACsec connection with peers B and C. The evaluator then issued a command in the TOE to disable MACsec on peer C. The evaluator observed that no client could communicate with

peer C while it was disabled. The evaluator then re-enabled MACsec on peer C and observed that communications with peer C were successful.

Test 22: The evaluator configured the TOE with CKN1, which expires in 10 minutes, and CKN2, which does not expire. The tester set up a valid MACsec channel between the TOE and the peer using CKN1. After 10 minutes, the evaluator analyzed the TOE logs and packet capture. The evaluator determined that a new SAK was distributed using CKN2.

Test 23: The TOE does not generate a new group CAK, therefore this test is not applicable.

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

2.4.6.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.6.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.6.3 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] states the TOE includes a manager account that corresponds to the required 'Authorized Administrator' also referred to as 'Security Administrator' in some requirements or text.

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 sections entitled "User, Password, and Session Management" and "Management Interfaces" in [CC-Guide] describe how to configure and log in via both the local console and remote SSH.

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.

Testing of TOE security protocols (e.g., SSH and TLS) along with the manipulation of X509 certificates was conducted using primarily the TOE CLI that is available via SSHv2. Refer to protocol testing results.

Testing of timeout values, authentication, TOE updates, self-tests, and changes to time were tested using CLI over SSH.

The TOE is not distributed.

2.5 Protection of the TSF (FPT)

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

2.5.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 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 6.5 of [ST] explains the TOE maintains and protects passwords for administrative user accounts as authentication data. Locally defined passwords are not stored in plaintext form, instead the TOE stores the

password as salted SHA-512 hashes. The TOE does not offer any functions that will disclose to any user a plain text password.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.2 Protection of CAK Data (MACSEC10:FPT_CAK_EXT.1)

2.5.2.1 MACSEC10:FPT_CAK_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 CAKs are stored and that they are unable to be viewed through an interface designed specifically for that purpose. If these values are not stored in plaintext, the TSS shall describe how they are protected or obscured.

Section 6.5 of [ST] states that CAK is stored in an encrypted form and the TOE does not offer any functions that will disclose to any user a CAK.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.3 Failure with Preservation of Secure State (MACSEC10:FPT_FLS.1)

2.5.3.1 MACSEC10:FPT_FLS.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 indicates that the TSF will shut down if a self-test failure is detected. For TOEs with redundant failover capability, the evaluator shall examine the TSS to determine that it indicates that the failed components will shut down if a self-test failure is detected.

Section 6.5 of [ST] states that if the TOE encounters a self-test failure, failure of integrity check of the TSF executable image, or failure of noise source health tests it will shut down. The TOE will not restart as long as it has a failure and will need administrator intervention.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it describes the behavior of the TOE following a self-test failure and actions that an administrator should take if it occurs.

The section entitled "MACsec Selftest" in [CC-Guide] provides instructions for the administrator to issue two commands to ensure that POST self-tests failures, FIPS module failures, and entropy failures result in secure failure. It indicates that in the event of such a failure, an EMERGENCY log will be issued and the switch will immediately reboot. The reboot will continue as long as the self-test continues to fail.

Component Testing Assurance Activities: The following test may require the vendor to provide access to a test platform that provides the evaluator with the ability to modify the TOE internals in a manner that is not provided to end customers:

Test 24: The evaluator shall modify the TSF in a way that will cause a self-test failure to occur. The evaluator shall determine that the TSF shuts down and that the behavior of the TOE is consistent with the operational guidance. The evaluator shall repeat this test for each type of self-test that can be deliberately induced to fail.

For TOEs with redundant failover capability, the evaluator shall determine that the failed components shut down and the behavior of the TOE is consistent with the operational guidance. For each component, the evaluator shall repeat each type of self-test that can be deliberately induced to fail.

