www.GossamerSec.com

Assurance Activity Report for Alcatel-Lucent Enterprise OmniSwitch Series 6360, 6465, 6560, 6860, 6865, 6900, 9900 with AOS 8.9 R11

Version 0.2
October 9, 2023

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

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

Document: AAR-VID11404

Revision History

Revision	Date	Authors	Summary
Version 0.1	September 11, 2023	Gossamer	Initial draft
Version 0.2	October 9, 2023	Gossamer	Respond to validation questions

The TOE Evaluation was Sponsored by:

ALE USA Inc.
2000 Corporate Center Drive
Thousand Oaks, CA 91320

Evaluation Personnel:

- Cornelius Haley
- Dip Pudasaini

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 6
1.1 Equivalence 6
1.1.1 Evaluated Platform Equivalence 6
1.1.2 CAVP Certificate Justification 8
1.2 References 9
2. Protection Profile SFR Assurance Activities 10
2.1 Security audit (FAU) 10
2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1) 10
2.1.2 User identity association (NDcPP22e:FAU_GEN.2) 12
2.1.3 Protected audit trail storage (NDcPP22e:FAU_STG.1) 12
2.1.4 Protected Audit Event Storage (NDcPP22e:FAU_STG_EXT.1) 14
2.1.5 Action in case of possible audit data loss (NDcPP22e:FAU_STG_EXT.3/LocSpace) 17
2.2 Cryptographic support (FCS) 19
2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1) 19
2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2) 22
2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4) 26
2.2.4 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS_COP.1/DataEncryption) 28
2.2.5 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash) 32
2.2.6 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS_COP.1/KeyedHash) 35
2.2.7 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS_COP.1/SigGen) 36
2.2.8 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1) 37
2.2.9 SSH Server Protocol - per TD0631 (NDcPP22e:FCS_SSHS_EXT.1) 39
2.2.10 TLS Client Protocol Without Mutual Authentication - per TD0634 & TD0670 (NDcPP22e:FCS_TLSC_EXT.1) 48
2.2.11 TLS Client Support for Mutual Authentication - per TD0670 (NDcPP22e:FCS_TLSC_EXT.2) 58
2.3 Identification and authentication (FIA) 59
2.3.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1) 59
2.3.2 Password Management (NDcPP22e:FIA_PMG_EXT.1) 62

2.3.3 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7) 64
2.3.4 Password-based Authentication Mechanism (NDcPP22e:FIA_UAU_EXT.2) 64
2.3.5 User Identification and Authentication (NDcPP22e:FIA_UIA_EXT.1) 65
2.3.6 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/Rev) 67
2.3.7 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2) 73
2.3.8 X.509 Certificate Requests (NDcPP22e:FIA_X509_EXT.3) 75
2.4 Security management (FMT) 76
2.4.1 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate) 76
2.4.2 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData) 78
2.4.3 Management of TSF Data (NDcPP22e:FMT_MTD.1/CryptoKeys) 79
2.4.4 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1) 80
2.4.5 Restrictions on Security Roles (NDcPP22e:FMT_SMR.2) 82
2.5 Protection of the TSF (FPT) 83
2.5.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1) 83
2.5.2 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1) 84
2.5.3 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1) 84
2.5.4 TSF testing (NDcPP22e:FPT_TST_EXT.1) 86
2.5.5 Trusted update (NDcPP22e:FPT_TUD_EXT.1) 88
2.6 TOE access (FTA) 93
2.6.1 TSF-initiated Termination (NDcPP22e:FTA_SSL.3) 93
2.6.2 User-initiated Termination (NDcPP22e:FTA_SSL.4) 94
2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1) 95
2.6.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1) 96
2.7 Trusted path/channels (FTP) 97
2.7.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1) 97
2.7.2 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin) 99
3. Protection Profile SAR Assurance Activities 102
3.1 Development (ADV) 102
3.1.1 Basic Functional Specification (ADV_FSP.1) 102

3.2 Guidance documents (AGD) 103
3.2.1 Operational User Guidance (AGD_OPE.1) 103
3.2.2 Preparative Procedures (AGD_PRE.1) 104
3.3 Life-cycle support (ALC) 106
3.3.1 Labelling of the TOE (ALC_CMC.1) 106
3.3.2 TOE CM Coverage (ALC_CMS.1) 106
3.4 Tests (ATE) 106
3.4.1 Independent Testing - Conformance (ATE_IND.1) 106
3.5 Vulnerability assessment (AVA) 108
3.5.1 Vulnerability Survey (AVA_VAN.1) 108

1. Introduction

This document presents evaluations results of the Alcatel-Lucent Enterprise OmniSwitch series 6360, 6465, 6560, 6860, 6865, 6900, 9900 with AOS 8.9 R11 NDcPP22e 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

This section presents the test environment and explains why the test subset was adequate to address all product installations. The TOE was tested at Gossamer's test facility in Columbia Maryland. Each device under test was configured according to the step-by-step instructions in [CC-Guide]. The evaluation includes nine models that implement all security features within software and do not rely upon hardware specific features.

Model	Software Image	Micro-architecture	Processor ID	Network Interface	Rationale for inclusion in Set of Evaluated Models
OmniSwitch 6360 (OS6360)	Nosa.img	ARM Cortex-A9	Marvell 98DX236S	Marvell AlleyCat 3	Fully Tested
			Marvell 98DX233S	Marvell AlleyCat 3	Equivalent to OS6360 by S, P & N
OmniSwitch 6465 (OS6465)	Nos.img	ARM Cortex-A9	Marvell 98DX3233	Marvell AlleyCat 3	Equivalent to OS6465T by S, P & N
OmniSwitch 6465T (OS6465T)	Nos.img	ARM Cortex-A9	Marvell 98DX3233	Marvell AlleyCat 3	Fully Tested
OmniSwitch 6560 (OS6560)	Nos.img	ARM Cortex-A9	Marvell 88F6820	Marvell AlleyCat 3	Equivalent to OS6465T by S, P & N
OmniSwitch 6860 (OS6860E)	Uos.img	ARM Cortex-A9	Broadcom BCM56342	Broadcom Helix4	Equivalent to OS6860E by S, P & N
			Broadcom BCM56340	Broadcom Helix4	Fully Tested

OmniSwitch 6860 (OS6860N)	Uosn.img	Goldmont	Intel Atom C3558	Broadcom Trident3	Fully Tested
			Intel Atom C3338	Broadcom Trident3	Equivalent to tested Uosn.img by S, P & N
OmniSwitch 6865 (OS6865)	Uos.img	ARM Cortex-A9	Broadcom BCM56342	Broadcom Helix4	Equivalent to OS6860E by S, P & N
OmniSwitch 6900 (OS6900)	Yos.img	Rangeley	Intel Atom C2538	Broadcom Tomahawk	Fully Tested
		Goldmont	Intel Atom C3558	Broadcom Trident3	Equivalent to tested Yos.img by S, P & N
		Broadwell	Intel Xeon D1518	Broadcom Trident3	Equivalent to tested Yos.img by S, P & N
OmniSwitch 9900 (OS9900)	Mos.img, Meni.img, & Mhost.img	Rangeley	Intel Atom C2518	Marvell Prestera DX	Fully Tested

S – Same software image, P – Same processor architecture & instruction set, N – Same network interfaces

The Green rows in the table above identify the models that were fully tested during the NDcPP22e evaluation. Other rows are consider evaluated as a result of equivalence. The six (6) models which were fully tested each used a unique software image. The models that were not tested each used one of the six software images that were tested.

For a device to be considered equivalent to a tested device it must run the exact same software image as a tested device. The equivalent device must be using a processor microarchitecture that is the same as a tested device. The equivalent device must also utilize network interfaces that are the same as a tested device. All processors with the same Micro-architecture are considered equivalent and each micro-architecture was tested during the evaluation.

The Intel Atom C3558 and C3338 processors are x86 microserver chips in the Denverton family and are based on the Goldmont microarchitecture. Refer to the "Goldmont" web page, https://en.wikipedia.org/wiki/Goldmont, which contains a table entitled "Server processors (Denverton)" showing these processors are in the Denverton family and have the Goldmont microarchitecture. The Atom C3558 processor was CAVP tested, but is identified in its CAVP cert as Intel Atom (Denverton0. This is strictly a naming difference, where the CAVP certificate used the processor's family, rather than microarchitecture in the CAVP certificate's OE. The Goldmont microarchitecture was CAVP tested.

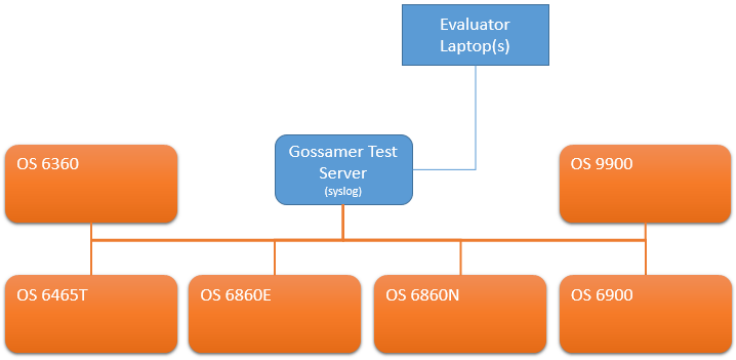

Figure 1 Test Setup

The evaluators ran all tests on all six (6) TOE devices and observed identical results.

