www.GossamerSec.com

Assurance Activity Report for Extreme Networks ExtremeSwitching Series (X440-G2, X460-G2, X465, X435, X695) and 5520 Series Switches running EXOS 31.3.100

Version 1.0
10/25/2022

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

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

Revision History

Revision	Date	Authors	Summary
Version 0.1	10/06/2022	Sykes	Initial draft
Version 1.0	10/25/2022	Sykes	ECR comments

The TOE Evaluation was Sponsored by:

Extreme Networks, Inc.
6480 Via Del Oro
San Jose, CA 95119

Evaluation Personnel:

- Neal Haley
- Katie Sykes

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

LIST OF TABLES

1. Introduction

This document presents evaluations results of the Extreme Networks ExtremeSwitching Series (x440-G2, x460-G2, x465, x435, x695) and 5520 Series Switches running EXOS 31.3.100 NDcPP22e evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

This section explains why the test subset was adequate to address all product installations.

1.1.1 Evaluated Platform Equivalence

The TOE consists of the following appliance series all running EXOS software version 31.3.100.

- ExtremeSwitching Series x440-G2
- ExtremeSwitching Series x460-G2
- ExtremeSwitching Series x435
- ExtremeSwitching Series x465
- ExtremeSwitching Series x695
- 5520 Series

The evaluation team performed the entire suite of tests on each of the following devices all running EXOS 31.3.100:

- u5520 model: 5520-12MW-36W-EXOS
- x435 model: X435-8T-4S
- x440 model: X440G2-12p-10G4
- x465 model: X465-24S

The table below contains the list of specific appliance models included in the TOE. **Bold** rows with grey background represent devices specifically tested during the evaluation. The primary difference between models in the same series are differences in the number of network ports, amount of local storage and the amount of installed memory.

Model	CPU	CPU Mfg	CPU Type	Micro Architecture	CPU Cores
X440-G2-12t-10GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-12p-10GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-24t-10GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-24p-10GE4	CN7010	Marvell (formerly	MIPS	Octeon III	1

Model	CPU	CPU Mfg	CPU Type	Micro Architecture	CPU Cores
		Cavium)			
X440-G2-48t-10GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-48p-10GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-24t-10GE4-DC	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-48t-10GE4-DC	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-24x-10GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-24fx-GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-12t8fx-GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X440-G2-24t-GE4	CN7010	Marvell (formerly Cavium)	MIPS	Octeon III	1
X460-G2-24t-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-48t-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-24p-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-48p-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-24x-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-48x-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-24t-GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-48t-GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-24p-GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-48p-GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-16mp-32p-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-24p-24hp-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2
X460-G2-24ht-10GE4	CN6120	Marvell (formerly Cavium)	MIPS	Octeon II	2

Model	CPU	CPU Mfg	CPU Type	Micro Architecture	CPU Cores
X435-8T-4S	Broadcom BCM53549	ARM	Cortex -A	Cortex-A72 Armv7	4
X435-8P-4S	Broadcom BCM53549	ARM	Cortex -A	Cortex-A72 Armv7	4
X435-8P-2T-W	Broadcom BCM53548	ARM	Cortex -A	Cortex-A72 Armv7	4
X435-24T-4S	Broadcom BCM53547	ARM	Cortex -A	Cortex-A72 Armv7	4
X435-24P-4S	Broadcom BCM53547	ARM	Cortex -A	Cortex-A72 Armv7	4
X465-24W	C3338	Intel	Atom	Denverton	2
X465-48T	C3338	Intel	Atom	Denverton	2
X465-48P	C3338	Intel	Atom	Denverton	2
X465-48W	C3338	Intel	Atom	Denverton	2
X465i-48W	C3538	Intel	Atom	Denverton	4
X465-24MU	C3538	Intel	Atom	Denverton	4
X465-24MU-24W	C3538	Intel	Atom	Denverton	4
X465-24S	C3338	Intel	Atom	Denverton	2
X465-24XE	C3538	Intel	Atom	Denverton	4
X695-48Y-8C	C3758	Intel	Atom	Denverton	8
5520-24T	Broadcom BCM56377	ARM	Cortex -A	Cortex A72 ARMv8	4
5520-24W	Broadcom BCM56377	ARM	Cortex -A	Cortex A72 ARMv8	4
5520-48T	Broadcom BCM56376	ARM	Cortex -A	Cortex A72 ARMv8	4
5520-48W	Broadcom BCM56376	ARM	Cortex -A	Cortex A72 ARMv8	4
5520-12MW-36W	Broadcom BCM56375	ARM	Cortex -A	Cortex A72 ARMv8	4
5520-48SE	Broadcom BCM56376	ARM	Cortex -A	Cortex A72 ARMv8	4
5520-24X	Broadcom BCM56375	ARM	Cortex -A	Cortex A72 ARMv8	4

Table 1 TOE Models and Processors

The TOE security behavior is the same on all the switches in the same series for each of the SFRs because all security functions provided by the TOE are implemented in the single software image that runs on all models in that series. Thus, the evaluation team performed testing on one device of each image type. There exist FOUR

distinct images for the SIX Series of Extreme Switches identified above. All TOE devices in each of the SIX series, leverage the same software image running the same EXOS version as follows:

- One image for all x440-G2 and x460-G2 series devices
- One image for all x435 series devices
- One image for all x465 series and x695 devices
- One image for all 5520 series devices.

The x460-G2 Series is included based on equivalency with the x440 Series models (from which one model was tested). Although the CPU for the x460 is Octeon II while the CPU for the x440 series is Octeon III, both series share the same software image, same instruction set, same memory addressing scheme and same network interfaces. The Octeon II and Octeon III are both included in the Entropy report and in the CAVP certificates.

The x695 Series model is included based on equivalency with the x465 Series models (from which one model was tested). Both series share the same software image, same instruction set, same memory addressing scheme and the same network interfaces.

Devices within these series of switches that comprise the TOE have different hardware characteristics. These characteristics affect only non-TSF relevant functions of the switches (such as throughput, processing speed and amount of storage). While the CPUs differ across the various models, algorithm cert testing of the Extreme Networks FIPS Object Module was successfully performed on all series of CPUs. All cryptographic operations are addressed in the algorithm testing.

The test procedures were based on the available guidance and proved identical across all models tested. Similarly, the results proved to be identical in each case. This further substantiates that the underlying physical device did not have any bearing on the security claims.

1.1.2 CAVP Equivalence

The EXOS utilizes Extreme Networks FIPS Object Module, a firmware cryptographic module based on OpenSSL. Extreme Networks FIPS Object Module Version 2.0.16m is used in the Cavium Octeon (MIPS) platforms, while Extreme Networks FIPS Object Module Version 2.0.16i is used in the Intel and ARM platforms. Both versions of this module have the same algorithm code. This cryptographic module operates in FIPS mode and exclusively implements all cryptographic functionality.

All microarchitectures included in the evaluation have CAVP testing as required. The following table identifies the CAVP Certificates applicable to the cryptographic functionality included in the evaluation.

Requirement Component	Capabilities	Certificate #			
		Cavium	Cavium	Intel	Broadcom

		Octeon II (FOM v2.0.16m)	Octeon III (FOM v2.0.16m)	Atom C3338 (FOM v2.0.16i)	BCM 53549 BCM 56375 (FOM v2.0.16i)
FCS_CKM.1 Cryptographic Key Generation	RSA KeyGen Mod: 2048, 3072 DSA KeyGen Mod: 2048, 3072 ECDSA KeyGen P-256, P-384, P-512	C1154	A2782	C1156	A1391
FCS_CKM.2 Cryptographic Key Establishment	KAS FFC KAS ECC	A2782	A2782	A1391	A1391
FCS_COP.1/ DataEncryption Cryptographic Operation (AES Data Encryption / Decryption)	AES-CBC, CTR Key Length: 128, 256	C1154	A2782	C1156	A1391
FCS_COP.1/SigGen Cryptographic Operation (Signature Generation and Verification)	RSA SigGen (FIPS186-4) PKCS 1.5 Mod: 2048, 3072 RSA SigVer (FIPS186-4) PKCS 1.5 Mod: 2048, 3072	C1154 A2782	A2782	C1156 A1391	A1391
FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)	SHA-1, SHA2-256, SHA-512	C1154	A2782	C1156	A1391
FCS_COP.1/ KeyedHash Cryptographic Operation	HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA-512	C1154	A2782	C1156	A1391
FCS_RBG.1 Random Bit Generation	Counter DRBG Mode: AES-256	C1154	A2782	C1156	A1391

Table 2 CAVP Certificates

1.2 References

The following evidence was used to complete the Assurance Activities:

- Extreme Networks ExtremeSwitching Series (x440-G2, x460-G2, x465, x435, x695) and 5520 Series Switches running EXOS 31.3.100 Security Target, Version 1.0, 10/25/2022 (ST)
- Extreme Networks ExtremeXOS Common Criteria Configuration Guide 31.3.100, 9037401-00, Rev AA, October 2022 (**Admin Guide**)

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profiles and describes the findings in each case.

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 of the ST states that for cryptographic keys, the act of generating, importing, changing or deleting a key is audited by key name and the associated administrator account is identified.

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 “Audit Record Samples”, Table 1 in the **Admin Guide** provides actual audit records of each auditable event required by FAU_GEN.1

The "Audit Logs and Syslog" section in the **Admin Guide** describes the log filters that can be applied to all log targets (e.g. syslog) as well as the log record format. For each captured audit, the generated record contains the date and time, the type of event, the subject identity (for example, IP address or User name), the outcome, and the severity of the event.

From a review of the ST, the Guidance and through testing, the evaluator also determined that the guidance contains all of the administrative actions and their associated audit events that are relevant to the PP and to use of the TOE. These administrative actions are consistent with the security requirements implemented in the TOE and were found to have appropriate management capabilities identified in the guidance documentation.