Test 24: The evaluator used 3 special builds provided by the vendor to cause failures. Each build caused a specific type of failure: one caused the integrity check of the image to fail, one caused a failure of each FIPS self-tests, and one caused a failure of noise source health tests. The evaluator uploaded each build (one at a time) into boot flash on the device and then configured the TOE to boot to that failure image. During TOE initialized the failures were detected and the TOE generated an EMERGENCY log message indicating the failure and the TOE rebooted. This cycle continued until the evaluator was able to use console access to cause the TOE to boot to a valid image. The evaluator tested using each of the 3 failure images.

2.5.4 Replay Detection - per TD0746 (MACSEC10:FPT_RPL.1)

2.5.4.1 MACSEC10:FPT_RPL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.4.2 MACSEC10:FPT_RPL.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 how replay is detected for MPDUs and how replayed MPDUs are handled by the TSF.

Section 6.5 of [ST] states that the TOE detects and logs all attempts to replay MPDUs and MKA frames. This section goes on to describe the approach used by the TOE to detect replay.

The TOE detects replay by allowing an administrator to enables replay protection within the MACsec policy context with either a default or admin-specified window size. With replay protection enabled, packets are expected to arrive within the replay protection window number of packets. For example, with a window size of 10, any packet arriving out-of-sequence by more than 10 packets will be discarded. A window size of 0 (the default) enforces strict order of packet reception, discarding all packets not received in perfect sequence. The no form of this command disables replay protections and resets the window size to its 0 default.

Component Guidance Assurance Activities: None Defined

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

Before performing each test the evaluator shall successfully establish a MACsec channel between the TOE and a MACsec-capable peer in the operational environment sending enough traffic to see it working and verify the PN values increase for each direction.

Test 25: The evaluator shall set up a MACsec connection with an entity in the operational environment. The evaluator shall then capture traffic sent from this remote entity to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the remote entity where the PN values in the SecTag of these packets are less than the lowest acceptable PN for the SA. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded. (TD0746 applied)

Test 26: The evaluator will capture frames during an MKA session and record the lowest PN observed in a particular time range. The evaluator shall then send a frame with a lower PN, and then verify that this frame is dropped. The evaluator will verify that the device logged this event.

Test 25: The evaluator enabled replay protection on the TOE in order to run this test. The evaluator set up a successful MACsec connection between the TOE and a test system. The evaluator captured MACsec traffic sent from the test system to the TOE. The evaluator then attempted to send the same traffic, which contains an old packet number (PN). The TOE successfully detects the invalid PN and drops the traffic along with reporting an audit log of the event. The evaluator then attempted the same test, only this time the evaluator sent MKA traffic that was already sent. The evaluator noted that the TOE successfully detects the invalid PN, drops the traffic, and reports the error in an audit log.

Test 26: This test has been performed in MACsec10:FPT_RPL.1_t1-Part2.

2.5.5 Replay Detection for XPN - per TD0728 (MACSEC10:FPT_RPL_EXT.1)

2.5.5.1 MACSEC10:FPT_RPL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.2 MACSEC10:FPT_RPL_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 includes XPN in the description of how replay is detected for MPDUs and how replayed MPDUs are handled by the TSF.

Section 6.5 of [ST] states that the TOE supports extended packet numbering (XPN) per IEE 802.1AE-2018 using a MACsec policy configured GCM cipher suites that are based on 128-bit or 256-bit AES keys. When leveraging an XPN cipher suite, the counter used to detect replayed packets is extended to 64 bits. All other replay detection logic and mechanisms remain the same.

Component Guidance Assurance Activities: If the use of XPN or the XPN cipher suites used by the TOE are configurable, the evaluator shall examine the guidance documentation to determine that it describes how this is configured.

The section entitled "MACsec Configuration (using pre-shared keys)" in [CC-Guide] provides instructions to define a MACsec policy that includes identifying the GCM ciphersuite that is to be used by the TOE. This section indicates the TOE supports GCM cipher suites with both 128-bit and 256-bit AES keys. It also indicates that Extended Packet Numbering (XPN) cipher suites are supported. The example shows how to to configure two sample ciphersuites 'gcm-aes-256' and 'gcm-aes-xpn-256'.