1.1.2 CAVP Certificate Justification

The TOE is the Alcatel-Lucent Enterprise OmniSwitch series 6360, 6465, 6465T, 6560, 6860E, 6860N, 6865, 6900, and 9900 with AOS 8.9 R11 which includes a cryptographic library. This cryptographic library performs all cryptographic operations. This module is used for 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	A3688
AES-CTR (128 and 256 bits)	FCS_COP.1/DataEncryption	FIPS Pub 197 ISO 10116 NIST SP 800-38A ISO 19772	A3688
AES GCM (128 and 256 bits)	FCS_COP.1/DataEncryption	ISO 19772 FIPS Pub 197 NIST SP 800-38A	A3688
Cryptographic hashing			
SHA-1, SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	FIPS Pub 180-4 ISO/IEC 10118-3:2004	A3688
Keyed-hash message authentication			
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (digest sizes and block sizes of 256, 384 and 512 bits)	FCS_COP.1/KeyedHash	FIPS Pub 198-1 FIPS Pub 180-4 ISO/IEC 9797-2:2011	A3688

Cryptographic signature services			
RSA Digital Signature (rDSA) (2048, 3072 bits)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 9796-2	A3688
ECDSA Digital Signature (P-256, P-384, P-521)	FCS_COP.1/SigGen	FIPS Pub 186-4 ISO/IEC 14888-3	A3688
Random bit generation			
CTR_DRBG(AES), Hash_DRBG, HMAC_DRBG with sw based noise sources with a minimum of 256 bits of non-determinism	FCS_RBG_EXT.1	FIPS SP 800-90A ISO/IEC 18031:2011	A3688
Key generation			
RSA Key Generation (2048-bit)	FCS_CKM.1	FIPS Pub 186-4 ISO/IEC 9796-2	A3688
ECC Key Generation (P-256, P-384, P-521)	FCS_CKM.1	FIPS PUB 186-4	A3688
FFC Scheme using Diffie-Hellman Group 14	FCS_CKM.1	NIST SP 800-56A Rev 3	Tested with known good impl
Key establishment			
RSA	FCS_CKM.2	RSAES-PKCS1-v1_5	Tested with known good impl
KAS ECC	FCS_CKM.2	NIST SP 800-56A Rev 3	A3688
FFC Schemes using 'safe-prime' groups	FCS_CKM.2	NIST SP 800-56A Rev 3	Tested with known good impl

1.2 References

The following evidence was used to complete the Assurance Activities:

[ST] Alcatel-Lucent Enterprise OmniSwitch series 6360, 6465, 6560, 6860, 6865, 6900, 9900 with AOS 8.9 R11 Security Target

[CC-Guide] Preparation and Operation of Common Criteria Evaluated OmniSwitch Products (NDcPP), AOS Release 8.9.R11, July 2023.

2. Protection Profile SFR Assurance Activities

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For the administrative task of generating/import of, changing, or deleting of cryptographic keys as defined in FAU_GEN.1.1c, the TSS should identify what information is logged to identify the relevant key.

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

Section 6.1 of [ST] states that the for certificate management (e.g., generation, import, update, deletion, view and verification of certificates), the audit record includes the full "aaa certificate" command line entered by the administrator, which provides the certificates and/or keys involved in the operation, and whether the operation succeeded or failed. The audit record also includes the description of the error in case of a failure.

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation and ensure that it provides an example of each auditable event required by FAU_GEN.1 (i.e. at least one instance of each

auditable event, comprising the mandatory, optional and selection-based SFR sections as applicable, shall be provided from the actual audit record).

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

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

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

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

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

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

The evaluator created a list of the required audit events. The evaluator then collected the audit event when running the other security functional tests described by the protection profiles. For example, the required event for FPT_STM_EXT.1 is Changes to Time. The evaluator collected these audit records when modifying the clock

using administrative commands. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.2 User identity association (NDcPP22e:FAU_GEN.2)

2.1.2.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the Guidance Documentation requirements for 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.1.1.

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

2.1.3.1 NDcPP22e:FAU_STG.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 NDcPP22e:FAU_STG.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 of [ST] explains that audit data is stored in the audit file set located in the flash file system. The amount of audit data that can be generated depends on the maximum size allowed per audit file, which can be changed using the command: swlog output flash file-size <size in KB>.

This section also states that audit files are protected from modification and deletion by enforcing filesystem access control on groups of commands. Only the Security Administrator has privileges to clear the audit records.

The TOE is not distributed.

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

The section entitled "Protected Audit Trail Storage (FAU_STG.1)" in [CC-Guide] explains that the audit files are protected from deletion and modification. Only the Security Administrator can delete or modify the audit files. The audit files can be cleared using the CLI command “swlog clear all”.

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

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

no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to access the audit trail can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

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

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

Test 1: The TOE allows administrators to be granted read only permissions. The evaluator defined a read-only administrator account, logged in using that account, and showed that attempts to clear, remove or delete the audit logs were rejected.

Test 2: The evaluator logged in as an administrator and ran the command to clear all audit data and observed that the TOE deleted the current audit records and continued logging.

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] explains that the TOE audit functionality can be also configured to send the audit events to an external syslog server using TLS for protecting the communication channel. Audit events are stored locally in the flash memory using a circular chain of audit files. Whenever the audit file reaches a configurable threshold (maximum size for audit files), the TOE overwrites the content of the oldest audit file in the set of circular audit files (in case the set is full).

The TOE is not distributed.

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

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

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

The section entitled "Security Audit Event Storage (Extended - FAU_STG_EXT)" in [CC-Guide] explains that the TOE provides option to configure external syslog server for transmitting the audit events to an external audit server. Audit server can be configured in the switch using the ***swlog output socket <domain_name> tls*** command. OmniSwitch establishes a secure connection over TLS connection with the configured audit server. The audit logs for various events are stored locally in the OmniSwitch; simultaneously the logs are transmitted over the secure TLS connection to the configured audit server.

This section further describes that the audit data of all events is captured in SWLOG file. The amount of the audit data that can be captured depends on the size of the SWLOG files. The size (in kilobytes) of the SWLOG file for storing the audit data locally can be configured using the CLI command "swlog output flash-file-size <kilobytes>". The allowed values for the maximum size of the audit log files are 125 to 12500 Kilobytes. By default, the switch logging file size is set to 1250 kilobytes. The total audit storage capacity would be eight times the SWLOG size configured through the above command. The default total audit storage capacity is 8*1250 Kilobytes.

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

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

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

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

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

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

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

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

Test 1: The evaluator configured the system (per guidance) to securely transfer audit data. The evaluator then 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 evaluator generated audit data until the local storage space of 90% was exceeded. The evaluator verified that when the local audit storage of 90% was filled to the maximum, the existing audit data was overwritten based on the following rule: overwrite oldest records first.

Test 3 & 4 are not applicable.

2.1.5 Action in case of possible audit data loss (NDcPP22e:FAU_STG_EXT.3/LocSpace)

2.1.5.1 NDcPP22e:FAU_STG_EXT.3.1/LocSpace

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

The evaluator shall examine the TSS to ensure that it details how the Security Administrator is warned before the local storage for audit data is full.

For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how each TOE component realises this SFR. Since this SFR is optional, it might only apply to some TOE components but not all. This might lead to the situation where all TOE components store their audit information themselves but FAU_STG_EXT.3/LocSpace is supported only by one of the components. In particular, the evaluator has to verify, that the TSS describes for every component supporting this functionality, whether the warning is generated by the component itself or through another component and name the corresponding component in the latter case. The evaluator has to verify that the TSS makes clear any situations in which audit records might be 'invisibly lost'.

Section 6.1 of [ST] explains that the TOE provides a mechanism to display a warning to the administrator if the storage capacity has reached 90% of the configured size in any of the audit files.

The TOE is not distributed.

Component Guidance Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

The evaluator shall also ensure that the guidance documentation describes how the Security Administrator is warned before the local storage for audit data is full and how this warning is displayed or stored (since there is no guarantee that an administrator session is running at the time the warning is issued, it is probably stored in the log files). The description in the guidance documentation shall correspond to the description in the TSS.

The section entitled "Security Audit Event Storage (Extended - FAU_STG_EXT)" in [CC-Guide] also explains that there is a mechanism to display a warning to the user if the storage capacity has reached a configurable threshold limit on the configured size of the SWLOG files. The warning message is appended to audit log file. The location of the audit log file has been described in above paragraph.

The default threshold limit before a warning message is generated by the OmniSwitch is 90%. This Threshold limit of the storage space value can be changed using the CLI command ***swlog size-trap-threshold <threshold>***.

Component Testing Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

The evaluator shall verify that a warning is issued by the TOE before the local storage space for audit data is full.

For distributed TOEs the evaluator shall verify the correct implementation of display warning for local storage space for all TOE components that are supporting this feature according to the description in the TSS. The evaluator shall verify that each component that supports this feature according to the description in the TSS is capable of generating a warning itself or through another component.

The TOE generates a warning message as an audit event. While performing FAU_STG_EXT.1 tests, the evaluator verified that the warning audit was properly generated.