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 each audit event when running the other security functional tests described in this AAR as required by the PP. The evaluator 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.

Please refer to NDcPP22e:FAU_GEN.1.

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

Please refer to NDcPP22e:FAU_GEN.1.

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

For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated for building up a secure channel between two TOE components. If for some reason (could be e.g. TSS or Guidance

Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

This activity was accomplished in conjunction with the testing of FAU_GEN.1.1.

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

2.1.3.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.3 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 the ST states the TOE is a standalone TOE that stores audit data locally and can also export all logs to an external audit server. An administrator-configurable number of log messages can be stored locally on the appliance. Approximately, 20KB for nvram and 200-20000 records for memory-buffer can be stored locally. All local audit records exist in a circular buffer, FIFO manner; when the buffer gets full, the oldest message is overwritten first. There is no access to audit data storage; the CLI allows displaying of logs but there is no access to log files. Only Security Administrators can view the audit records. In this way, the audit records are protected against unauthorized access and deletion. Clearing the local audit trail is done per target and it wipes all audit records for that target.

The TOE can be configured to export audit records to an external syslog server. This communication is protected by TLS. The transmission of audit logs to the external audit server is done in real time, with audit records transferred as they are generated. If the connection to the external audit server is lost, the TOE continues to save local audit logs so there is no loss of audit. There is automated log reconciliation process (syncing) between the locally stored records with the external audit server upon the re-establishment of the connection.

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 "Audit Logs and Syslog" section in the **Admin Guide** states that the transmission of audit logs to the external audit (syslog) server is done in real time, with each audit record transferred as it is generated. If the connection to the external audit server is lost, the EXOS device continues to save local audit logs so there is no loss of audit. An automated log reconciliation process (syncing) occurs between the locally stored records with the external audit server when the connection is reestablished. The stored audit records cannot be directly accessed, but can be viewed via the CLI.

The "Enable a TLS Connection to the Syslog Server" section in the **Admin Guide** EXOS communicates with an external syslog (audit) server by establishing a trusted channel between itself and the syslog server. Implementation of the trusted channel uses TLS v1.2 with server-side X.509v3 certificate-based authentication, in which the X.509v3 certificate that is presented by the syslog server is checked against the configured trusted CA (certificate authority) chain and then validated by OCSP.

The "Configure the Size of the Logging Buffer" section in the **Admin Guide** states that the size of the local buffer can be configured from 200 to 20000 log messages. When the local buffer gets full, the oldest message is overwritten first, a process commonly called FIFO (First In, First Out). For NVRAM, the storage limit is 20KB. When that limit is reached, the oldest messages are overwritten. This is consistent with the description of the local audit storage behavior identified in FAU_STG_EXT.1.3.

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.

The TOE has both volatile (memory-buffer) and non-volatile (nvram) local storage. All local audit records exist in a circular buffer, FIFO manner; when the buffer gets full, the oldest message is overwritten first. Audit records stored in the memory-buffer are subject to the FIFO rules, but will be wiped on reboot.

Test 1: The evaluator configured the TOE to send audit records via a secure TLS connection to an external server running rsyslog 8.16.0. The evaluator tested the TOE's ability to generate and transmit as a part of every test that includes associated audit messages. The evaluator observed via packet captures that the audit data was not sent in the clear. The evaluator also observed that all audit data was successfully received by the audit server.

Test 2 (1, 2, 3): The TOE has two local log storage locations: volatile (memory-buffer) and non-volatile (nvram). The evaluator generated audit data until the local storage space in each location was exceeded. The evaluator confirmed that in both cases, that when the local audit storage is filled to the maximum, the existing audit data is overwritten based on the following rule: overwrite oldest records first. This behavior complies with the behavior defined in FAU_STG_EXT.1.3.

Test 3: NA - the TOE does not claim FAU_STG_EXT.2/LocSpace.

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

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 the ST states that the TSF supports RSA key generation scheme using cryptographic key sizes of 2048 bit or greater that meet the FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3; standard. The TSF also supports ECDSA (appendix B.4), FFC key generation (appendix B.1) and FFC schemes using 'safe-prime' groups that meet NIST SP 800-56A Revision 3 and RFC 3526. The key pairs can be used to generate a Certificate Signing Request (CSR).

The TOE supports asymmetric key generation using RSA, ECDSA and FFC key establishment as part of TLS and SSH. Table 14 in the ST maps these key generation/establishment schemes to services that the schemes are used for which include remote administration via SSH and secure communications with an external syslog server via TLS. The TOE acts as a client for TLS (ECDSA, FFC) and a server for SSH (RSA, FFC). The TOE supports DH group 14 key establishment scheme that meets standard RFC 3526, section 3 for interoperability.

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

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a

cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Generate SSH Host Keys" in the **Admin Guide** provides instructions for generating an SSH2 host key. SSH2 host keys are used to authenticate connections between the device and clients on remote systems. A host key must be generated before the device can accept incoming SSH connections. Generation of a new key overwrites the previous key.

Section "User Key-Based Authentication" in the **Admin Guide** provides instructions for generating an SSH key pair for SSH user key-based authentication. Instructions include configuring the keys on the switch, associating the key with a user, as well as commands for disassociating the key and deleting it all together.

Section "X.509 Certificate-Based Authentication" in the **Admin Guide** provides instructions for generating a certificate signing request and key pair that can then be signed by a Certificate Authority (CA). The resulting CSR-signed certificate can then be imported to the TOE for use as the syslog client side certificate used for mutual authentication. Instructions include commands for downloading CA and CSR certificates to the trust store, as well as commands for removing these certificates from the trust store.

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 CAVP certificates identified in Section 1.1.2.

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 the ST indicates that in support of secure cryptographic protocols, the TOE supports key establishment schemes, including:

- RSA key establishment as specified in Section 7.2 of RFC 3447, Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1,

- ECC key establishment as specified in NIST SP 800-56A Revision 3, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography',
- FFC key establishment in as specified in NIST SP 800-56A Revision 2, 'Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography',
- FFC Schemes using 'safe-prime' groups that meet the following: NIST SP 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography and groups listed in RFC 3526.

The TOE supports asymmetric key generation using RSA, ECDSA and FFC key establishment as part of TLS and SSH. Table 14 in the ST maps these key generation/establishment schemes to services that the schemes are used for which include remote administration via SSH and secure communications with an external syslog server via TLS. The TOE acts as a client for TLS (ECDSA, FFC) and a server for SSH (RSA, FFC). The TOE supports DH group 14 key establishment scheme that meets standard RFC 3526, section 3 for interoperability.

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

This activity has been performed in FCS_CKM.1.

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 CAVP certificates identified in Section 1.1.2.

2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.3.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for2). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

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

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

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

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

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

Section 6.2 in the ST states that the TOE is designed to destroy Critical Security Parameters (CSPs) when no longer required for use to mitigate the possibility of disclosure. Volatile memory (RAM) is cleared with overwriting zeros. Non-volatile EEPROM, is destructed with overwrite consisting of zeros. For non-volatile flash memory, destruction is done by direct overwrite consisting of zeros. Table 15 in the ST describes each type of plaintext key material, its origin and storage location, and the method of zeroization. The evaluator found that the description of keys and storage locations is consistent with functions carried out by the TOE and that all relevant keys are accounted for.

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).

Section “Zeroization” in the **Admin Guide** describes each type of plaintext key material, its origin and storage location, and the method of zeroization including use of the “unconfigure switch all” command which resets the platform to factory defaults. This is consistent with the description in the TSS. The ST and the Admin Guide do not identify any circumstances or configurations that do not strictly conform to the key destruction requirements.

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 in the ST states that the TOE provides symmetric encryption and decryption capabilities using AES in CBC and CTR modes (128, 256 bits). AES is implemented in support of the following protocols: TLS, and SSH. Table 4, in the ST identifies the TOE's cryptographic functions including cryptographic algorithms, modes and key sizes.

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.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Supported TLS Ciphers and Curves" in the **Admin Guide** specifies the TLS ciphers and curves supported in the evaluated configuration when the switch is configured in FIPS mode.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

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 CAVP certificates identified in Section 1.1.2.

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 in the ST states that the TOE supports hashing using SHA-1, SHA-256, and SHA-512 validated conforming to ISO/IEC 10118-3:2004. SHS hashing is used within several services including, NTP hashing, TLS, and SSH. SHA-256 is used in conjunction with RSA signatures 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.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured. The supported set of ciphers includes the use of HMAC-SHA1, HMAC-SHA2-256 and HMAC-SHA2-512.

Section "Supported TLS Ciphers and Curves" in the **Admin Guide** specifies the TLS ciphers and curves supported in the evaluated configuration when the switch is configured in FIPS mode. The supported set of TLS ciphersuites includes the use of SHA256 and SHA384.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

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 in the ST states that the TOE supports keyed hash HMAC-SHA1, HMAC-SHA256, and HMAC-SHA512 conforming to ISO/IEC 9797-2:2011 with cryptographic key sizes: 160, 256, 512 bits, message digest sizes: 160,

256, 512 bits and a 160/256/512 output MAC. The SHA-1/256 and 512 algorithms have block sizes of 512 and 1024 bits respectively.

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.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Supported TLS Ciphers and Curves" in the **Admin Guide** specifies the TLS ciphers and curves supported in the evaluated configuration when the switch is configured in FIPS mode. The supported set of TLS ciphersuites includes the use of SHA256 and SHA384.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

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.

Section 6.2 in the ST states that the TOE supports RSA signature generation and verification according to RSASSA-PKCS1v1_5 with 2048-bit and 3072-bit key sizes utilizing SHA-1, SHA-256, SHA-512.