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

Test 29: The evaluator shall establish a MACsec connection between the TOE and a test system using the GCM-AES-XPN-128 cipher suite if selected, otherwise use GCM-AES-XPN-256. The evaluator shall write or obtain a script to send a small frame with a known payload (such as five bytes of all zeroes) to the TOE. The evaluator shall activate a packet capture tool on the connection between the TOE and the test system and then use the test system to send this frame to the TOE 4,294,967,267 (2^32 + 1) times. The evaluator shall use the packet capture tool to verify that for the first and last frames sent, the least significant 32 bits are the same. This means the most

significant bits should have been incremented during this test. Since the IV is different the two encrypted frames should be different.

Note that if traffic is sent to the TOE at a rate of 10 GB/s, this will take approximately 5 minutes as per IEEE 802.1AE-2018.

Test 30: If both cipher suites were selected, then the evaluator shall reconfigure the TOE using the second cipher suite and rerun Test 29 to demonstrate support for both cipher suites. (TD0728 applied)

Test 29: The evaluator established a MACsec connection w/ the TOE as one peer while collecting a packet capture of the MACsec traffic. This connection used GCM-AES-XPN-256. The evaluator sent a small frame 2^32+1 times to the TOE. The evaluator observed that the first 32 bits were the same in the first frame and in the last frame. The evaluator also observed that the two encrypted frames were different.

Test 30: The evaluator repeated the prior test using GCM-AES-XPN-128 and observed the same correct behavior.

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

2.5.6.1 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 6.5 of [ST] states the TOE stores it's SSH host private keys and TLS server certificate private keys in plaintext form but does not offer any functions to output the cryptographic key value. Similarly, there is no function to view any other encrypted key.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.7.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.2 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] explains the TOE is a hardware appliance that includes a reliable real-time clock, for maintaining time. The TOE uses the clock to support several security functions including timestamps for audit records, timing elements of cryptographic functions, and inactivity timeouts. The TOE provides the administrator the ability to manually set the clock.

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 "Date and Time Configuration" in [CC-Guide] explains how to set the date and time. The section entitled "Updating Date and Time through NTP" explains that while the protocol is supported, it is not claimed nor tested in the evaluated configuration.

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 found 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.

2.5.8 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.5.8.1 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 6.5 of [ST] states that the TOE performs a suite of self-tests to verify its integrity. The TOE performs an integrity test of its firmware (by validating the firmware's RSA digital signature) product and also performs a set of power-up self-tests including AES, SHS, HMAC, RSA, ECDSA and DRBG known answer tests. The TOE automatically performs its known answer power on self-tests (POST) on its CryptoComply cryptography library by computing a trial cryptographic operation (e.g., AES encryption) and then comparing the calculated result to the known correct result (already compiled into the library). This ensures that the TOE's implementations work correctly. Should any of the tests fail, the TOE halts the boot process.

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 Tests" in [CC-Guide] explains the errors that may result from the self-tests, and recommends rebooting the device or loading new firmware in the event of a self-test error.

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.

The evaluator logged into each TOE device and initiated a restart. While the TOE rebooted, the evaluator observed messages which indicated the TOE performed the TOE self-testing as claimed by the Security Target. The TOE also generated boot messages indicating that it was the image that was about to execute.

2.5.9 TRUSTED UPDATE (NDCPP22E:FPT_TUD_EXT.1)

2.5.9.1 NDCPP22E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.9.2 NDCPP22E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.9.3 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.

Section 6.5 of [ST] explains that upgrading the ArubaOS firmware is a manual process performed by an authorized administrator. An administrator can use the “show version” and “show images” commands to query the TOE’s loaded and active firmware versions.