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] states that the TOE supports asymmetric key generation schemes using RSA, ECC and FFC safe-primes. The TOE generates RSA and ECDSA asymmetric cryptographic keys that are used to protect communications for TLSv1.2 and SSHv2. The TOE acts as a client for TLS generating (RSA and ECC keys) during key exchanges and a server for SSH generating (RSA, ECC and FFC Safe-prime) during key exchanges. RSA and ECDSA keys can also be generated during the creation of a Certificate Signing Request (CSR) or for use as an SSH host key.

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 "Cryptographic Key Generation (FCS_CKM.1)" in [CC-Guide] explains that the TOE generates keys for use with a CSR, for use as an SSH hostkey, and as part of TLS and SSH session negotiations. This section describes that the TOE supports the following:

- RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.3;
- ECC schemes using “NIST curves" P-256, P-384, P-521 that meet the following: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4; and

- FFC schemes using 'safe-prime' groups that meet the following: 'NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography' and RFC3526.

This section refers the administrator to section entitled "Generate a certificate signing request" for instructions to create RSA or ECDSA certificate signing request (CSR). The administrator uses the CSR to obtain a certificate from an external Certificate Authority. The administrator must then import the certificate into the OminSwitch to be used in TLS negotiations with external entities (e.g., a syslog server).

This section explains that the TOE implementation of SSH uses RSA-SHA2-256, RSA-SHA2-512, ECDSA SHA2 NISTP256, ECDSA SHA2 NISTP384 and ECDSA SHA2 NISTP521 keys for session establishment. This section explains that SSH hostkey generation automatically during reload with RSA (sizes of 2048-bit or greater) and ECDSA SHA2 NISTP256 algorithms using the ***ssh-keygen*** command.

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.

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

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)

Section 6.2 of [ST] states that the TOE performs key establishment based on RSA and ECDSA asymmetric cryptographic keys that are used to protect communications for TLSv1.2 and SSHv2. Section 6.2 of [ST] includes table 6-4 that maps RSA, ECC and FFC Safe-primes key establishment to SSHv2, as well as mapping RSA and ECC Key Establishment to TLSv1.2.

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 "Cryptographic Key Establishment (FCS_CKM.2)" in [CC-Guide] explains that with CC mode enabled, the TOE supports the below cryptographic key establishment methods:

- RSA-based key establishment schemes that meet the following:

  RSAES-PKCS1-v1_5 as specified in Section 7.2 of [RFC8017], "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1";

- Elliptic curve-based key establishment schemes that meets the following: NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”;
- FFC schemes using 'safe-prime' groups that meet the following: 'NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography' and RFC3526

This section further explains that session negotiation includes an exchange of asymmetric public keys and derivation of symmetric session keys, implemented for both SSH and TLS secure channel. Once the session is initialized and functioning, all traffic between endpoints is communicated over a secure channel, using the negotiated encryption algorithm and key size. For both SSH and TLS there are no specific user configuration for selecting the key establishment, it is based on negotiation during session establishment.

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.

Gossamer tested against a 3rd party implementation to ensure the TOE is complaint with the RFC for Diffie-Hellman group 14 and group 16. For SSH, that validation of a good implementation of diffie-hellman-group14-sha1, diffie-hellman-group14-sha256, and diffie-hellman-group16-sha512 was performed under FCS_SSHS_EXT.1.7 Test 2. During the use of this test, OpenSSH was leveraged to validate the TOE's implementation of a good implementation of these DH groups.

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 fthe 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 6.2.2 of [ST] contains Table 6-5 that identifies the keys used by the TLS protocol while section 6.2.3 contains Table 6-7 that identifies the keys used by the SSHv2 protocol. Together these tables list all of the keys that support any of the SFRs in the Security Target. Section 6.2.4 also states that the TOE destroys key material by overwriting it with zeroes and releasing the allocated memory only after it was properly destroyed. Persistent keys such as keys associated with certificates are stored in non-volatile storage and destroyed based upon user commands. Ephemeral keys such as session keys, shared secrets, and keys within certificates received during a protocol negotiation are held in RAM only and undergo Zeroization and deallocation when session terminates. These claims are consistent with the function of the TOE observed by the evaluator during testing.

Component Guidance Assurance Activities: A TOE may be subject to situations that could prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and any other supporting information used). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command [Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table)] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

The section entitled "Cryptographic Key Destruction (FCS_CKM.4)" in [CC-Guide] explains that the cryptographic key destruction is carried out by the internal operation of the OmniSwitch software according to the type of storage and user operation is not involved. The TOE is not subject to any delays in cryptographic key destruction.

Component Testing Assurance Activities: None Defined

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

2.2.4.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 of [ST] indicates that the TOE provides symmetric encryption and decryption capabilities using AES in CBC mode (128 and 256 bit key sizes), AES in CTR mode (128 and 256 bit key sizes) as well as using AES in GCM mode (128 and 256 bit key sizes). AES is implemented in support of TLS and SSH protocols.

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 "Cryptographic Operation (FCS_COP)" in [CC-Guide] contains a table that shows with CC mode enabled the TOE supports TLSv1.2 and SSHv2 protocols using the following data encryption and decryption algorithms:

- AES (CBC mode) with 128-bit and 256-bit keys,
- AES (CTR mode) with 128-bit and 256-bit keys, and
- AES (GCM mode) with 128-bit and 256-bit keys.

This section also explains that there is no specific configuration for selection of key sizes offered by the TOE. Enabling CC mode automatically limits the key sizes offered to those allowed in an evaluated configuration. The key size used for a specific session is selected based on the negotiation during SSH and TLS session establishment.

There is no user specific configuration for all the crypto algorithms offered by the TOE during SSH and TLS connection establishment. Enabling CC mode automatically limits the key sizes offered to those allowed in an evaluated configuration. The cryptographic algorithms used for a specific SSH or TLS session are selected based on the negotiation during SSH/TLS connection establishment.

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.

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

2.2.5.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 of [ST] explains that the TOE supports hashing using SHA-1, SHA-256, SHA-384 and SHA-512 with message digest sizes 160, 256, 384, and 512. These algorithms were validated as conforming to FIPS 180-4, Secure Hash Standard (SHS). Hashing is used within several services including TLS and SSH. SHA-256 is used to store passwords and in conjunction with published hashes for verification of software image integrity.

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 "Cryptographic Operation (FCS_COP)" in [CC-Guide] contains a table that shows with CC mode enabled the TOE supports TLSv1.2 and SSHv2 protocols using the following hash algorithms:

- SHA-1,
- SHA-256,
- SHA-512, and
- SHA-384.

This section also explains that there is no specific configuration for selection of key sizes offered by the TOE. Enabling CC mode automatically limits the key sizes offered to those allowed in an evaluated configuration. The key size used for a specific session is selected based on the negotiation during SSH and TLS session establishment.

There is no user specific configuration for all the crypto algorithms offered by the TOE during SSH and TLS connection establishment. Enabling CC mode automatically limits the key sizes offered to those allowed in an evaluated configuration. The cryptographic algorithms used for a specific SSH or TLS session are selected based on the negotiation during SSH/TLS connection establishment.

Component Testing Assurance Activities: The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

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

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

2.2.6.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 of [ST] explains that the TOE supports keyed hash HMAC-SHA1, HMAC-SHA256, and HMAC-SHA384, HMAC-SHA512 conforming to ISO/IEC 9797-2:2011. Supported cryptographic key sizes: 160, 256, 384, 512 bits and message digest sizes: 160, 256, 384, 512 bits. Keyed hash use matches validated hash algorithms implemented by the TOE.

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 "Cryptographic Operation (FCS_COP)" in [CC-Guide] contains a table that shows with CC mode enabled the TOE supports TLSv1.2 and SSHv2 protocols using the following keyed hash algorithms.

- HMAC-SHA-1 with 160-bit key (for SSHv2 and TLSv1.2);
- HMAC-SHA-256 with 256-bit key (for SSHv2 and TLSv1.2);
- HMAC-SHA-512 with 512-bit key (for SSHv2); and
- HMAC-SHA-384 with 384-bit key (for SSHv2 and TLSv1.2).

This section also explains that there is no specific configuration for selection of key sizes offered by the TOE. Enabling CC mode automatically limits the key sizes offered to those allowed in an evaluated configuration. The key size used for a specific session is selected based on the negotiation during SSH and TLS session establishment.

There is no user specific configuration for all the crypto algorithms offered by the TOE during SSH and TLS connection establishment. Enabling CC mode automatically limits the key sizes offered to those allowed in an evaluated configuration. The cryptographic algorithms used for a specific SSH or TLS session are selected based on the negotiation during SSH/TLS connection establishment.

Component Testing Assurance Activities: For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate

HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and message data using a known good implementation.

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

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

2.2.7.1 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.

Table 6-1 in Section 6.2 of [ST] indicates that RSA Digital Signature Algorithm with 2048 and 3072 bit key sizes are used for cryptographic signatures. It also indicates that the TOE can support ECDSA Digital Signatures with a key size of 256 bits or greater for cryptographic signatures (specifically NIST curves P-256, P-384, or P-521).

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

The section entitled "Cryptographic Operation (FCS_COP)" in [CC-Guide] contains a table that shows with CC mode enabled the TOE supports TLSv1.2 and SSHv2 protocols using signature generation and verification with the following supported algorithms.

- RSA PKCS#1 v1.5 with SHA-1, SHA-256, SHA-384 and SHA-512, using 2048-bit and 3072-bit keys;
- ECDSA with SHA-256, SHA-384 and SHA-512 using NIST P-256, P-384, and P-521 curves.