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.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured. The supported set of ciphers includes the use of HMAC-SHA1, HMAC-SHA2-256 and HMAC-SHA2-512.

Section "Supported TLS Ciphers and Curves" in the **Admin Guide** specifies the TLS ciphers and curves supported in the evaluated configuration when the switch is configured in FIPS mode.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key 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 CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.8 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1)

2.2.8.1 NDcPP22e:FCS_NTP_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS to ensure it identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained. The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp.

Section 6.2 in the ST states that the TOE implements the NTPv3 protocol to synchronize with an external time server. The TOE authenticates updates using an administrator-configured SHA256-based message authentication. The TOE does not synchronize based on broadcast and multicast time updates. The TOE supports configuration of multiple simultaneous time servers and follows RFC 5905 algorithm to prioritize them.

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the version of NTP supported, how to configure multiple NTP servers for the TOE's time source and how to configure the TOE to use the method(s) that are selected in the ST.

Section "Network Time Protocol" in the **Admin Guide** provides instructions for configuring the TOE to use NTP including how to add and delete NTP servers and how to enable ntp authentication and create a SHA-256 key. The TOE supports NTPv3 and there is no further configuration needed.

Testing Assurance Activities: The version of NTP selected in element 1.1 and specified in the ST shall be verified by observing establishment of a connection to an external NTP server known to be using the specified version(s) of NTP. This may be combined with tests of other aspects of FCS_NTP_EXT.1 as described below.

The evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server and confirmed via packet capture that the TOE establishes a connection to the external NTP server using NTPv3 as claimed in the ST.

2.2.8.2 NDcPP22e:FCS_NTP_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: For each of the secondary selections made in the ST, the evaluator shall examine the guidance document to ensure it instructs the Security Administrator how to configure the TOE to use the algorithms that support the authenticity of the timestamp and/or how to configure the TOE to use the protocols that ensure the integrity of the timestamp.

Assurance Activity Note:

Each primary selection in the SFR contains selections that specify a cryptographic algorithm or cryptographic protocol. For each of these secondary selections made in the ST, the evaluator shall examine the guidance documentation to ensure that the documentation instructs the administrator how to configure the TOE to use the chosen option(s).

Section "Manage NTP Authentication" in the **Admin Guide** indicates that when FIPS mode is enabled, NTP uses the OpenSSL FIPS library and supports only SHA-256 for authentication. This section further provides instructions for enabling ntp authentication and creating a SHA-256 key.

Testing Assurance Activities: The cryptographic algorithms selected in element 1.2 and specified in the ST will have been specified in an FCS_COP SFR and tested in the accompanying Evaluation Activity for that SFR. Likewise, the cryptographic protocol selected in in element 1.2 and specified in the ST will have been specified in an FCS SFR and tested in the accompanying Evaluation Activity for that SFR.

[Conditional] If the message digest algorithm is claimed in element 1.2, the evaluator will change the message digest algorithm used by the NTP server in such a way that the new value does not match the configuration on the TOE and confirms that the TOE does not synchronize to this time source.

The evaluator shall use a packet sniffer to capture the network traffic between the TOE and the NTP server. The evaluator uses the captured network traffic, to verify the NTP version, to observe time change of the TOE and uses the TOE's audit log to determine that the TOE accepted the NTP server's timestamp update.

The captured traffic is also used to verify that the appropriate message digest algorithm was used to authenticate the time source and/or the appropriate protocol was used to ensure integrity of the timestamp that was transmitted in the NTP packets.

The evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server in NDcPP22e:FCS_NTP_EXT.1.1-t1 using the correct authentication algorithm. For this test, the evaluator changed the NTP server configuration to demonstrate that the TOE would not synchronize if the wrong message digest algorithm was used. The evaluator observed that the TOE rejected the connection and there was no time synchronization between the TOE and the NTP server.

2.2.8.3 NDcPP22e:FCS_NTP_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the TOE to not accept broadcast and multicast NTP packets that would result in the timestamp being updated.

Section "Disable the NTP Broadcast Client" in the **Admin Guide** provides instructions for disabling the NTP broadcast client which listens for NTP packets from an NTP broadcast server.

Testing Assurance Activities: The evaluator shall configure NTP server(s) to support periodic time updates to broadcast and multicast addresses. The evaluator shall confirm the TOE is configured to not accept broadcast and multicast NTP packets that would result in the timestamp being updated. The evaluator shall check that the time stamp is not updated after receipt of the broadcast and multicast packets.

The evaluator configured the TOE to get NTP time updates from the evaluator's NTP Server. The evaluator also configured the NTP server to send broadcast and multi-cast time updates such that they would be visible to the TOE. The evaluator observed that the TOE did not accept the time updates from the NTP Server.

2.2.8.4 NDcPP22e:FCS_NTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test 1: The evaluator shall confirm the TOE supports configuration of at least three (3) NTP time sources. The evaluator shall configure at least three NTP servers to support periodic time updates to the TOE. The evaluator shall confirm the TOE is configured to accept NTP packets that would result in the timestamp being updated from each of the NTP servers. The evaluator shall check that the time stamp is updated after receipt of the NTP packets. The purpose of this test to verify that the TOE can be configured to synchronize with multiple NTP servers. It is up to the evaluator to determine that the multi- source update of the time information is appropriate and consistent with the behaviour prescribed by the RFC 1305 for NTPv3 and RFC 5905 for NTPv4.

Test 2: (The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers).

The evaluator shall confirm that the TOE would not synchronize to other, not explicitly configured time sources by sending an otherwise valid but unsolicited NTP Server responses indicating different time from the TOE's current system time. This rogue time source needs to be configured in a way (e.g. degrade or disable valid and configured NTP servers) that could plausibly result in unsolicited updates becoming a preferred time source if they are not discarded by the TOE. The TOE is not mandated to respond in a detectable way or audit the occurrence of such unsolicited updates. The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers. It is up to the evaluator to craft and transmit unsolicited updates in a way that would be consistent with the behaviour of a correctly-functioning NTP server.

(TD0528 applied)

Test 1: The evaluator configured the TOE with 3 valid NTP connections. The evaluator changed the time on the NTP servers and observed that the TOE updated its time and synced with the three valid NTP servers.

Test 2: The evaluator kept the TOE configured with the same 3 NTP servers as in test 1. The evaluator collected network traffic while monitoring the time on the TOE while an untrusted NTP server was configured to broadcast to the TOE. The evaluator confirmed via packet capture that the TOE ignored the NTP packets and could not update time using traditional authenticated updates with the invalid NTP server.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.9 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.9.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.2 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.

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.

Section 6.2 in the ST states that the TOE implements a NIST-approved platform-based AES-CTR Deterministic Random Bit Generator (DRBG), in accordance with ISO/IEC 18031:2011. The DRBG is seeded by an entropy source that is at least 256-bit value. The TOE implements a random bit generator in support of various cryptographic operations, including, RSA key establishment schemes, Diffie-Hellman key establishment schemes and SSH session establishment. All random number generation functionality is continuously health tested as per the tests defined in section 11.3 of SP 900-90A. Any initialization or system errors during bring-up or processing of this system causes a reboot resulting in the DRBG being reseeded.

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

Enabling FIPS mode configures the TOE to use the proper DRBG methods. Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

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

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

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

instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

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

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

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

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

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

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.10 SSH Server Protocol (NDcPP22e:FCS_SSHS_EXT.1)

2.2.10.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.2 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.

Section 6.2 of the ST states that the TOE uses SSH for to facilitate secure remote administrative sessions (CLI). The TOE's SSH implementation supports the following:

- Use of 2048-bit RSA keys in support of SSH_RSA for public key-based authentication which is consistent with the RSA signature verification algorithm specified in FCS_COP.1/SigGen;
- Password based authentication;
- Public key based authentication. The TOE ensures and verifies that the SSH client's presented public key matches one that is stored within the TOE's SSH server's authorized keys database.

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.

Test 1: The evaluator generated an RSA 2048 bit key pair on the test server and then uploaded the key to the TOE and configured the admin user with the public key. The evaluator then attempted to login to the TOE using this

public key and observed that the login was successful.

Test 2: The evaluator attempted to connect to the TOE using a SSH client using a pubkey not configured on the TOE. The evaluator found that an unconfigured pubkey would not be used to establish an SSH session and the TOE would revert back to password authentication.

Test 3: The evaluator attempted to connect to the TOE using a SSH client alternately using the correct and incorrect password. The evaluator found that only the correct password would yield a successful SSH session.

Test 4: This test was performed as part of Test 3 as described above.

2.2.10.3 NDcPP22e:FCS_SSHS_EXT.1.3

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

Section 6.2 of the ST states that the TOE's SSH implementation supports dropping SSH packets greater than 262126 bytes. This is accomplished by buffering all data for a particular SSH packet transmission until the buffer limit is reached and then dropping the packet.

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 attempted to connect to the TOE using a SSH client and then sent a packet too large (262150 bytes). The evaluator observed that the TOE rejected the packet and the session was disconnected by the TOE.

2.2.10.4 NDcPP22e:FCS_SSHS_EXT.1.4

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

Section 6.2 of the ST states that the TOE's SSH implementation supports the following encryption algorithms: aes128-cbc, aes256-cbc, aes128-ctr and aes256-ctr to ensure confidentiality of the session. These algorithms are consistent with those listed in the requirement. No options included in the RFCs have been implemented.

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).

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

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 connect to the TOE using a SSH client alternately using each of the ciphers that can be claimed to determine which ciphers are supported with successful connections. The evaluator captured packets associated with each of the connection attempts and observed that the TOE was successfully able to connect using each of the algorithms claimed in the ST.

2.2.10.5 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.

Section 6.2 of the ST states that the TOE's SSH implementation supports the use of 2048-bit RSA keys in support of SSH_RSA for public key-based authentication. This is consistent with the algorithm specified in the requirement.