This section explains that the TOE firmware is digitally signed with RSA 3072 using SHA-256. The TOE uses one of two embedded (within the TOE’s firmware images) public keys to verify the digital signature. The firmware is readily available on the HPE website. Uploading the firmware to the devices does require successful authentication to the devices in order to issue the CLI commands needed to update. The TOE will validate the firmware validation during the loading process and will reject the firmware if validation fails. HPE signs the firmware images and includes the HPE signing public keys within the running firmware. Once the TOE has successfully verified a new firmware image, it is loaded and becomes active upon the next reboot.

The TOE does not claim to support automatic checking for updates and is not distributed.

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 Signing and Verification" in [CC-Guide] explains how to verify the version of currently active software through the “show version” command, and how to query the loaded but inactive version of the software through the “show images” command in the case of delayed activation. The sections entitled "Firmware Validation" and "Copying the software and rebooting the switch" describe how firmware validation is performed through signature verification and explains what happens with a successful and unsuccessful verification. The section entitled "Copying the software and rebooting the switch" describes how to copy the image onto the switch for an update.

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 has changed as expected.

Test 2: The evaluator attempted to perform a TOE update using a legitimate update that was modified using a hex editor. The TOE rejected the modified 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 modified update and the product version did not change.

The evaluator attempted to perform a TOE update using an image with a digital signature which was created using the wrong private key (i.e., wrongSignature). The TOE rejected the modified update and the product version did not change.

Test 3: Not applicable. The TOE does not use published hashes for updates.

2.6 TOE ACCESS (FTA)

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

2.6.1.1 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 6.6 of [ST] states the TOE can be configured by an administrator to set an interactive session timeout value (any integer value in minutes). The inactivity timeout is 30 minutes by default. This session timeout value is applicable to both local and remote CLI sessions. An SSHv2, Web or RestAPI remote session that is inactive (i.e., no commands issuing from the remote client) for the defined timeout value will be terminated. The TOE RestAPI interface is not interactive, but does enforce the same session timeout as the Web interface.

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 "Session Timeout" in [CC-Guide] explains how to configure the inactivity time period for both local and remote sessions through the “session-time” command. Similarly, the section entitled “Password Policy and Session Configuration for Web Interface” explains how to configure the inactivity time period for Web interface and RestAPI interface.

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, WebUI and RestAPI 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.

2.6.2 USER-INITIATED TERMINATION (NDCPP22E:FTA_SSL.4)

2.6.2.1 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 6.6 of [ST] states the TOE allows a user to terminate both local and remote sessions (including SSH, Web and RestAPI sessions). The TOE accepts the 'exit' command to terminate local and remote CLI sessions. The TOE offers a logout Web operation and a RestAPI logout URL to terminate Web and RestAPI sessions.

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 "Session Timeout" in [CC-Guide] explains how to terminate a local or remote session with the "exit" command.

The instructions to terminate a Web session is provided in the section entitled "Password Policy and Session Configuration for Web Interface", where the administrator is told how to locate the 'logout' operation on the Web interface.

The instructions to terminate a RestAPI session is provided in the section entitled "Connecting to the Rest API Interface through the Management Port". This section indicates that a logout URL is offered to terminate RestAPI sessions.

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 "exit". The evaluator observed that the session ended and a login prompt was presented.

Test 2: The evaluator repeated this test using an SSH, a Web and a RestAPI connection and observed that the session ended and the connections was terminated. The SSH connection was terminated using the 'exit'

command, the Web connection was terminated by the 'logout' operation, and the RestAPI connection was terminated by the 'logout URL'.

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

2.6.3.1 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 6.6 of [ST] states the TOE terminates local sessions that have been inactive for an administrator-configured period of time.

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 "Session Timeout" in [CC-Guide] explains how to set the inactivity timeout to ensure administrators are logged out after a period of inactive usage.

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 "session-timeout" command for periods of 1 minute, 3 minutes and 5 minutes.