The signature algorithms used by an SSHv2 or TLSv1.2 session is determined by the negotiated cryptographic parameters and authentication methods. With CC mode enabled, the TOE only supports these signature generation and verification algorithms.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator generates or obtains 10 messages for each modulus size/SHA combination supported by the TOE. The TOE generates and returns the corresponding signatures.

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

Signature Verification Test

For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (d, e). Each private key d is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, e, messages, or signatures are altered so that signature verification should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key e values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

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

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

2.2.8 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.8.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.8.2 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] states that the TOE OpenSSL library includes a Deterministic Random Bit Generator (DRBG) used for key generation and random data (e.g. shared secrets). The DRBG uses the Hash_DRBG with SHA-256 algorithm by default; if there is a failure in the instantiation, the DRBG will fall back to CTR_DRBG using the AES-256 algorithm, and then to HMAC_DRBG using the HMAC-SHA-256 algorithm. These OpenSSL DRBG are seeded from a software-based NDRNG based on HAVEGE (Hardware Volatile Entropy Gathering and Expansion).

The Entropy description is provided in a separate (non-ST) document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

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

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

The section entitled "Extended: Random bit generation (FCS_RBG_EXT.1)" in [CC-Guide] describes that the TOE performs all deterministic random bit generation (DRBG) services in accordance with ISO/IEC 18031:2011. It uses HASH_DRBG, CTR_DRBG and HMAC_DRBG of the openssl package for providing randomness to the cryptographic operation. This is an internal operation of the TOE and no user operation is involved.

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

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input

and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

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

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

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

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

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

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

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

2.2.9.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.2 NDcPP22e:FCS_SSHS_EXT.1.2

TSS Assurance Activities: The evaluator shall check to ensure that the TSS contains a list of supported public key algorithms that are accepted for client authentication and that this list is consistent with signature verification

algorithms selected in FCS_COP.1/SigGen (e.g., accepting EC keys requires corresponding Elliptic Curve Digital Signature algorithm claims).

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

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

Section 6.2 of [ST] indicates that the TOE SSHv2 implementation supports authentication methods using public-keys and passwords. It also states that when a user attempts to establish a SSHv2 session, the TOE verifies that the user has a public key. Section 6.2 contains Table 6-6 which indicates that public keys can be verified using RSAPKCS#1v1.5 with SHA-256 and SHA-512 (rsa-sha2-256 or rsa-sha2-512) or using ECDSA withSHA-256, SHA-384 and SHA-512, and NIST curves P-256, P-384 and P-521 (ecdsa-sha2-nistp256, ecdsa-sha2-nistp384 or ecdsa-sha2-nistp521).The evaluator confirmed this list corresponds with the FCS_COP.1/SigGen selections in [ST] which include RSA and ECDSA using curves P-256, P-384 and P-521.

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. This test was repeated using rsa, rsa-sha2-256, rsa-sha2-512, ECDSA-p256, ECDSA-p384 and ECDSA-p521 keys.

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.

2.2.9.3 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 6.2.3 of [ST] states that the TOE limits the size of SSHv2 packets to 262126 bytes; packets greater than this size in an SSH transport connection are dropped.

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 262126 bytes. The TOE rejected the packet and the connection was closed.

2.2.9.4 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 6.2.3 of [ST] states that the TOE implements the Secure Shell version 2 (SSHv2) protocol. The SSHv2 protocol complies with [RFC4251], [RFC4252], [RFC4253], [RFC4254], [RFC4344], [RFC5656], [RFC6668] and [RFC8332]. This section also contains Table 6-6 which shows that the TOE supports SSHv2 encryption algorithms for AES128-cbc, aes256-cbc, aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com.

The evaluator confirmed that FCS_SSHS_EXT.1.4 included the following selection: aes128-cbc, aes256-cbc, aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com.

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 "Enable CC Mode" in [CC-Guide] explains that OmniSwitch runs in the Common Criteria mode under which it will have all the Common Criteria functions enabled.

The Common Criteria mode will only allow console and SSH access to the OmniSwitch. In Common Criteria mode the cryptographic algorithms for SSH are limited to only evaluated encryption algorithms, key exchanges, public key algorithms and data integrity MAC algorithms. The evaluator further ensured that the section entitled "Enable CC Mode" included the commands to deploy the TOE in the CC enabled mode.

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
- aes128-gcm@openssh.com
- aes256- gcm@openssh.com

2.2.9.5 NDcPP22e:FCS_SSHS_EXT.1.5

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

Section 6.2 of [ST] states that the TOE host public key authentication can be performed using RSAPKCS#1v1.5 with SHA-256 and SHA-512. This translates to rsa-sha2-256 and rsa-sha2-512 host key algorithms. The TOE can also

support ECDSA host key algorithms using ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521. The evaluator confirmed that FCS_SSHS_EXT.1.5 included these selections.

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 "Enable CC Mode" in [CC-Guide] explains that OmniSwitch runs in the Common Criteria mode under which it will have all the Common Criteria functions enabled.

The Common Criteria mode will only allow console and SSH access to the OmniSwitch. In Common Criteria mode the cryptographic algorithms for SSH are limited to only evaluated encryption algorithms, key exchanges, public key algorithms and data integrity MAC algorithms. The evaluator further ensured that the section entitled "Enable CC Mode" included the commands to deploy the TOE in the CC enabled mode.

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:

- rsa-sha2-256,
- rsa-sha2-512,
- 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.

2.2.9.6 NDcPP22e:FCS_SSHS_EXT.1.6

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

Section 6.2 of [ST] states contains Table 6-6 which shows that the TOE supports HMAC-SHA1, HMAC-SHA1-96, HMAC-SHA2-256 and HMAC-SHA2-512 integrity algorithms. The evaluator confirmed that FCS_SSHS_EXT.1.6 included the following selection: hmac-sha1, hmac-sha1-96 hmac-sha2-256, hmac-sha2-512. The evaluator also noted that Table 6-6 indicate the TOE supports aes128-gcm@openssh.com and aes256-gcm@openssh.com which supports the requirement selection of "implicit".

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 "Enable CC Mode" in [CC-Guide] explains that OmniSwitch runs in the Common Criteria mode under which it will have all the Common Criteria functions enabled.

The Common Criteria mode will only allow console and SSH access to the OmniSwitch. In Common Criteria mode the cryptographic algorithms for SSH are limited to only evaluated encryption algorithms, key exchanges, public key algorithms and data integrity MAC algorithms. The evaluator further ensured that the section entitled "Enable CC Mode" included the commands to deploy the TOE in the CC enabled mode.

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-sha1-96,
- hmac-sha2-256,
- hmac-sha2-512,
- implicit

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

2.2.9.7 NDcPP22e:FCS_SSHS_EXT.1.7

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

Section 6.2 of [ST] states contains Table 6-6 which shows that the TOE supports key establishment (key exchange) using diffie-hellman-group14-sha1, diffie-hellman-group14-sha256, as well as ecdh-sha2-nistp256, ecdh-sha2-nistp384 and ecdh-sha2-nistp512. The evaluator confirmed that FCS_SSHS_EXT.17 included these same key exchange methods.

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 "Enable CC Mode" in [CC-Guide] explains that OmniSwitch runs in the Common Criteria mode under which it will have all the Common Criteria functions enabled.

The Common Criteria mode will only allow console and SSH access to the OmniSwitch. In Common Criteria mode the cryptographic algorithms for SSH are limited to only evaluated encryption algorithms, key exchanges, public key algorithms and data integrity MAC algorithms. The evaluator further ensured that the section entitled "Enable CC Mode" included the commands to deploy the TOE in the CC enabled mode.

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,
- diffie-hellman-group14-sha256,
- diffie-hellman-group16-sha512,
- ecdh-sha2-nistp256,
- ecdh-sha2-nistp384,
- ecdh-sha2-nistp521

2.2.9.8 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 6.2.3 of [ST] indicates that the TOE controls that the SSHv2 session does not transmit more than 228 packets (less than 1GB) or that the duration of the session is more than one hour using the same session key. If the data transmission threshold is surpassed, or the session time limit is reached, new session keys are established between both ends.

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

The section entitled SSH Server (FCS_SSHS_EXT.1) in [CC-Guide] explains that within SSH connections the same session keys are used for a threshold of no longer than one hour (3600s), and no more than 2^30 bytes of transmitted data. After either of the thresholds are reached a rekey needs to be performed. These are the default rekey values that cannot be modified by the administrator.

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 rekey data limit on the TOE is 1GB. The evaluator performed the Data Limit rekey test and found that the TOE rekeyed before 1GB 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

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

2.2.10.1 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 6.2.2 of [ST] indicates 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_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_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
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_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 "Enable CC Mode" in [CC-Guide] explains that OmniSwitch runs in the Common Criteria mode under which it will have all the Common Criteria functions enabled.

The section entitled "TLS Client Protocol (FCS_TLSC_EXT.2) and (FCS_TLSC_EXT.1.1)" in [CC-Guide] explains that administrator can configure custom cipher suites using the ***ssl cipher*** command. This section provides a table that

shows the TLS 1.2 ciphersuites and their equivalent OpenSSL cipher names which are used to create the cipher list for the above command. These are the only ciphers that are allowed in an evaluated configuration.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Test 3: The evaluator established a TLS session from the TOE. A modified test server negotiates TLS_RSA_WITH_AES_128_CBC_SHA, but returns a DSA Certificate. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is not negotiated successfully.