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).

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

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.

Test 1: The evaluator attempted to connect to the TOE using a SSH client alternately using each of the authentication algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator confirmed that the TOE supports the ssh-rsa algorithm as claimed. The evaluator followed up with a disallowed authentication algorithm and confirmed that it was not accepted.

Test 2: This test was performed as part of Test 1.

2.2.10.6 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 the ST states that the TOE's SSH implementation supports hashing algorithms hmac-sha1, hmac-sha2-256, and hmac-sha2-512 to ensure the integrity of the session. This is consistent with the algorithms identified in the requirement.

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

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are

considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

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 connect to the TOE using a SSH client alternately using each of the MAC algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator followed up with a disallowed MAC algorithm to ensure it was not accepted. The evaluator observed that only the hmac-sha1, hmac-sha2-256 and hmac-sha2-512 algorithms were able to successfully connect to the TOE.

Test 2: This was tested as part of Test 1.

2.2.10.7 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 the ST states that the TOE's SSH implementation supports the enforcement of diffie-hellman-group14-sha1, diffie-hellman-group14-sha256, diffie-hellman-group16-sha512 and diffie-hellman-group18-sha512

as allowed key exchange methods. This is consistent with the key exchange methods identified in the requirement.

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

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

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 - This test was performed as part of test 2 where the evaluator attempted to connect to the TOE using diffie-hellman-group1-sha1. The TOE rejects the attempt as expected.

Test 2 - The evaluator attempted to establish an SSH connection with each of the following claimed key exchange methods: diffie-hellman-group14-sha1, diffie-hellman-group14-sha256, diffie-hellman-group16-sha512 and

diffie-hellmangroup18-sha512 and observed that the connections were successful. The evaluator then attempted to establish an SSH connection using diffie-hellman-group1-sha1 and observed that the connection failed.

2.2.10.8 NDcPP22e:FCS_SSHS_EXT.1.8

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

a) Both thresholds are checked by the TOE.

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

Section 6.2 of the ST states that SSH session rekey limits are administrator configurable such that an SSH connection is rekeyed before 60 minutes of connection time or 1 GB of data traffic, whichever threshold is met first.

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

Section "Configure the SSH Rekeying Interval" in the **Admin Guide** provides instructions for configuring the SSH rekey data limit and the SSH rekey time interval using the 'configure ssh2 rekey' command. This section states that SSH servers rekey an SSH connection after the configured interval is reached or the configured amount of data is transferred (whichever occurs first). Notes in this section instruct the user that the data limit must not exceed 1GB and the time interval must not exceed 60 minutes.

Testing Assurance Activities: The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

For testing of the time-based threshold the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

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

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached

(e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8).

The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

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

If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1(3)/Functions).

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

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

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

The evaluator configured the rekey data limit to 1 MB. The evaluator attempted to connect to the TOE using a SSH client sending data and confirmed that a rekey happened at the 1 MB threshold. Next the evaluator configured the rekey time limit to 15 minutes. The evaluator attempted to connect to the TOE using an SSH client and confirmed that a rekey happened when the configured threshold was reached.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.11 TLS Client Protocol Without Mutual Authentication (NDcPP22e:FCS_TLSC_EXT.1)

2.2.11.1 NDcPP22e:FCS_TLSC_EXT.1.1

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

Section 6.2 of the ST states that the TOE exclusively supports TLS v1.2 secure communication protocol that complies with RFC 5246. The TOE supports mutual X509v3 certificate-based authentication and the following ciphers:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256,
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384

These ciphersuites are consistent with those identified in the FCS_TLSC_EXT.1 requirement.

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.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported TLS Ciphers and Curves" in the **Admin Guide** specifies the TLS ciphers and curves supported in the evaluated configuration when the switch is configured in FIPS mode.

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 a 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 each of the claimed ciphersuites in turn and verified that the connections were successful.

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_DHE_RSA_WITH_AES_128_CBC_SHA256 but then sends a DSA key in its certificate message. The evaluator used 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 and observed that the TLS session was 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 and observed that the TOE rejected the connection.

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 the connection.

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 and 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.

2.2.11.2 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 the ST states that the TOE implements reference identifier matching according to RFC 6125. The reference identifier is specified during configuration of the TLS connection. Supported reference identifiers are DNS names for the SAN and CN. As part of negotiating TLS connections, the TOE will verify that the peer certificate's Subject Alternative Name (SAN) or Common Name (CN) contains the expected identifier. The CN is checked only if the SAN is absent. The TOE only establishes a connection if the peer certificate is valid, trusted, and has a matching reference identifier and if the revocation check passed. The TOE does not implement certificate pinning. The TOE does not support identifiers that include wildcards or IP addresses. The TOE supports X509v3 certificates as defined by RFC 5280 to mutually authenticate TLS connections. The TOE presents Elliptic Curves/ Groups Extension with the following curves/groups: secp256r1, secp384r1and secp521r1.

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.

Section "TLS Negotiation" in the **Admin Guide** states that the TOE supports reference identifier matching, according to RFC 6125. The reference identifier is specified during configuration of the TLS connection. Supported reference identifiers are DNS names for the Subject Alternative Name (SAN) and the Common Name (CN). As part of negotiating the TLS connection, EXOS verifies that the peer certificate's SAN or CN contains the expected reference identifier. The CN is checked only if the SAN is absent. Then, a connection is established only if the peer certificate is valid, trusted, has a matching reference identifier, and passes the revocation check.

Section "Enable a TLS Connection to the Syslog Server" in the **Admin Guide** provides the command used to specify the remote syslog server certificate reference identifier.

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, CN=2001:0DB8:0000:0000:0008:0800:200C:*when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

This negative test corresponds to the following section of the Application Note 64/105: 'The exception being, the use of wildcards is not supported when using IP address as the reference identifier.'

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

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.)

Test 1: The evaluator configured the TOE with a DNS name in the reference identifier. The evaluator then established a TLS session from the TOE targeting a server using a valid certificate with a SAN/CN matching the domain name used by the client and observed a successful connection. The evaluator then attempted to establish a TLS session from the TOE targeting a server using a server certificate that does not contain an identifier in the

Subject Alternative Name (SAN) and a Common Name (CN) that does not match the reference identifier. The evaluator observed that the connection failed.

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. The evaluator observed that the connection failed.

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. The evaluator observed that the TLS session was negotiated successfully.

Test 5: As demonstrated by testing, the TOE does not support the use of wildcards.

Test 6: Not applicable. The TOE does not claim IP addresses are supported as reference identifiers.

Test 7: Not applicable. The TOE does is not claiming FPT_ITT communication.

2.2.11.3 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: This test was performed as part of FCS_TLSC_EXT.1.1-t1 which demonstrates successful connections with a valid certificate chain.

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

- match the reference identifier -- Corresponds to FCS_TLSC_EXT.1.2 Tests 1 through 7.
- 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: Not applicable. The TOE does not support administrative override.

2.2.11.4 NDcPP22e:FCS_TLSC_EXT.1.4

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

Section 6.2 of the ST states that the TOE presents Elliptic Curves/ Groups Extension by default with the following curves/groups: secp256r1, secp384r1and 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.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are

considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported TLS Ciphers and Curves" in the **Admin Guide** specifies the TLS ciphers and curves supported in the evaluated configuration when the switch is configured in FIPS mode.

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

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

- ECDHE w/ P-256 curve
- ECDHE w/ P-384 curve
- ECDHE w/ P-521 curve

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.12 TLS Client Support for Mutual Authentication (NDcPP22e:FCS_TLSC_EXT.2)

2.2.12.1 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 the ST states that the TOE supports X509v3 certificates as defined by RFC 5280 to mutually authenticate TLS connections.

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 "Generate a Certificate Signing Request" section in the **Admin Guide** provides instructions for generating a CSR that can be signed by a Certificate Authority (CA).

The "Install Certificates on the Syslog Client" section in the **Admin Guide** provides instructions for installing certificates on the syslog client including installing the CSR generated and signed client-side certificate for TLS mutual authentication.

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.

TD0670 Applied.

The tests for FCS_TLSC_EXT.1 and FIA_X509_EXT.* were all executed with a server requiring mutual authentication (i.e., requiring the TOE to present a certificate). In each successful TLS connection (e.g. FCS_TLSC_EXT.1.1, test 1), the evaluator observed that the TOE TLS client sends both client Certificate (type 11) and Certificate Verify (type 15) messages during its negotiation 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 of the ST states that a user account will be locked after an administrator-configurable (1 to 10) number of unsuccessful authentication attempts. Once the user is locked out, all further authentication attempts are reported as unsuccessful, even when correct information is provided. To regain access, the user has to wait an administrator-configurable time duration before being allowed to successfully authenticate. The TOE ensures that authentication failures by remote administrators cannot lead to a situation where no administrator access is available, by distinguishing between local and remote login attempts.

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.

Section "Configure Password Settings" in the **Admin Guide** provides instructions for configuring the number of successive unsuccessful authentication attempts and the time period.

Section "Create a Failsafe Account" in the **Admin Guide** provides instructions for configuring a failsafe account that provides access to the device if the other accounts cannot be used. It is a last resort for accessing a device.

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

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

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

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

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

Test 1 & 2: The evaluator performed lockout testing on each TOE device with two different values to verify that the configuration works correctly. The first case was performed with an invalid attempt threshold of 3 attempts and a lockout time of 1 minute. The second case was performed with an invalid attempt threshold of 5 attempts and a lockout time of 3 minutes. In each case, the evaluator observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator also observed that the account was unlocked after the configured time period had passed.

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 charters supported for administrator passwords.