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

2.6.4.1 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 6.3 of [ST] states that users can connect to the TOE via a local console or remotely using SSHv2, WebUI or RestAPI. These are the only methods the evaluator observed as being available for administration. Since the TOE REST API interface is not session-based interactive interface, there are no banners for the RestAPI interface.

Section 6.6 of [ST] states that the TOE can be configured to display a warning banner before administrators successfully establish interactive sessions with the TOE, (i.e., console, SSH CLI and WebUI), allowing users to terminate their session prior to performing any functions.

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 "Configuring Login Banner" in [CC-Admin] explains how to configure the banner messages.

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. The TOE also provides a banner for HTTPS connections prior to login.

2.7 Trusted path/channels (FTP)

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

2.7.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 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 6.7 of [ST] explains the TOE must be configured to use TLS to ensure that any exported audit records are sent only to the configured server so they are not subject to inappropriate disclosure or modification. The TOE is acting as a client in this instance and receives a certificate from the audit server for identification. See section 6.2 for a description of the TLS protocol implemented by the TOE.

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.

The section entitled "Secure Remote Logging" in [CC-Guide] explains how to configure secure remote logging, and states that the TLS tunnel must be restarted on the audit server if the connection is unintentionally broken.

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).

A successful TOE TLS connection supporting communication to an external audit server was established. Examining the packet capture from that test evaluators saw that the connection between the TOE component and the external syslog server was established; the TOE initiated the connection; and Application data that was transferred is encrypted (i.e., not plaintext).

The evaluator began a packet capture of traffic between the TOE and external audit server. With the connection established, the evaluator physically disconnected the network between the TOE and the remote audit server. The evaluator left the network disconnected for a period of time before reconnected the wiring. Because the TOE automatically reconnects broken TLS connections, the evaluator waited for the syslog server to begin receiving audit data again and stopped the packet capture shortly after traffic began flowing after the disruption. The evaluator observed that no data was transmitted unprotected.

This disconnection was performed with a short duration where the TOE simply continued use of the existing TLS session, and with a long duration where the TOE fully negotiated a new TLS session (after discarding the old session).

Upon completion of these activities, the resulting transcripts and packet captures were inspected. This data showed the following:

Test 1: The TOE support for TLS protected syslog was demonstrated.

Test 2: The TOE initiated a TLS connection for TLS protected syslog.

Test 3: Syslog communication was not plaintext.

Test 4: A physical disruption in the network resulted in a TLS session interruption and no data was transmitted unprotected.

2.7.2 Inter-TSF Trusted Channel (MACsec Communications) (MACSEC10:FTP_ITC.1/MACSEC)

2.7.2.1 MACSEC10:FTP_ITC.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.2 MACSEC10:FTP_ITC.1.2/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.3 MACSEC10:FTP_ITC.1.3/MACSEC

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: None Defined

Component Testing Assurance Activities: None Defined

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

2.7.3.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.3.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.3.3 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 6.7 of [ST] states the TOE provides SSHv2 secured remote administration. Additionally, an HTTPS/TLS connection is available which presents a Web GUI administrative interface and the RESTAPI interface. The administrator can initiate the remote session. The remote session is secured (disclosure and modification) using CAVP tested cryptographic operations, and all remote security management functions require the use of the SSHv2 protected channel or HTTPS/TLS protected channel.

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

The section entitled "User, Password, and Session Management" and "Management Interfaces" sections of the [CC-Guide] describe how to configure and log in via the remote SSH interfaces. The section entitled “Connecting to the Management Port” explains how to establish a session to the switch using the SSH and Web interface by using SSH client software or an HTTPS browser. The section entitled ‘Connecting to the Management Port’ in [CC-Guide] - provides step 3 and 4 that explain how to log into the HTTPS interface. The section entitled “Connecting to the Rest API Interface through the Management Port” in [CC-Guide] provides instructions to login using the REST API interface.