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

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

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

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

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

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

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

2.2.10.2 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] indicates that when acting as a TLS client, the TOE verifies that the certificate presented by the TLS server during the TLS handshake corresponds to the server, using one of the following methods.

- If the server certificate includes a DNS reference identifier in a Subject Alternative Name (SAN) field, verify that the DNS reference identifier matches the server hostname, according to [RFC6125]/
- If the server certificate does not include a DNS reference identifier in a Subject Alternative Name (SAN) field, verify that the DNS reference identifier provided in the Common Name (CN) matches the server hostname, according to [RFC6125].

The TOE does not support the use of wildcards in DNS names included in certificates.

The TOE is not distributed.

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

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

The section entitled "TLS Client Protocol (FCS_TLSC_EXT.2) and (FCS_TLSC_EXT.1.1)" in [CC-Guide] explains that the configured server Hostname for Syslog client application is being used as a reference identifier for validating the TLS certificate. OmniSwitch supports CN (Common Name) and SAN (Subject Alternative Name). However, it does not support constructing reference identifiers (domain name) using wildcards in certificates.

This section refers the admin to an earlier section entitled "Security Audit Event Storage (Extended - FAU_STG_EXT)" where instructions are provided to cause the TOE to transmit the generated audit data to an external entity using TLS. While the TOE is capable of being configured to accept an IP address or IPv6 address to identify the external entity for a TLS connection, that form of reference identifier was not part of the evaluated configuration. The TOE requires the FQDN be configured as the external entity reference identifier to allow matching of the CN or SAN within the certificate presented by the external entity during TLS negotiation against the FQDN configured on the TOE.

The TOE is not distributed, therefore there is not FPT_ITT.1 channel.

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

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

or

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

or

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

Note that for some tests additional conditions apply.

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

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

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

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

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

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

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

each supported SAN type (e.g. IPv4, IPv6, FQDN, URI). When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the SAN.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

The TOE utilizes TLS for FTP_ITC.1 and FTP_TRP communications, therefore tests 1 through 5 are applicable.

The evaluator performed tests 1 through 5 using a reference identifier that was a DNS name.

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

Test 2: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier

in the SAN that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

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

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

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

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

Test 6: The TOE does not support IP address identifiers in the SAN or CN, therefore this test is not applicable.

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

2.2.10.3 NDcPP22e:FCS_TLSC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

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

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

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

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

2.2.10.4 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 6.2.2 of [ST] states that the TOE implements the Supported Elliptic Curves Extension according to [RFC4492] with NIST curves secp256r1, secp384r1, and secp521r1. This behavior is performed by default and there is no security management function to disable it. The evaluator confirmed that FCS_TLSC_EXT.1.4 included the following selection: secp256r1, secp384r1, and secp521r1.

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

The section entitled "TLS Client Protocol (FCS_TLSC_EXT.2) and (FCS_TLSC_EXT.1.1)" in [CC-Guide] explains that the TOE implements the Supported Elliptic Curves Extension according to [RFC4492] with NIST curves secp256r1, secp384r1, and secp521r1. This behavior is performed by default and there is no security management function to disable it.

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

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

- secp256r1
- secp384r1
- secp521r1

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.11.1 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 supports TLS when exporting audit logs to an external server, and when communicating with authentication servers. The TOE is capable of providing a certificate in the TLS negotiation

when the TLS server requests a certificate. The evaluator confirmed that FIA_X509_EXT.2.1 indicated the TOE supports certificate authentication for TLS.

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

The section entitled "Mutual Authentication" in [CC-Guide] explains that for OmniSwitch to be able to use mutually authenticated TLS communication with an external syslog server, it is necessary for the OmniSwitch to obtain a Mutual Authentication Client certificate to identify itself to the syslog server. This section also explain that the OmniSwitch Mutual Authentication Client certificate needs to be with a server during the TLS negotiation. Therefore, the OmniSwitch must have a certificate to share. The certificate can be obtained either via generation of the keys and CSR within the TOE, or generation of the keys and certificate by an external Certificate Authority. This section and its subsections provide instructions to perform these two actions.

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

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

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

The evaluator configured the Test server used during testing of FCS_TLSC_EXT.1 such that the server sent a Certificate Request message. The evaluator observed that the TOE responded to these requests by sending the configured Certificate to the server. The evaluator observed that the TOE sent Certificate (type 11) and client Certificate Verify (type 15) messages during its negotiation of a TLS channel and that Application Data is sent.

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 6.3.1 of [ST] indicates that the TOE provides authentication failure settings that include a 'lockout window', a 'lockout threshold' and a 'lockout duration'. The 'lockout window' is the length of time a failed user login attempt is aged before it is no longer counted as a failed user login attempt (default is to count all attempts). The 'lockout threshold' is the number of failed user login attempts allowed within a given lockout window. The 'lockout duration' is the length of time a user account remains locked out until it is automatically unlocked.

The TOE prevents the establishment of a remote session after a defined number of unsuccessful attempts (lockout threshold) using the password-based authentication method.

An account is unlocked when the lockout duration expires or when the account is manually unlocked by another administrator.

Section 6.3 states that the pre-defined admin account can login at either the console or via SSH and is subject to be locked due to exceeding the lockout threshold through the SSH interface. However, the admin user account cannot be locked due to failing authentication attempts on the local console. The admin user account can login via the console even if the account is locked out at the SSH interface.

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 [Authentication Failure Management (FIA_AFL.1)] in [CC-Guide] describes that the TOE will detect and track any unsuccessful login attempts. This section describes three commands to configure the authentication failure handling mechanism.

The duration for which the failed login attempts must be counted can be configured by "***user lockout-window*** <***minutes***> " command.

The number of failed login attempts for the configured user lockout window can also be set by "***user lockout-threshold*** <***number***>" command.

The duration for which the unsuccessful login attempt user must be locked can also be set by "***user lockout-duration*** <***minutes***>" command.

This section also notes that while a value of zero (0) may be configured for these values, this disables the functionality. Since these functions are required functionality in the evaluated configuration, the administrator must NOT configure zero (0) for the lockout-window, lockout-threshold or lockout-duration.

This section explains that the default "admin" account is considered the privileged administrator and has full administrative privileges for all commands on the TOE. It is strongly recommended that the primary admin user (i.e., admin) have access to the local console as this user is the only admin user able to login locally if otherwise locked out.

Thus, the "admin" account is expected to be available only on the local console and not via SSH, thus ensuring that this admin account may always connect through the local console.

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

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

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

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

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

Test 1: The evaluator configured the number of unsuccessful authentication attempts to three, and then performed three login attempts into the "admin" user account with an incorrect password followed by a login attempt with the correct password, and verified that the login failed.

Test 2: The TOE supports unlocking accounts by administrator action as well as by use of a lockout time period so both methods were tested.

For unlocking based on lockout duration, 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 account, showing it 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 using good credentials was successful.

For unlocking based on explicit administrator action, 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 account, showing it had been locked. The evaluator logged in to another administrator account and issued a command to unlock the first account. The evaluator then attempted to login to the first account using good credentials and found that the TOE allowed the login because the account had been unlocked.

2.3.2 Password Management (NDcPP22e:FIA_PMG_EXT.1)

2.3.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 of the ST states that the TOE allows the configuration of a password policy that includes a minimum length, and the combination of upper and lower case, numbers and special characters. The set of supported special characters is: '@', '#', '$', '%', '^', '&', '*', '(', ')', '~', '[', ']', ':', ';', '|', '_', '/', '.', '<', and '>'.

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 "Password Management (Extended - FIA_PMG_EXT)" in [CC-Guide] identifies the combination of characters that can be used in a password. This section states that the administrator can define the password settings for the composition of strong password by configuring the global password policy settings in the TOE.

This section states that password can be composed of any combination of upper and lower case letters, numbers and a set of special characters. The evaluator confirmed the set in the AGD matched the set in the Security Target requirement FIA_PMG_EXT.1.

The is section also indicates that in CC enabled mode minimum password length is settable by the Security Administrator, and that the minimum password length shall be configurable between 1 (the minimum password length in default switch operation) and 30 (the maximum password length in default switch operation).

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 different passwords. 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 that was accepted as valid. The evaluator also confirmed that with a minimum password length of 8 configured, that a minimum length of 8 was required. This was confirmed by

attempting to set passwords with 7 characters and observing the TOE reject the password; as well as by attempting to set a password of 8 characters and observing that the TOE accepted the password change.

2.3.3 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.3.3.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

The section entitled "Protected authentication feedback (FIA_UAU.7)" in [CC-Guide] describes that the TOE provides only obscured feedback to the administrative user while the authentication is in progress at the local console. When the user provides the password, the password is not displayed on the local console until the authentication is successful or failed.

Component Testing Assurance Activities: The evaluator shall perform the following test for each method of local login allowed:

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

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

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

2.3.4.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See FIA_UIA_EXT.1.

Component Guidance Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

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

Component Testing Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See FIA_UIA_EXT.1

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

2.3.5.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the TSS shall describe how the overall TOE

functionality is split between TOE components including how it is ensured that no unauthorized access to any TOE component can occur.

For distributed TOEs, the evaluator shall examine the TSS to determine that it describes for each TOE component which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration. For each TOE component that does not support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2 the TSS shall describe any unauthenticated services/services that are supported by the component.