Section 6.3 of the ST states that passwords can be composed of any combination of upper and lower case letters, numbers, and the following special characters: “!”, “@”, “#”, “$”, “%”, “^”, “&”,“*”, “(“, and “)”. The minimum password length is settable by the Authorized Administrator and is configurable to between 1 and 32 characters. In the evaluated configuration, a minimum password length of 15 characters is recommended.

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.

Section "Configure Password Settings" in the **Admin Guide** identifies the characters that may be used in passwords consistent with the ST and addresses the composition of strong passwords with instructions for password composition including the specifying required characters, maximum password age and the password policy history. Instructions are also provided for configuring the minimum password length between 1 and 32 characters. In the evaluated configuration the minimum password length is 15 characters.

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

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

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

Test 1&2: The evaluator performed attempts to set passwords of varying lengths and characters to demonstrate that passwords comply with a minimum length and support the claimed alphabet.

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 TOE always protects authentication data as it is entered, no administrative actions are needed for this feature.

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 not echoed back while being entered at 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.

See 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 the ST states that the TOE requires any user to be identified and authenticated before any management action. In the evaluated configuration, the TOE does not allow unauthenticated configuration of the TOE's network routing/switching services and does not allow any unauthenticated management action. The TOE does not offer any services or access to its functions, except for the displaying a message of the day banner, without requiring a user to be identified and authenticated.

A requesting user will be prompted to enter a user name and password upon establishing a successful connection. The TOE will then compare the entered credentials against the known user database. If the combinations match, the TOE will then attribute (bind) the administratively-assigned role (a predetermined group of privileges that dictate access to TOE functions) to that user for the duration of their logged-in management session.

For remote administration (implemented as a CLI over SSH) the TOE can be configured to authenticate using a public key mechanism (RSA), or a password-based mechanism. If a user attempts public key-based authentication and it succeeds, the authentication process is completed and the user is granted access. If the user fails to authenticate using a public key certificate, then the TOE falls back to password-based authentication and requires the user to enter a valid username and password. When a user attempts to authenticate using password based authentication, they are prompted to enter in a valid username and password. If the user succeeds, the authentication process is completed and the user is granted access to the command line prompt for the CLI. If the authentication process fails, then the user will be prompted to re-enter their credentials. When RSA authentication is used, the TOE checks the presented public key against its authorized keys database and verifies the user's possession of a private key by negotiating a secure channel using the public key associated with that private key.

For local administration, the CLI is accessed by connecting to the console port with the provided RJ45-to-DB9 cable. Local administration supports only password-based authentication.

Upon successful authentication, the TOE assigns an administratively defined role to the user for the duration of the user's logged-in session. The TOE facilitates all administrative actions through the CLI. Successful login is indicated by TOE offering a CLI command prompt.

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.

Section "Establish a Serial Connection" in the **Admin Guide** provides instructions for connecting a terminal to the serial console interface to monitor and configure the system directly.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Enable and Configure SSH" in the **Admin Guide** provides instructions for enabling SSHv2 on the TOE, configuring the SSH rekey interval and enabling SSH and console session timeout.

Section "Generate SSH Host Keys" in the **Admin Guide** provides instructions for generating an SSH2 host key. SSH2 host keys are used to authenticate connections between the device and clients on remote systems. A host key must be generated before the device can accept incoming SSH connections.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Supported TLS Ciphers and Curves" in the **Admin Guide** specifies the TLS ciphers and curves supported in the evaluated configuration when the switch is configured in FIPS mode.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** provides instructions for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

Section "User Key-Based Authentication" in the **Admin Guide** provides instructions for configuring SSH public key for user authentication.

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 following user interfaces where authentication is provided.

- Local console using local authentication with password
- SSH CLI using local authentication with password
- SSH CLI using local authentication with public key

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

Test 2 - Using each interface the evaluator was able to observe that the TOE displayed a banner to the user before login and that there were no other services available nor any configuration options offered to administrators to control services available prior to authentication.

Test 3 - This test was performed as part of test 1 & 2. Using each interface, the evaluator found that, prior to login, no functions were available to the administrator with the exception of acknowledging the banner.

Test 4 - Not applicable. The TOE is not a distributed TOE.

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)

Test 1a: The successful connection with a valid cert chain was demonstrated in FCS_TLSC_EXT.1.1-t1.

Test 1b -- For the invalid chain, the evaluator configured a server certificate with an invalid certification path by deleting an intermediate root CA so that the certificate chain was invalid because of a missing (or deleted) certificate. The connection between the peers was refused.

Test 2 -- The evaluator attempted to make a connection between the TOE and a test server. The test server then presented an expired certificate during the negotiation resulting in a failed connection.

Test 3 -- The evaluator attempted to make a connection between the TOE and a test server. The test server then presented a certificate during the TLS negotiation where the certificate was valid. A packet capture was obtained of this TLS negotiation which shows that the connection was successful. The evaluator revoked certificates in the chain (individually) and attempted the same connection. The attempt after revoking the certificate was not successful.

Test 4 -- The evaluator configured an OCSP responder to present a certificate that does not have the OCSP signing purpose. The evaluator established a connection between the TOE and a test server such that the TOE receives OCSP response signed by the invalid certificate and ensured that the connection 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, the connection failed.

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, the connection failed.

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, the connection was refused.

Test 8a, 8b, 8c: Not applicable. The TOE does not support ECDSA Certificates.

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 server to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then attempted a connection between the TOE and the test server and observed that the TOE rejected the connection.

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 attempted a connection between the TOE and the test server and observed that the TOE rejected the connection.

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step

and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

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

Section 6.3 of the ST states that the TOE supports the use of X.509v3 certificates as defined by RFC 5280 to mutually authenticate external IT entities. When a X509 certificate is presented during a TLS handshake, the TOE verifies the trust chain, performing validation of the certificates and carries out revocation checking of each certificate. The revocation check is performed by sending an OCSP request to a trusted OCSP responder and verifying the signed response. If the TOE cannot establish a connection to the OCSP Responder to determine the revocation status of a certificate, it will not accept the certificate and the session will not be established. Certificate pinning is not supported.

The TOE supports the use of X.509v3 certificates as defined by RFC 5280 to authenticate connections with authorized IT entities. When certificate based authentication is used, the TOE validates the presented certificate, checking its chain of trust against the TOE's internal trusted store, and performs a certificate revocation check. Certificate validation includes path validation (including checking CA certificates) certificate processing (including validating the extendedKeyUsage field), and extension processing (including checking the BasicConstraints extension). Verifying the chain of trust includes validating each certificate in the chain, verifying that the certificate path consists of trusted CA certificates, and performing revocation checks on all certificates in the path. Revocation checking is implemented using OCSP. If any part of the authentication fails, the connection is terminated at the handshake stage. Specifically, the TOE implements a mutually authenticated secure channel, using TLSv1.2, to connect to a trusted external audit server.

The TOE supports the following methods to obtain a certificate from a trusted CA:

- Manually import certificates in PEM format from an external server.

Once the CA certificate is downloaded, and prior to adding it to an existing list of trusted certificates, the TOE verifies the following:

- The Basic constraints extension with the CA flag is set to true
- The Key usage extension with the "keyCertSign" bit is set
- The Certificate is not expired

All certificates are stored in a private, persistent location on the TOE. There is no direct access to stored certificates using regular interfaces.

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.

Section "X.509 Certificate-Based Authentication" in the **Admin Guide** states that secure syslog uses X.509 certificates for confidentiality and integrity. When an X.509v3 certificate is presented for authentication during a TLS handshake, the TOE validates the certificate, checks the chain of trust against its internal trusted store and performs a certificate revocation check.

Certificate validation includes the following:

- Validating the path, including checking CA certificates.
- Processing the certificate, including validation of the `extendedKeyUsage` field.
- Processing extensions, including the `BasicConstraints` extension.

Chain of trust verification includes the following:

- Validating each certificate in the chain.
- Verifying that the certificate path consists of trusted CA certificate

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 of the ST states that the TOE generates certificate requests and validates the CA used to sign the certificates. Once the certificate has been issued, the administrator can import the X.509v3 certificate into the TOE. This allows the TOE to determine which CA certificate(s) to use during the validation process. In order to verify the revocation status of the presented certificates Online Certificate Status Protocol (OCSP) is used. If the connection to determine the certificate validity cannot be established, the TOE does not accept the certificate.

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.

Section "X.509 Certificate-Based Authentication" in the **Admin Guide** states that the network peers in the operating environment to which EXOS will connect (using TLS) must be configured to present a valid X.509v3 certificate issued by a trusted CA.

The "Generate a Certificate Signing Request" section in the **Admin Guide** provides instructions for generating a CSR that can be signed by a Certificate Authority (CA).

The "Install Certificates on the Syslog Client" section in the **Admin Guide** provides instructions for installing certificates on the syslog client including installing the trusted CA and the CSR generated and signed client-side certificate for TLS mutual authentication.

Section "TLS Negotiation" in the **Admin Guide** states that if the TLS session fails because the OCSP server cannot be contacted, the administrator is instructed to verify the network path to the OCSP server and the status of the server, and to fix any issues.

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 connection between the TOE and a remote test server and obtained a packet capture of the activity. This was repeated when the OCSP server was available and unavailable. When available the connection succeeded. When unavailable, the connection failed.

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 ST does not select 'device-specific information'.

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

Section "Generate a Certificate Signing Request" in the **Admin Guide** provides instructions for generating a private-key and CSR that can be signed by a CA. The instructions include a description of the "configure ssl csr" command which allows the administrator to establish the claimed fields: common name, organization, organizational unit and country.

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 evaluator viewed the contents of the CSR and confirmed that it provided public key and other required information including the information that the evaluator supplied during the generation process.

Test 2 - The evaluator attempted to import a certificate that was signed by a root CA that was not installed on the TOE. The attempt failed. The evaluator then successfully installed the certificate resulting from the CSR request in test 1.