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.

The evaluator performed the following on the SSH protected CLI and the HTTPS/TLS protected WebUI:

a) The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE.
b) The evaluator connected to the TOE and performed a login using an administrator account
c) The evaluator then terminated the connection by 'Logout' and terminated the packet capture.

The evaluator performed the following on the HTTPS/TLS protected RestAPI interface:

a) The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE.
b) The evaluator initiated a RestAPI operation providing a valid username and password for an administrator.

This data showed the following:

Test 1: The TOE support for SSH protected remote administration was demonstrated.

Test 2: Remote administration sessions protected by SSH 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 [CC-Guide] to use when configuring the TOE.

The completeness of the [CC-Guide] 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.

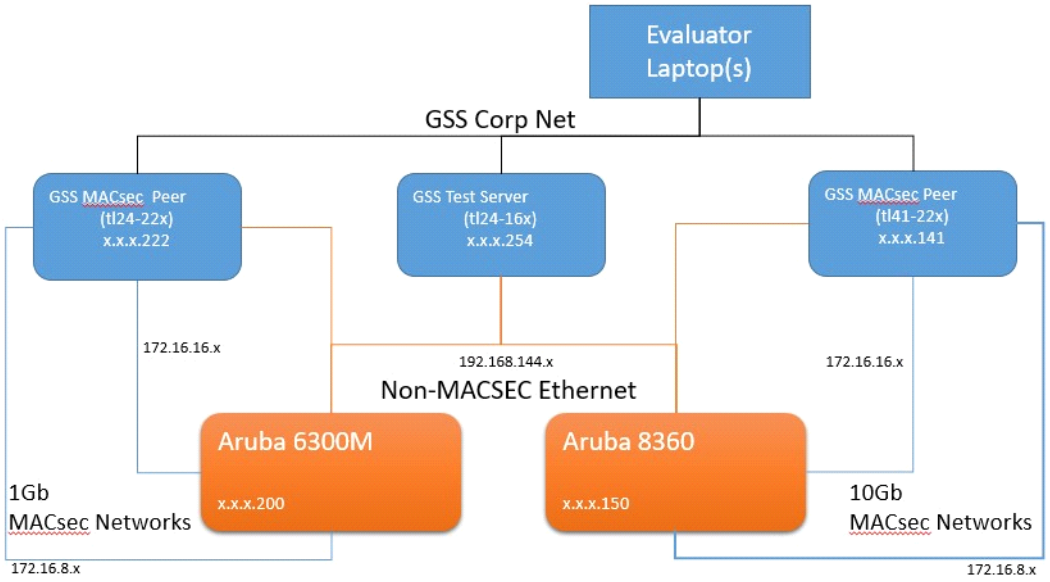
The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

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 Tested:

- Aruba 6300M running AOS 10.11
- Aruba 8360v2 running AOS 10.11

Supporting Products:

- None

Supporting Software:

The Gossamer Test servers utilized both a windows and Ubuntu environment. The Windows supporting software included the following.

- Windows 8.0, 10.0 and 11 (Evaluator Laptops)
- Wireshark version 4.0.6, 4.0.7 and 4.0.8

- Windows SSH Client - Putty version 0.74, 0.76 & 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 1.0.2g
- Openssh client version 7.2p2
- Big Packet Putty, Openssh-client version 7.2p2
- Rsyslog version 8.16.0
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Nmap version 7.01 (Linux)
- Stunnel 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 components that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic libraries, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis. (TD0547 applied)

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)
- cve.org CVE Database (https://www.cve.org/)
- 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 April 3, 2024. The search was conducted with the following search terms: "ArubaOS", "AOS 10.11", “macsec”, "TLS", "SSH", "Cortex A9", "NPX 1046A", "Xeon D-1518", "Xeon D-1527", "Xeon D-1637", "Atom C2538", "AOS-CX Cryptographic Module", "AOS-CX RSA Engine".