Section 6.3 of [ST] states that the TOE provides local and remote access to administrators; local access is provided through the serial console (connected to the available ports), whereas remote access is provided through the SSHv2 protocol using an SSH client. The TOE requires the administrator to identify and authenticate to the TOE prior to accessing any of the management functionality, regardless of the mechanism being used to interface with the TOE (e.g. serial console, SSH). The TOE can perform identification and authentication locally using its local database. A successful logon occurs when the user presents to the TOE a valid administrative user name along with either the correct password or verification of a public-key.

The TOE is not distributed.

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

The section entitled "Enable CC Mode" in [CC-Guide] states that the Common Criteria mode will only allow console and SSH access to the TOE. In Common Criteria mode, TOE can be accessed locally through serial console and remotely through SSH. The evaluator further ensured that this section included the commands to deploy the TOE in the CC enabled mode. The SSH communication supports both password based and public-key based authentication [missing section for the upload/install of ssh pub key on the TOE].

The section entitled "Identification and Authentication (FIA)" explains that the TOE factory default configuration enables only the default network administration user over the CLI interface on the local console. All other management interfaces that require authentication must be unlocked with the "***aaa authentication {console*** | ***telnet*** | ***ftp*** | ***http*** | ***snmp*** | ***ssh*** | ***default} server1 [server2...] [local]***" CLI command.

In addition, the section entitled "User Identification and Authentication (Extended - FIA_UIA_EXT)" describes that the TOE ensures each administrative user is successfully identified and authenticated before allowing any administrative actions on behalf of that user. This is supported by identifying/authenticating each user through local user database. This section also states that for successful authentication, the user must be configured on the

local user database using CLI "***user username username {password password | password-prompt}***" command" prior to authentication is attempted.

The section entitled "User authentication (Extended - FIA_UAU_EXT)" provides detailed information on the login process for both local console and remote access.

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

Test 2 - Using each interface the evaluator found that no functions were available to the administrator accessing the remote CLI prior to successful login. This matched the selection in the SFR.

Test 3 - Using each interface the evaluator found that no functions were available to the administrator accessing the local console prior to successful login. This matched the selection in the SFR.

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

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

2.3.6.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

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

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

e) Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

f) Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

g) Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

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

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

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

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

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

(TD0527 12/2020 update applied)

The TOE validates certificates as part of the TLS Session establishment with a syslog server. 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 and that the revocation checking was performed. The evaluator caused the test server to return revoked certificates in the chain (individually) and attempted the same connection from the TOE. Attempts using revoked certificates were not successful and revocation checking was performed. This test was performed with revocation checking via OCSP.

Test 4 OCSP-- The evaluator configured an OCSP responder to present a certificate that does not have the OCSP signing purpose. The evaluator established a 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.

2.3.6.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation). For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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

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

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

Test 1: The evaluator configured a test syslog server 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 6.3.2 of [ST] states that when acting as a TLS client, the TOE parses and validates the TLS server certificate. Certificate validation and certificate path validation consist of the following steps.

1. Verify that the certificate has a correct format and has not expired
2. Verify that the chain of trust from the certificate up to the CA root certificate is maintained
3. Verify that the basicConstraints extension exists and the CA flag is set to TRUE for all CA certificates in the path
4. Verify that the CA root certificate is trusted
5. Verify that the certificate has not been revoked
6. Verify that the extendedKeyUsage field in the certificate corresponds to the use of the certificate ("Client Authentication", "Server Authentication", "OCSP Signing" purpose)

If all these steps are successful, then the certificate is considered valid. If any of these steps fails, then the certificate is considered invalid.

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 Validation" in [CC-Guide] discusses X.509 certificate validation. This section explains that the validation of a certificate involves checking many fields within the certificate. All such checks

must be successful in order for the certificate to be considered valid. OmniSwitch performs the following checks on a certificate to determine its validity.

- Verify that the certificate has a correct format and has not expired
- Verify that the chain of trust from the certificate up to the CA root certificate is maintained
- Verify that the CA root certificate is trusted
- Verify that the basicConstraints extension exists and the CA flag is set to TRUE for all CA certificates in the path
- Verify that the certificate has not been revoked
- Verify that the extendedKeyUsage field in the certificate corresponds to the use of the certificate "Server Authentication" for a certificate from a syslog server connection
- Verify the subject name of the configured server matches the Subject name within the X.509 certificate presented by the server during TLS negotiations.

These checks are performed when the administrator manually uses the CLI command: ***aaa certificate verify*** described in the following section. These checks are also performed when the OmniSwitch establishes a TLS session with a configured syslog server. The validation of the “Subject” name occurs during TLS session establishment.

This section explains that the OmniSwitch uses the OCSP protocol as the primary method for checking the revocation status of the certificates. OmniSwitch TLS client applications obtain the OCSP Server URLs from within the certificates presented during TLS negotiation and verification the revocation status of those certificates by contacting the OCSP responder identified by the certificate AIA information. No configuration is necessary to ensure this behavior.

The OmniSwitch will treat an OCSP response indicating stauts is Unknown the same as a status of Revoked. Only when OmniSwitch can positively verify a certificate’s revocation status with its OCSP Responder is the certificate treated as valid.

Component Testing Assurance Activities: None Defined

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

2.3.7.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.7.2 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 6.3.2 of [ST] states that when acting as a TLS client, the TOE parses and validates the TLS server certificate. If mutual authentication is required, the TOE sends the certificate used for that purpose that is located at the certificate directory (/flash/switch/cert.d). The TOE obtains the revocation status of the certificate by validating the certificate using the OCSP protocol. If the OCSP responder is not reachable, then the validation fails and the certificate is assumed to be revoked.

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 Validation" indicates that the OmniSwitch validates X.509 certificate when they are imported into the TOE as well as when they are received during TLS session negotiation. This section explains that the validation of a certificate involves checking many fields within the certificate. All such checks must be successful in order for the certificate to be considered valid. This section includes a list of the verification steps performed while validating a certificate. It also indicates that OmniSwitch uses the OCSP protocol as the primary method for checking the revocation status of the certificates. This section states that OmniSwitch will treat an OCSP response indicating stauts is Unknown the same as a status of Revoked. Only when OmniSwitch can positively verify a certificate's revocation status with its OCSP Responder is the certificate treated as 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 TLS audit server. For this channel, the TOE was an initiator of the connection from the TOE the audit server protected by TLS. The evaluator demonstrated that when the revocation server for the certificate presented by the remote test server was available, the TOE was successful in establishing the TLS session.

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

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

2.3.8.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 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.

The NDcPP22e:FIA_X509_EXT.3 requirement in [ST] does not include the selection for '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 "Generate a certificate signing request" in [CC-Guide] contains instructions to generate a X.509 CSR, a network administrator must specify the “Subject” to be used by the certificate. This “Subject” is composed of several possible values which include the following information:

Common name

Organization Name
Organization Unit
Locality
State
Country

Subsections describe how to generate RSA and ECDSA keys to be used with the CSR, as well as providing sample commands to create the CRS.

The subsection "Import and Verify the certificate" section also provides an example of the CLI command to generate a CSR and to validate a certificate as part of a manual import process.

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

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

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

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

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

2.4 Security management (FMT)

2.4.1 Management of security functions behaviour (ndcPP22e:FMT_MOF.1/ManualUpdate)

2.4.1.1 ndcPP22e:FMT_MOF.1.1/ManualUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs it is required to verify the TSS to ensure that it describes how every function related to security management is realized for every TOE component and shared between different TOE components. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

There are no specific requirements for non-distributed TOEs.

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

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

For distributed TOEs the guidance documentation shall describe all steps how to update all TOE components. This shall contain description of the order in which components need to be updated if the order is relevant to the update process. The guidance documentation shall also provide warnings regarding functions of TOE components and the overall TOE that may cease to operate during the update (if applicable).

The section entitled "Download AOS 8.9 R11" in [CC-Guide] explains that the Common Criteria Certified TOE firmware/software can be downloaded from the TOE vendor website. The evaluator ensured that the same section includes the instructions on how to download the TOE firmware/software.

The section entitled "Trusted Update (FPT_TUD_EXT)" in [CC-Guide] explains that the TOE provides the ability to manually initiate and authenticate firmware/software updates to the TOE using a published hash prior to installing those updates. The evaluator ensured that the same section includes the commands to display the currently running TOE firmware/software version and the steps necessary to manually perform a TOE software update.

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 TOE restricts the ability to perform manual updates to Security Administrators with prior authentication. This has been tested in FIA_UIA_EXT.1 where the evaluator showed that the TOE does not offer any services through the administrative interfaces prior to authenticating the connecting user other than the display of a TOE warning banner. See Test Case FPT_TUD_EXT.1 for results of a successful update with prior authentication.

2.4.2 MANAGEMENT OF TSF DATA (NDCPP22E:FMT_MTD.1/COREDATA)

2.4.2.1 NDCPP22E:FMT_MTD.1.1/COREDATA

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 of [ST] states that no functions are available to any user before being identified and authenticated. Only the authorized administrator can configure TSF-related functions.

Since only authorized administrators have access to the CLI and all security management operations are available only via the CLI, the ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors is limited to authorized administrators.