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.

Not applicable as the TOE is not a distributed TOE.

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).

Section "Software Upgrade" section in the **Admin Guide** describes the necessary steps to perform a manual update. This includes where to obtain the upgrade files and the method of transferring the upgrade files to the TOE. It also describes the image integrity checking. Digital signature is used to demonstrate the authenticity of the image downloaded to the switch. Only images with digital signature validated on the switch can be installed, otherwise the installation will fail. The signature is computed for the images when they are built and then included in the final image. During downloading process on the switch, the signature is verified against the image using the installed public key.

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.

Test 1 - The evaluator configured a user as a non-administrator and then attempted to import an update as the non-administrator. The evaluator confirmed that this action was not available to the non-administrative user.

Test 2 - See FPT_TUD_EXT.1 for a successful update.

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 6.3 of the ST states that the TOE does not offer any services or access to its functions, except for the displaying a message of the day banner, without requiring a user to be identified and authenticated. The TOE requires any user to be identified and authenticated before any management action.

Section 6.4 of the ST states that the TOE allows remote administration via SSHv2 session and local administration via a directly connected console cable. Both remote and local administration utilize a Command-Line Interface (CLI). The CLI provides access to all management functions used to administer the TOE. The TOE requires each user to be successfully authenticated before allowing any other action on behalf of that user. Security management is restricted to administrators. The trust store is accessed when administrators import/remove certificates as described in the **Admin Guide**. The trust store is protected by default and is restricted such that only administrators have access.

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 TSF data manipulating functions and the corresponding configuration information from specific sections of the **Admin Guide** are identified or referenced throughout this AAR with the requirement to which they apply.

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. All security management functions are restricted to a valid administrator with the admin role.

The "Change the Passwords for the Default Accounts" section in the **Admin Guide** states that an EXOS device is configured with two accounts, admin and user and provides instructions for configuring these accounts.

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

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

All of the management functions have been exercised under the other SFRs as demonstrated throughout the AAR.

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

2.4.3.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 of the ST states that only administrators can perform management operations including the command to generate and delete cryptographic keys. Administrators can also import and delete CA certificates and their keys into the trust store.

Table 15 in Section 6.2 of the ST identifies the CSPs used in the TOE and includes commands that can be triggered manually by an administrator to delete/zeroize RSA and TLS client keys. Section 6.2 further descries the TOE's SSH implementation uses 2048-bit RSA keys in support of SSH_RSA for public key-based authentication.

Section 6.3 of the ST indicates that the TOE generates certificate requests and validates the CA used to sign the certificates. Once the certificate has been issued, the administrator can import the X.509v3 certificate into the TOE.

Refer to the Guidance assurance activities below for a description of how these operations are performed.

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

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

Section “Supported SSH Ciphers and Keys” in the **Admin Guide** lists the SSH ciphers and keys supported in the evaluated configuration. Section “Enable and Configure SSH” provides instructions for the administrator to restrict SSH algorithms and keys and to generate SSH host keys. Section “User Key-Based Authentication” describes how to generate an SSH key pair, associate the key with a specific user as well as instructions for disassociating a key from a user and removing the key from the database.

Section “Supported TLS Ciphers and Curves” in the **Admin Guide** lists the TLS Ciphers and Curves supported in the evaluated configuration. Beyond enabling FIPs mode, there is no further configuration needed to restrict the available ciphers/curves to those allowed.

Section “X.509 Certificate-Based Authentication” in the **Admin Guide** describes how to generate a certificate signing request which results in a CSR and key pair being generated. Section “Install Certificates on the Syslog Client” provides instructions for downloading trusted CA certificates or CSR-signed certificates to the TOE’s trust store and also includes commands for removing these certificates from the trust store.

Section "Zeroization" in the **Admin Guide** identifies the Critical Security Parameters (CSPs) used in the TOE including identification of the TLS and SSH related keys. It further identifies use of the administrator command “unconfigure switch [all/erase]” which zeroizes these keys by resetting the platform to factory defaults.

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 successful generation of a new CSR and private key, and the subsequent import of CA and signed certificate by an authorized administrator is demonstrated in NDcPP22e:FIA_X509_EXT.3.

The evaluator configured a user as non-administrator and attempted to generate a new CSR as the non-administrator user. The evaluator confirmed that the management of crypto keys and certificates was not available to the non-administrative user.

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 of the ST provides the TSS which details all TOE security management functions available through the TOE interfaces. The TOE is compliant with all requirements in the ST as identified in the TSS and Guidance assurance activities throughout this report.

Section 6.4 of the ST states that the TOE allows remote administration via SSHv2 session and local administration via a directly connected console cable. Both remote and local administration utilize a Command-Line Interface (CLI). The CLI provides access to all management functions used to administer the TOE.

Section 6.3 in the ST further states that for local administration, the CLI is accessed by connecting to the console port with the provided RJ45-to-DB9 cable. Local administration supports only password-based authentication.

Section "Establish a Serial Connection" in the **Admin Guide** describes connecting a terminal to the serial console interface in order to configure the system directly. This is performed by connecting the RJ-45 or USB cable to the console port on the device and connecting the other end of the cable to the terminal or computer serial port. This is sufficient to inform the administrator how to ensure that the interface is local.

Component Guidance Assurance Activities: See TSS Assurance Activities

See TSS Assurance Activities.

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 6.4 of the ST describes the TOE supported roles and restrictions. The TOE supports two separate privilege levels: User and Security Administrator. There is no way to enable a "privileged" or "supervisor" level from a User account. All of the management functions are restricted to the Security Administrators of the TOE.

A user-level account has viewing access to all manageable parameters, and changing their password, with the exception of no access to:

- The user account database
- SNMP community strings

A person with an administrator-level account can view and change all switch parameters. With this level, users can be added and deleted, and the password associated with any account name can be changed.

The term "Security Administrator" is used to refer to any administrative user with the appropriate role with sufficient privilege to perform all relevant functions. All administrators have the same permissions and all users have the same permissions. The privilege level determines the functions the user can perform.

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 **Admin Guide** contains instructions for managing all required aspects of the TOE both locally and remotely as documented throughout this AAR. The TOE allows remote administration via SSHv2 session and local administration via a directly connected console cable. Both remote and local administration utilize a Command-Line Interface (CLI). The CLI provides access to all management functions used to administer the TOE.

Section "Establish a Serial connection" in the **Admin Guide** provides instructions for connecting a terminal to the serial console interface to monitor and configure the TOE with a direct, local connection.

Section "Configure the Out-of-Band Management Interface" in the **Admin Guide** provides instructions for configuring the IP address for the management interface so that the TOE can be remotely accessed using the out-of-band management port. Section "Configure the In-Band Management Interface" in the **Admin Guide** provides similar instructions for the x435 TOE model which does not include an out of band management interface.

Section "Device Access" in the **Admin Guide** describes the two options for accessing an EXOS device - serial connection or SSH using the 'ssh' command.

Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Enable and Configure SSH" in the **Admin Guide** provides instructions for enabling SSHv2 on the TOE, configuring the SSH rekey interval and enabling SSH and console session timeout.

Section "Generate SSH Host Keys" in the **Admin Guide** provides instructions for generating an SSH2 host key. SSH2 host keys are used to authenticate connections between the device and clients on remote systems. A host key must be generated before the device can accept incoming SSH connections.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** provides instructions for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

Section "User Key-Based Authentication" in the **Admin Guide** provides instructions for configuring SSH public key for user 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.

The TOE allows remote administration via SSHv2 session and local administration via a directly connected console cable. Both remote and local administration utilize a Command-Line Interface (CLI). The CLI provides access to all management functions used to administer the TOE. The CLI was used both locally and remotely (SSH) throughout the course of testing.

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 the ST states that the TOE protects Critical Security Parameters (CSP) such as stored passwords and cryptographic keys so they are not directly accessible via normal administrative interfaces. All passwords are encrypted using AES256-CBC. Passwords are stored in the TOE's configuration file in encrypted format. The master key used for AES encryption of passwords is generated randomly; the salt part involved in the key generation is also random. This key is generated afresh for every usage and stored in RAM. The salt is generated using cryptographically strong pseudo-random bytes. The fixed string, salt and the cipher name (AES) are passed to OpenSSL's EVP_BytesToKey key derivation function. EVP_BytesToKey will return the derived key and initialization vector (IV) that will be used by the cipher AES-CBC to encrypt the password(s). Additionally, when login-related configuration information is accessed through regular TOE interfaces, it is obfuscated with a series of asterisks.

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 the ST states that the TOE protects Critical Security Parameters (CSP) such as stored passwords and cryptographic keys so they are not directly accessible via normal administrative interfaces. All passwords are encrypted using AES256-CBC. Passwords are stored in the TOE's configuration file in encrypted format. The master key used for AES encryption of passwords is generated randomly; the salt part involved in the key generation is also random. This key is generated afresh for every usage and stored in RAM. The salt is generated using cryptographically strong pseudo-random bytes. The fixed string, salt and the cipher name (AES) are passed to OpenSSL's EVP_BytesToKey key derivation function. EVP_BytesToKey will return the derived key and initialization vector (IV) that will be used by the cipher AES-CBC to encrypt the password(s). Additionally, when login-related configuration information is accessed through regular TOE interfaces, it is obfuscated with a series of asterisks.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.3 Reliable Time Stamps (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 the ST states that the TOE is a hardware appliance that implements a hardware-based real-time clock that is managed by the embedded OS, which also controls the exposure of administrative functions. This clock is used to produce reliable timestamps that are available for audit trail generation, synchronization with the operational environment, session inactivity checks, and certificate expiration validation. The TOE also has the option to use NTP for network time.

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.