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 section entitled "Management of TSF Data (FMT_MTD)" in [CC-Guide] explains that the ability to update/manage switch data is permitted only for administrator or any user having the desired privileges. Administrator privilege can be controlled by modifying privileges (RW/RO) assigned to a user for specific families/domains using the CLI command ***user*** <***username***> ***read-write***.

This section also states that the ability to delete, modify, import and generate cryptographic keys is also permitted only for admin user or any user having the desired privilege.

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

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

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

2.4.3.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

The section entitled "Management of TSF Data (FMT_MTD)" in [CC-Guide] explains that the ability to update/manage switch data is permitted only for administrator or any user having the desired privileges. Administrator privilege can be controlled by modifying privileges (RW/RO) assigned to a user for specific families/domains using the CLI command ***user <username> read-write***.

This section also states that the ability to delete, modify, import and generate cryptographic keys is also permitted only for admin user or any user having the desired privilege.

This section lists the keys the administrator is able to manage in a table "List of keys managed by administrator". This table indicate that that TOE allows the administrator to manage (delete, modify, import and generate) SSH host keys, user public keys for SSH authentication and the key pair associated with the TOE's identity certificate.

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 generate a cryptographic key before being authenticated as an administrator. The attempt was unsuccessful. The evaluator logged into the TOE as an authorized administrator and attempted to generate a cryptographic key and observed that the generation 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] indicates that security management functions can be performed through the Command Line Interface (CLI). The CLI can be accessed from the serial console or through a SSHv2 client.

Section 6.4 of [ST] enumerates the administrative functions that the TOE allows the administrator to perform. The following is the list provided which correspond to those required by FMT_SMF.1.

- Login at the local console or via SSH
- Configure the access banner for local and remote login sessions
- Configure session inactivity time for login sessions
- Install TOE firmware updates with the capability of verifying the integrity of those updates
- Configure authenticate failure parameters for login sessions (e.g. unsuccessful authentication attempts)
- Configure user login attempt lockout settings
- Configure audit behavior
- Manage cryptographic keys
- Configure cryptographic functionality
- Re-enable an Administrator account
- Set the date and time
- Manage the trust store and X.509v3 certificates (designate certificates as trust anchors, import X.509v3 certificates)

These functions were observed by the evaluator during testing.

Component Guidance Assurance Activities: See TSS Assurance Activities

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

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

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

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

2.4.5.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.5.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.5.3 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 6.5 of [ST] states that the TOE maintains administrative user roles which allow administrative accounts to have read-only or read-write permission to command families. Any administrative user that can login to the TOE, regardless of permission, is considered a security administrator.

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

The section entitled "Security management roles (FMT_SMR)" in [CC-Guide] states that the administrator must maintain roles, associate users with roles and be able to access the TOE locally and remotely. The TOE has an admin user by default. The admin user has all the privileges to manage the switch both remotely and locally. The admin user can create different users in the switch and specific privileges can be assigned to the users (R/W for specific modules) using the ***user*** <***username***> ***[read-only*** | ***read-write [families...*** | ***domains...***| ***all*** | ***none*** | ***all-except [families*** | ***domains....]*** command. This section further explains that the TOE can be administered locally and remotely. The section also provides commands for configuring the access for console and ssh authentication.

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 console timeout values, authentication, and self-tests were tested using local console.

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] states that the TOE stores the hashed value of the password in a directory of the flash filesystem that is protected from read/access from any user, including administrators.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.2.1 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 that the TOE stores key material in directories of the flash filesystem which are protected from read/access from any user, including administrators. The TOE does not offer any functions that will disclose to any users a stored cryptographic key.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.3.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.3.2 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] states that the TOE has an internal system clock which is used to generate reliable timestamps for security related purposes like auditing and certificate validation. This system clock can be updated by the security administrator through the CLI.

The TOE does not rely upon a virtualization system.

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

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

The section entitled "Time stamps (FPT_STM_EXT)" in [CC-Guide] explains how an administrator can set the date, the time and the time zone on the TOE.

The TOE does not support NTP and does not rely upon an underlying VS.

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

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

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

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

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

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

Test 2: The TOE does support the use of NTP to set time. The NTP capabilities were tested as part of FCS_NTP_EXT.1 testing.

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

2.5.4 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.5.4.1 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] explains that the TOE performs self-tests during start-up and conditional tests, which ensures the integrity of the cryptographic module and the correct operation of the cryptographic algorithms.

Section 6.2 of [ST} explains that OpenSSL performs the following power-up self-tests to ensure that the module and all validated cryptographic algorithms work properly.

- Integrity verification of the shared libraries that comprise the cryptographic module
- Known Answer Test (KAT) for symmetric encryption and decryption algorithms
- KAT for the DRBG
- KAT for MAC and message digest algorithms
- KAT for RSA signature generation and verification algorithms

- KAT for the Elliptic Curve Diffie-Hellman algorithm
- Pair-wise Consistency Tests (PCT) for ECDSA asymmetric algorithms (consisting of performing signature generation and verification for a known ECDSA key.

OpenSSL also performs the following conditional tests during the execution of services:

- PCT on each generation of an RSA key pair, consisting of performing signature generation and verification of a predefined message using the generated RSA key pair, as well as public key encryption and private key decryption of a predefined message using the generated RSA key pair.
- PCT on each generation of an ECDSA key pair, consisting of performing signature generation and verification of a predefined message using the generated ECDSA key pair.

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 "TSF testing (Extended - FPT_TST_EXT)" in [CC-Guide] describes that in the event of a power-on self-test failure, the cryptographic application will force the TOE to reload and reinitialize the operating system and cryptographic application. This operation ensures no cryptographic algorithms can be accessed unless all power on self-tests are successful. In addition to this, when it fails the self-test, relevant error messages are displayed on the console.

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 caused the TOE to reboot. The evaluator confirmed that software/firmware integrity tests and cryptographic algorithm tests were performed before the TOE reboot completed.

2.5.5 TRUSTED UPDATE (NDCPP22E:FPT_TUD_EXT.1)

2.5.5.1 NDCPP22E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.2 NDCPP22E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.3 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] states that the TOE allows administrators to verify the executing version and the most recently installed version of the TOE software. It also allows manual updates of the TOE software, verifying its trust and integrity using a published hash before being installed. When the TOE firmware is reloaded (reload from command), if the integrity of the TOE image is verified successfully, the command can proceed and the new version of the TOE is installed. If the integrity verification fails, then the TOE image is rejected and the command does not proceed with the update.

The TOE does not provide a means for installing a trusted update of the TOE with a delayed activation. However, the administrator must reboot the TOE in order to make the changes effective.

The TOE 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 "Trusted Update (FPT_TUD_EXT)" in [CC-Guide] states that the OmniSwitch provides the ability to query the currently executing version of the TOE firmware/software as well as the most recently installed version of the TOE firmware/software. This is provided using the various show commands ***show microcode [working | certified | loaded | issu | image_dir]*** that display the firmware/software version running on the system.

The section entitled "Trusted Update (FPT_TUD_EXT)" explains that the administrator must manually initiate and authenticate firmware/software updates to the TOE using a published hash prior to installing those updates. The software images along with hash file (**imgsha256sum**) needs to be transferred to OmniSwitch securely using SSH for firmware or software updates. The section entitled "Download AOS 8.9 R11" indicates how the Security Administrator can obtain authentic firmware/software and published hash values for the updates.

This section further explains that after downloading the hash file (imgsha256sum) the administrator needs to compare the hash value of the file against the hash value published on the support website https://myportal.al-enterprise.com/s. And then use the ***image integrity check <image_dir> key-file <filename>*** command to check the integrity of the image files in working or certified directory.

The above command verifies the integrity of the images against the published hash. If the verification fails the administrator must remove the invalid update from the system. If the verification indicates Key matched the update will be applied.

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

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

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

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

2) An image that has not been signed

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

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

c) Test 3 [conditional]: If the TOE itself verifies a hash value over an image against a published hash value (i.e. reference value) that has been imported to the TOE from outside such that the TOE itself authorizes the installation of an image to update the TOE, the following test shall be performed (otherwise the test shall be omitted). If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall

perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

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

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

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

If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 3 shall be skipped.

The evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all methods supported (manual updates, automatic checking for updates, automatic updates).

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

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

Test 2: The TOE does not verify a digital signature to authorize the installation, so this test is not applicable.

Test 3: The TOE verifies updates using a published hash. The evaluator used a valid update to create two invalid updates. The evaluator then attempted to install each of these invalid updates:

1) An illegitimate update such that the hash of the update does not match the published hash; and
2) An legitimate update without a published hash.

For each image the evaluator showed the current TOE version that was installed, attempted the reload command to install the invalid image, then finally showed the TOE version after the update attempt. All attempts to apply invalid updates failed to install and the TOE remained at the original running version. The TOE does not support delayed activation, so that aspect of the Test Assurance activity is not applicable.

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] explains that the TOE terminates remote sessions after a configurable period of inactivity. After termination, administrative authentication is required to access any of the administrative functionality of the TOE.

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

The section entitled "TSF-initiated Session Locking and Termination (Extended - FTA_SSL_EXT, FTA_SSL)" in [CC-Guide] provides the command to configure inactivity time period for local administrative session termination. The TOE locks local and remote interactive sessions by terminating the session after configurable time interval of session inactivity. This is achieved using the ***session cli timeout <num>*** command. The active session gets terminated after the configured timeout interval is reached.