Section "Set the System Date, Time, and Time Zone" in the **Admin Guide** provides instructions for manually configuring the date, time and time zone.

Section "Network Time Protocol" in the **Admin Guide** provides instructions for configuring the TOE to use NTP including how to add and delete NTP servers and how to enable ntp authentication and create a SHA-256 key. The TOE supports NTPv3 and there is no further configuration needed.

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 set the clock from the local console and observed the time change via the local console interface.

Test 2: This test has been performed as part of the tests in NDcPP22e: FCS_NTP_EXT.1.1-t1 where the evaluator configured the TOE to get NTP time updates from the evaluator's NTP server and demonstrated that the TOE successfully synchronized its time with the NTP server.

Test 3: Not applicable. The TOE does not include an underlying VS.

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 the ST states that the TOE performs diagnostic self-tests during start-up and generates audit records to capture any failures. Some low-level critical failure modes can prevent TOE start-up, and as a result will not generate audit records. In such cases, the TOE will enter a failure mode displaying error codes, typically on the console. The TOE can be configured to reboot or to stop with errors displayed when non-critical errors are encountered. The cryptographic module performs self-tests during startup; messages from the module are displayed on the console and audit records are generated for both successful and failed tests. These self-tests comply with the FIPS 140-2 requirements for self-testing. The module performs known-answer algorithm testing (KAT), and integrity testing. For each KAT test, the TOE uses known data as inputs into each cryptographic function, computes a cryptographic result, and compares the calculated result to the expected/known value. The integrity testing verifies the digital signature of the image (as described below) to ensure that it has not been tampered with or corrupted. These self-tests cover all anticipated modes of failure, and therefore are sufficient to ensure that the TSF operates correctly. Failure of any of the FIPS mode tests during the boot process will stop the start-up process and prompt the user to reload. For all start-up tests, successful completion is indicated by the TOE reaching operational status.

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.

Section "Self-Test Audit Log Records" in the **Admin Guide** states that self-tests are performed during start-up of the device and audit records are generated for successful and failed tests. These self-tests, which consist of known-answer algorithm testing and integrity testing, comply with FIPS 140-2 requirements for self-testing. The tests cover all anticipated modes of failure. Failure of any self-test during the start-up process stops the process and prompts the user to reload.

The following is an example of a log entry for a successful self-test.
08/14/2021 14:17:49.99 <Noti:SNMP.Master.EnblFIPSModeOK> Self-Test passed. FIPS mode enabled.

The following is an example of a log entry for a failed self-test.
06/13/2021 13:46:29.61 <Erro:exsshd.EnblFIPSModeFail> Failed to enable FIPS mode:
error:2D080086:lib(45):func(128):reason(134)

When some low-level critical failure modes prevent the switch from starting up, audit records are not generated. In such cases, the switch enters a failure mode and displays error codes, typically on the console. You can configure the switch to reboot or to stop, with errors displayed, when non-critical errors are encountered. The cryptographic module performs self-tests during start-up. Messages from the module are displayed on the console and audit records are generated for both successful and failed tests.

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 initiated a reboot from each device and confirmed that the audit logs show that the FIPS self-tests were executed successfully.

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 the ST states that the TOE implements two boot partitions - primary and secondary. An authorized administrator can configure the partition that is to be used after rebooting of the TOE. Firmware updates are always installed into the inactive partition. The default patching behavior is to upload the image, verify image, install it into the inactive partition, change the boot partition, and reboot the TOE. Administrators can override this behavior but are trusted not to. Upgrading EXOS is a multi-step process performed by a Security Administrator.

An authorized user must authenticate to the Extreme Portal website at https://extremeportal.force.com where the software downloads are available. The downloaded image must be transferred to the appliance using a method such as TFTP. The TOE image files are digitally signed using a RSA mechanism. SHA-256 is used in conjunction with RSA signatures for verification of software image integrity. The TOE uses a public key to verify the digital signature; upon successful verification of this signature the TOE will apply the new image upon rebooting. If the signature verification cannot be carried out then the installation is terminated. The digital certificate used by the update verification mechanism is contained on the TOE. The version of the software can be queried by issuing the command: “show version”.

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.

Section "Software Upgrade" in the **Admin Guide** describes the necessary steps to perform a manual update. This includes where to obtain the upgrade files and the method of transferring the upgrade files to the TOE. The TOE

version may be queried using the 'show version' or 'show system' command. This section also describes the image integrity checking. Digital signature is used to demonstrate the authenticity of the image downloaded to the switch. Only images with digital signature validated on the switch can be installed, otherwise the installation will fail. The signature is computed for the images when they are built and then included in the final image. During downloading process on the switch, the signature is verified against the image using the installed public key.

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

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

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

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

2) An image that has not been signed

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

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

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

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

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

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

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

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

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

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

Test 2: The evaluator attempted to perform a TOE update using a legitimate update that was modified in three ways:

1. Corrupted Image/Valid Signature - A few bytes in the update file is modified via a hex editor
2. Valid Image/No Signature - Update is missing a signature
3. Valid Image/Invalid Signature - Update's signature is corrupted

In each case, the TOE rejected the modified update and the product version did not change.

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

2.6 TOE ACCESS (FTA)

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

2.6.1.1 NDCPP22E:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 of the ST states that the TOE implements remote and local administrative access via the CLI. The TOE's minimum lockout value must be configured to a non-zero value to enforce an administrator-defined inactivity timeout, after which the inactive session is automatically terminated. The inactivity timeout value is between 1-240 minutes, and the default value is 20 minutes. Once a session (local or remote) has been terminated, the TOE requires the user to re-authenticate.

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.

Section "Enable SSH and Console Session Timeout" in the **Admin Guide** provides instructions for configuring the inactivity time period for remote administrative session termination using the 'configure ssh2 idletimeout' command.

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.

Test 1: 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 the 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.

Section "Device Access" in the **Admin Guide** indicates that a local or remote interactive session can be terminated by the administrator running the 'exit' or ‘logout’ command.

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

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

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

Test 1 - The evaluator established a local session and manually logged out using the "logout" command. The session was terminated and the console displayed only the logon banner and the prompt for a username.

Test 2 - The evaluator established a remote session and manually logged out using the "logout" command. The session was terminated and the application closed requiring the user to reauthenticate to gain access.

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

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

Section "Enable SSH and Console Session Timeout" in the **Admin Guide** provides instructions for configuring the inactivity time period for local administrative session termination using the 'configure cli idle-timeout' command.

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

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

configured time period. If locking was selected from the component, the evaluator then ensures that reauthentication is needed when trying to unlock the session.

Test 1 - 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 "cli idle-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 6.4 of the ST states that the TOE allows remote administration via SSHv2 session and local administration via a directly connected console cable. Both remote and local administration utilize a Command-Line Interface (CLI). The CLI provides access to all management functions used to administer the TOE.

Section 6.6 of the ST states that the TOE can be configured to display administrator-defined advisory banners when administrators successfully establish interactive sessions with the TOE, allowing administrators to terminate their session prior to performing any functions. The TOE will display a customizable banner when a user initiates an interactive session either locally or remotely.

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

Section "Configure the Banner Message" in the **Admin Guide** provides instructions for configuring banner messages to provide information to users accessing the EXOS CLI. Instructions are provided for configuring both a pre-login and after-login message.

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.

Test 1 - The evaluator set a login banner for each method of access and then tested that the banner is displayed.

2.7 Trusted path/channels (FTP)

2.7.1 Inter-TSF trusted channel (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 the ST states that the TOE protects communications with the external audit server by establishing a trusted channel between itself and the audit server. To implement this trusted channel, the TOE uses TLS v1.2 protocol with certificate-based authentication. For certificate-based authentication, the X.509v3 certificate presented by the external audit server is first validated and then compared to the authorized certificates database.

In the evaluated configuration, the TOE must be configured to use TLS to ensure that exported audit records are sent only to the configured server so they are not subject to inappropriate disclosure or modification as the TOE validates the audit server against the TOE configuration using the certificates presented during TLS negotiation.

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.

Refer to FCS_TLSC_EXT.1 and FCS_TLSC_EXT.2 which describe the instructions for establishing TLS between the TOE and an external syslog server.

Section "Audit Logs and Syslog" in the **Admin Guide** states that if the connection to the external audit server is lost, the EXOS switch continues to save local audit logs so there is no loss of audit. An automated log reconciliation process (syncing) occurs between the locally stored records with the external audit server when the connection is reestablished.

Section "TLS Negotiation" in the **Admin Guide** states that if the TLS session fails because the OCSP server cannot be contacted, the administrator is instructed to verify the network path to the OCSP server and the status of the server and to fix any issues.

Section “Reconnect a TLS Session” in the **Admin Guide** provides instructions for manually reconnecting a TLS session that is inadvertently disconnected using the steps and commands to disable and enable the syslog 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.

Test 1 - This test was performed as part of NDcPP22E:FCS_TLSC_EXT.1/2 where the syslog TLS channel was fully tested and the evaluator confirmed that the TOE initiates the connection and that the traffic is not plaintext.

Test 2 - The TOE is able to initiate communication via the trusted channel for transmitting audit records to a secure syslog server. The evaluator configured the TOE's audit forwarding feature to forward the audits generated on the TOE's internal log server over a TLS protected connection to a secure syslog server. The evaluator used this configuration to capture many of the audit messages for the TOE and tested that this process was done over a trusted channel (TLS) in FCS_TLSC_EXT.1/2. In all of these test cases, the TOE initiates a connection with the test

server.

Test 3 - Packet captures were generated during TLS syslog testing to ensure it was encrypted. Throughout the testing process, the TOE never sends any plaintext channel data and is able to successfully establish a secure tunnel after a two-way handshake over the configured channels.

Test 4 - The evaluator physically disrupted the connection for about 40 seconds and then reconnected. The evaluator observed that the TLS session remained secure without requiring a new TLS handshake and no TSF data was sent in plaintext. A second test with a 4.5 hour disruption was also performed. After the physical disruption, the evaluator observed that a new TLS handshake was initiated. In both of these cases, the evaluator noted that at no time was any TSF data sent in plaintext. The evaluator analyzed the traffic in both cases and found no instance of sensitive data being sent outside the protected TLS channel.

2.7.2 Trusted Path (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 the ST states that the TOE protects remote management sessions by establishing a trusted path (using SSH) between itself and the administrator. The SSHv2 session is encrypted using AES encryption. The remote administrators are able to initiate the SSHv2 secure channel with the TOE. The remote session is secured (disclosure and modification) using CAVP tested cryptographic operations, and all remote security management functions require the use of this secure channel.

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

Section "Common Criteria Certification Configuration" in the **Admin Guide** states that when administrators log in with role-based credentials, their access is limited to commands they have privileges and permissions to use based on Common Criteria standards. Additionally, network management communication paths are protected against modification and disclosure using SSHv2 and TLS. The audit channel to an external syslog server is protected using TLS encapsulation. FIPS 140-2 Security Level 1 specifies the security requirements that are satisfied by a cryptographic module used in a security system that protects a system's sensitive information. Common Criteria compliance mode supports devices running EXOS version 31.3.100. Cryptographic Algorithm Validation System (CAVS) certifies all cryptographic algorithms required by and used in Common Criteria.

Section "FIPS Mode" in the **Admin Guide** states that when implemented, Federal Information Processing Standards (FIPS) mode disables some services and disables certain cryptographic, hashing, and signature algorithms that are considered insecure. When FIPS mode is enabled, EXOS uses the openssl-fips-2.0.16 OpenSSL library. Section "Enable FIPS Mode" in the **Admin Guide** provides instructions for enabling FIPS mode.

Section "Supported SSH Ciphers and Keys" in the **Admin Guide** specifies the SSH Ciphers and keys supported in the evaluated configuration. These ciphers and keys are claimed or allowed only when the switch is configured in FIPS mode and when certain additional restrictions are configured.

Section "Restrict SSH Algorithms and Keys" in the **Admin Guide** states that the claimed or allowed SSH ciphers and keys are supported only when the switch is configured in FIPS mode and with specific additional restrictions configured. Instructions are provided for further configuration to set the minimum supported DH groups and to disable disallowed ciphers and public key algorithms.

Section "Generate SSH Host Keys" in the **Admin Guide** provides instructions for generating an SSH2 host key. SSH2 host keys are used to authenticate connections between the device and clients on remote systems. A host key must be generated before the device can accept incoming SSH connections.

Section "User Key-Based Authentication" in the **Admin Guide** provides instructions for configuring SSH public key

for user authentication.

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.

Test 1 -2: The successful testing of the remote administration channel and the demonstration of its encryption can be found in FCS_SSHS_EXT.1. The evaluator verified that the results from the FCS_SSHS_EXT.1 tests were using the correct protocol and that there was no channel data being sent in plaintext.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

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

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

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

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

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

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

having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

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

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

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

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

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

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

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

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

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

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

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

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

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

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

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

As identified throughout this AAR, the **Admin Guide** provides instructions for configuring the TOE's cryptographic security functions. The **Admin Guide** provides instructions for configuring FIPS mode such that only the cryptographic algorithms and parameters used for the evaluated configuration are available and that it is clear that no other cryptographic engines have been evaluated or tested. There are warnings and notes throughout the **Admin 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 document to use when configuring the TOE:

- Extreme Networks ExtremeXOS Common Criteria Configuration Guide 31.3.100, 9037401-00, Rev AA, October 2022

The document included references to general Extreme Networks manuals which the evaluator could access via web links provided in the "Documentation and Training" section. The completeness of the documentation is addressed by their use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

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

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

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

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

3.3.2 TOE CM Coverage (ALC_CMS.1)

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

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

See section 3.3.1 above for an explanation of how all CM items are addressed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes 'evaluated configuration' instructions). The focus of the testing is to confirm that the requirements specified in Section 5.1.7 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

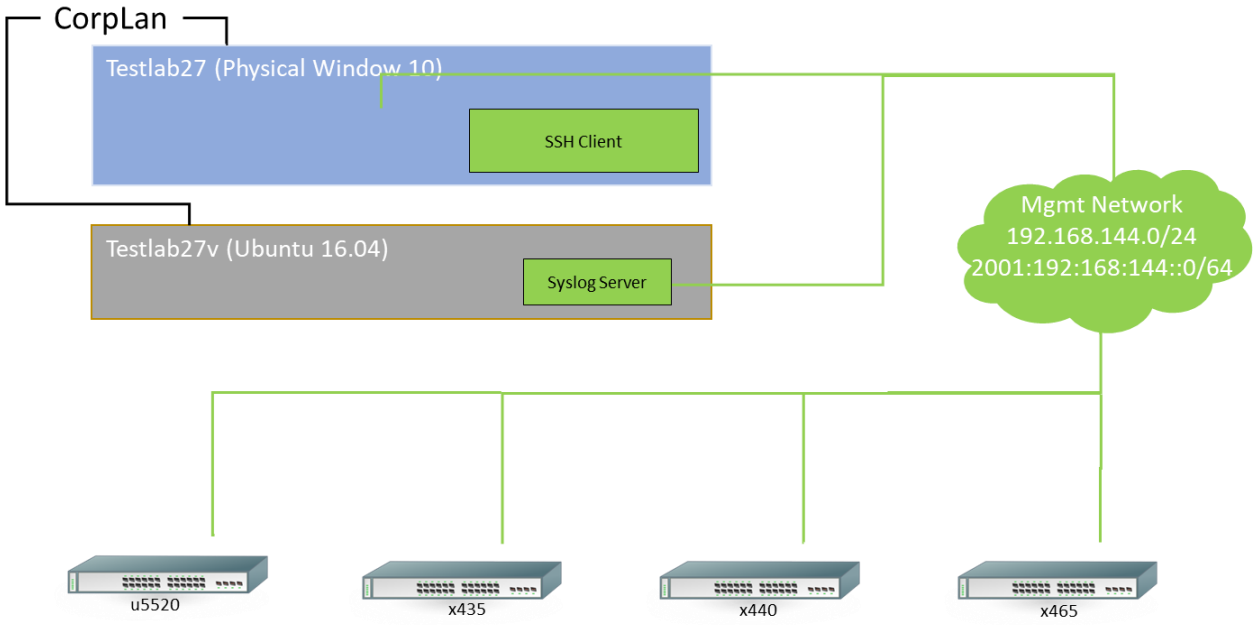
The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

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

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

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section B.4.3.1.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The test configuration consisted of the following TOE platforms along with supporting products.



TOE Platforms:

- u5520 model: 5520-12MW-36W-EXOS running EXOS 31.3.100
- x435 model: X435-8T-4S running EXOS 31.3.100
- x440 model: X440G2-12p-10G4 running EXOS 31.3.100
- x465 model: X465-24S running EXOS 31.3.100

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 3.4.7
- Windows SSH Client – Putty version 0.73 (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 and ntp server.

- Openssl version 1.0.2g
- Openssh client version 7.2p2
- Big Packet Putty, Openssh-client version 6.8p1
- Rsyslog version 8.16.0

- ntpd 4.2.8p4
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Nmap version 7.01
- Stunnel 5.30

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

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

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

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

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

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

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

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

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

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

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

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

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuzz testing. None of the public search for vulnerabilities, or the fuzz testing uncovered any residual vulnerability.

The evaluation team performed a public search for vulnerabilities in order to ensure there are no publicly known and exploitable vulnerabilities in the TOE from the following sources:

- National Vulnerability Database (<https://web.nvd.nist.gov/vuln/search>)
- Vulnerability Notes Database (<http://www.kb.cert.org/vuls/>)
- Rapid7 Vulnerability Database (<https://www.rapid7.com/db/vulnerabilities>)
- Tipping Point Zero Day Initiative (<http://www.zerodayinitiative.com/advisories>)
- Exploit / Vulnerability Search Engine (<http://www.exploitsearch.net>)
- 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 10/06/2022 with the following search terms: "Extreme", "EXOS", "XOS", "TLS", "SSH", "Intel Atom", "Cavium Octeon", "BCM53549", "OpenSSL 2.0.16", "BCM56375", "BCM56376", "BCM56377", "BCM53547", "BCM53548", "Broadcom".

3.5.2 Additional Flaw Hypotheses (AVA_VLA.1)

Assurance Activities: The following additional tests shall be performed:1.) [Conditional]: If the TOE is a TLS server and supports ciphersuites that use RSA transport (e.g. supporting TLS_RSA_WITH_* ciphers) the following test shall be performed. Where RSA Key Establishment schemes are claimed and especially when PKCS#1 v1.5* padding is used, the evaluators shall test for implementation flaws allowing Bleichenbacher and Klima et al. style attacks, including Bock et al's ROBOT attacks of 2017 in the flaw analysis. Even though Bleichenbacher's original paper is two decades old, Bock et al. found these attacks to still be effective in weakening the security of RSA key establishment in current products. Bleichenbacher and Klima et al. style attacks are complex and may be difficult

to detect, but a number of software testing tools have been created to assist in that process. The iTC strongly recommends that at least one of the tools mentioned in Bock et al's ROBOT attacks of 2017 webpage or paper, as effective to detect padding oracle attacks, be used to test TOE communications channels using RSA based Key Establishment (related sources: http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf, https://eprint.iacr.org/2003/052, https://robotattack.org/). Network Device Equivalency Considerations

Not applicable. The TOE is not vulnerable to Bleichenbacher attacks because it does not support a TLS server implementation or RSA key exchange ciphersuites.