This section also states that by default the inactivity time for TOE is set at 55 seconds.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following test:

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

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

2.6.2 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 that the TOE provides the function to logout (or terminate) both local and remote user sessions as directed by the user.

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

The section entitled "TSF-initiated Session Locking and Termination (Extended - FTA_SSL_EXT, FTA_SSL)" in [CC-Guide] provides the command to terminate the user's own interactive session on either the local console or a remote SSH connection.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following tests:

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

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

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

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

2.6.3 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 that the TOE terminates local sessions after a configurable period of inactivity. After termination, administrative authentication is required to access any of the administrative functionality of the TOE.

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

The section entitled "TSF-initiated Session Locking and Termination (Extended - FTA_SSL_EXT, FTA_SSL)" in [CC-Guide] provides the command to configure inactivity time period for local administrative session termination. The TOE locks local and remote interactive sessions by terminating the session after configurable time interval of session inactivity. This is achieved using the ***session cli timeout <num>*** command. The active session gets terminated after the configured timeout interval is reached.

This section also states that by default the inactivity time for TOE is set at 55 seconds.

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

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that reauthentication is needed when trying to unlock the session.

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

2.6.4 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 6.6 of [ST] states that the administrator can access the TSF via the local console (serial) or remotely via SSH. The TOE allows the configuration of a banner shown to the user before an interactive session is established at the local console or remotely via SSH.

The evaluator confirmed that the administrative methods described in this section match those available on the TOE.

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 "TOE access banners (FTA_TAB)" in [CC-Guide] provides the commands that allows an administrator to configure the banner message that is displayed on the CLI prior to logging in and after login.

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] states that remote connections to third-party syslog servers are supported for exporting audit records to an external audit server. Administrators can configure communication with an external audit server to be protected using TLS. The TOE is a TLS client when exporting audit records to a third party syslog server.

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 "Trusted Channel (FTP_ITC.1)" in [CC-Guide] explains that the TOE uses TLS as a trusted communication channel between itself and the audit server. This section explains that for a successful TLS connection to be established, the TOE must be configured in common-criteria (CC) mode. The evaluator further ensured that the section entitled "Enable CC Mode" in [CC-Guide] included the commands to configure the TOE to enable CC mode.

This section also references other sections of the [CC-Guide] where instructions are present to generate and import certificates used by the TLS connection. This include CA bundle and the TOE certificate.

The section entitled "Trusted Channel (FTP_ITC.1)" in [CC-Guide] states that a TLS connection will be established only upon successful certificate validation. This x509 certificate validation involves CA certificates as well. The section entitled "Authentication using X.509 Certificates (Extended - FIA_X509_EXT)" in [CC-Guide] explains that during TLS handshake, the external Syslog servers will send their certificate to the OmniSwitch client.

The section entitled "Trusted Channel (FTP_ITC.1)" in [CC-Guide] describes that when the TLS connection is broken, the syslog client application will try to re-establish the connection with the external server at regular time intervals. Audit data generated during the time when connection is down will be stored only locally and not sent to the audit server.

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

The evaluator shall perform the following tests:

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

b) Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

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

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

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a

duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

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

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

Further assurance activities are associated with the specific protocols.

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

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

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

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

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

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

2.7.2.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.3 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 that the TOE provides a trusted path for its remote administrative users accessing the TOE using SSH. Note that local administrator access is also allowed for command line access. All communications initiated by remote administrators to the TOE are protected using a secure channel via the SSHv2 protocol.

The protocols described as in section 6.7 are consistent with those listed in the requirement.

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 "Enable CC Mode" in [CC-Guide] states that Common Criteria mode will only allow console and SSH access to the OmniSwitch. CCE requires each administrative user to be successfully identified and authenticated before allowing any other TSF mediated actions on behalf of that administrative user. In Common Criteria mode, TOE can be accessed locally through serial console and remotely through SSH. SSH communication supports both password based and public-key based authentication. This section also includes the command to enable common-criteria mode which enables all CC functions.

In addition, the section entitled "User Identification and Authentication (Extended - FIA_UIA_EXT)" describes that the TOE ensures each administrative user is successfully identified and authenticated before allowing any administrative actions on behalf of that user. This is supported by identifying/authenticating each user through local user database. This section also states that for successful authentication, the user must be configured on the local user database prior to authentication is attempted.

The section entitled "User authentication (Extended - FIA_UAU_EXT)" in [CC-Guide] provides detailed information on the login process for both local console and remote access.

The section entitled "Trusted path (FTP_TRP.1)" provides the CLI operation that enables or disables the SSH connections to the TOE.

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

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

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

Further assurance activities are associated with the specific protocols.

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

The TOE offers remote administration via SSHv2 to provide the trusted path (with protection from disclosure and modification) for all remote administration sessions.

Test 1: The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE using SSH and observed that the connection was successful and did use SSHv2.

Test 2: The evaluator examined the packet capture and observed that the remote administration session was protected by SSH and did not contain plaintext data.

3. Protection Profile SAR Assurance Activities

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

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

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

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

The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

It should be noted that there may be some SFRs that do not have an interface that is explicitly 'mapped' to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

The Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the Supporting Document.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps (per TD0536):

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

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

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

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

As identified throughout this AAR, the [CC-Guide] provides instructions for configuring the TOE's cryptographic security functions. The [CC-Guide] provides instructions for configuring the cryptographic algorithms and parameters used for the evaluated configuration. The [CC-Guide] is clear that no other cryptographic configuration has been evaluated or tested. There are warnings and notes throughout the [CC-Guide] regarding use of functions that are and are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP22e:FPT_TUD_EXT.1.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

Preparative procedures are distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

In addition, the evaluator performs the EAs specified below.

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

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

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

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

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

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

The preparative procedures must

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

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

The evaluation team had the following documents to use when configuring the TOE:

- Preparation and Operation of Common Criteria Evaluated OmniSwitch Products (NDcPP), AOS Release 8.9.R11

The completeness of this documentation is addressed by its use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a 'hard label' (e.g., stamped into the metal, paper label) or a 'soft label' (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

See section entitled "Labelling of the TOE (ALC_CMC.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.

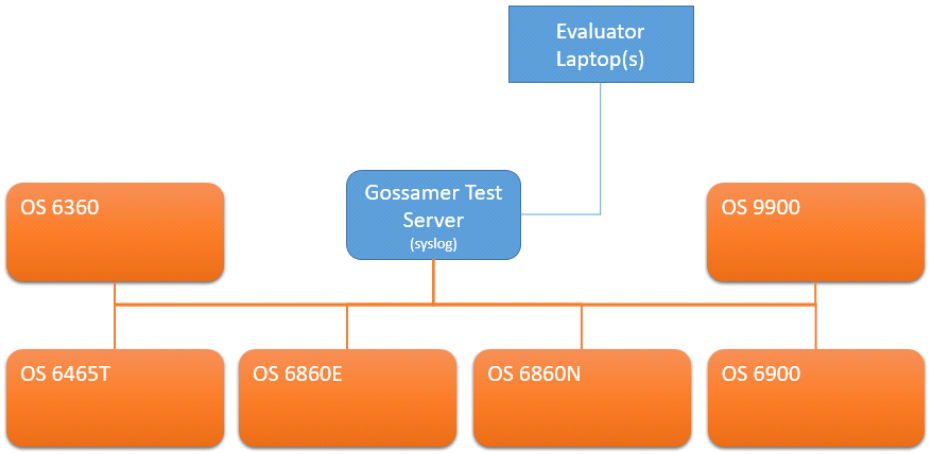

Figure 1 Network Setup

TOE Platforms:

- OS6360
- OS6465T
- OS6860E
- OS6860N
- OS6900
- OS9900

Supporting Products:

- None

Supporting Software:

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

- Windows 10.0
- Wireshark version 4.0.4
- Windows SSH Client - Putty version 0.67, 0.71, 0.73 & 0.74 (used to connect to device console and SSH)

The Gossamer Test servers with an Ubuntu environment acted as platforms to initiate testing. The test servers also acted as a syslog server.

- OpenSSL version 1.0.2g
- OpenSSH version 7.2p2
- Big Packet Putty, Openssh-client version 6.8p1,
- Rsyslog version 8.16.0
- NTPd version 4.2.8p4
- TCPdump version 4.9.3
- LibPCAP version 1.7.4
- NMAP version 7.01
- Stunnel version 5.30

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has

been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A, while an 'outline' of the assurance activity is provided below.

In addition to the activities specified by the CEM in accordance with Table 2, the evaluator shall perform the following activities.

The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

The developer shall provide documentation identifying the list of software and hardware components7 that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

If the TOE is a distributed TOE then the developer shall provide:

a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]

b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuzz testing. None of the public search for vulnerabilities, or 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/)
- cve.org CVE Database (https://www.cve.org/),
- SecurITeam Exploit Search (http://www.securiteam.com)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search)
- Offensive Security Exploit Database (https://www.exploit-db.com/)

The search was performed on October 9, 2023. The search was conducted with the following search terms: "Alcatel-Lucent", "OminSwitch", "SSH", "TLS", "ARM Cortex A9", "Intel Atom C3558", "Intel Atom C2338", “OpenSSL 3.0.7”. Intel Atom C2538”, “Intel Xeon D1518”, “Intel Atom C2518”, “Marvell 98DX236S”, “Marvell 98DX233S”, “Marvell 88F6820”, “Broadcom BCM56342”, “Broadcom BCM56340”.