



---

www.GossamerSec.com

# ASSURANCE ACTIVITY REPORT FOR CISCO NEXUS 9000 SERIES SWITCHES RUNNING NX-OS 10.4

---

Version 0.2

01/21/25

***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	01/08/25	Keenan	Initial draft
Version 0.2	01/21/25	Cummins	ECR comments addressed.

**The TOE Evaluation was Sponsored by:**

Cisco Systems, Inc.  
170 West Tasman Dr.  
San Jose, CA 95134

**Evaluation Personnel:**

- Allison Keenan
- Cody Cummins

**Common Criteria Versions:**

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

**Common Evaluation Methodology Versions:**

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



## TABLE OF CONTENTS

- 1. Introduction .....6
  - 1.1 Equivalence .....6
    - 1.1.1 Evaluated Platform Equivalence .....6
    - 1.1.2 CAVP Equivalence .....7
  - 1.2 References.....9
- 2. Protection Profile SFR Assurance Activities .....10
  - 2.1 Security audit (FAU) .....10
    - 2.1.1 Audit Data Generation (NDcPP22e:FAU\_GEN.1) .....10
    - 2.1.2 User identity association (NDcPP22e:FAU\_GEN.2).....12
    - 2.1.3 Protected Audit Event Storage (NDcPP22e:FAU\_STG\_EXT.1) .....13
  - 2.2 Cryptographic support (FCS) .....17
    - 2.2.1 Cryptographic Key Generation (NDcPP22e:FCS\_CKM.1) .....17
    - 2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS\_CKM.2) .....22
    - 2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS\_CKM.4).....27
    - 2.2.4 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS\_COP.1/DataEncryption)  
29
    - 2.2.5 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS\_COP.1/Hash).....33
    - 2.2.6 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS\_COP.1/KeyedHash) .....35
    - 2.2.7 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS\_COP.1/SigGen)...37
    - 2.2.8 Random Bit Generation (NDcPP22e:FCS\_RBG\_EXT.1).....39
    - 2.2.9 SSH Server Protocol - per TD0631 (NDcPP22e:FCS\_SSHS\_EXT.1).....41
    - 2.2.10 TLS Client Protocol Without Mutual Authentication - per TD0670 & TD0790  
(NDcPP22e:FCS\_TLSC\_EXT.1).....50
  - 2.3 Identification and authentication (FIA) .....60
    - 2.3.1 Authentication Failure Management (NDcPP22e:FIA\_AFL.1).....60
    - 2.3.2 Password Management - per TD0792 (NDcPP22e:FIA\_PMG\_EXT.1) .....62
    - 2.3.3 Protected Authentication Feedback (NDcPP22e:FIA\_UAU.7) .....64
    - 2.3.4 Password-based Authentication Mechanism (NDcPP22e:FIA\_UAU\_EXT.2).....65
    - 2.3.5 User Identification and Authentication (NDcPP22e:FIA\_UIA\_EXT.1) .....65



- 2.3.6 X.509 Certificate Validation (NDcPP22e:FIA\_X509\_EXT.1/Rev).....68
- 2.3.7 X.509 Certificate Authentication (NDcPP22e:FIA\_X509\_EXT.2) .....74
- 2.4 Security management (FMT).....77
  - 2.4.1 Management of security functions behaviour (NDcPP22e:FMT\_MOF.1/ManualUpdate).....77
  - 2.4.2 Management of TSF Data (NDcPP22e:FMT\_MTD.1/CoreData).....78
  - 2.4.3 Management of TSF Data (NDcPP22e:FMT\_MTD.1/CryptoKeys).....79
  - 2.4.4 Specification of Management Functions - per TD0631 (NDcPP22e:FMT\_SMF.1) .....81
  - 2.4.5 Restrictions on Security Roles (NDcPP22e:FMT\_SMR.2) .....83
- 2.5 Protection of the TSF (FPT) .....84
  - 2.5.1 Protection of Administrator Passwords (NDcPP22e:FPT\_APW\_EXT.1) .....84
  - 2.5.2 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT\_SKP\_EXT.1) .....85
  - 2.5.3 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT\_STM\_EXT.1) .....85
  - 2.5.4 TSF testing (NDcPP22e:FPT\_TST\_EXT.1) .....87
  - 2.5.5 Trusted update (NDcPP22e:FPT\_TUD\_EXT.1).....90
- 2.6 TOE access (FTA) .....96
  - 2.6.1 TSF-initiated Termination (NDcPP22e:FTA\_SSL.3).....96
  - 2.6.2 User-initiated Termination (NDcPP22e:FTA\_SSL.4) .....97
  - 2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA\_SSL\_EXT.1).....98
  - 2.6.4 Default TOE Access Banners (NDcPP22e:FTA\_TAB.1).....99
- 2.7 Trusted path/channels (FTP).....100
  - 2.7.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP\_ITC.1).....100
  - 2.7.2 Trusted Path - per TD0639 (NDcPP22e:FTP\_TRP.1/Admin).....103
- 3. Protection Profile SAR Assurance Activities.....105
  - 3.1 Development (ADV) .....105
    - 3.1.1 Basic Functional Specification (ADV\_FSP.1).....105
  - 3.2 Guidance documents (AGD).....106
    - 3.2.1 Operational User Guidance (AGD\_OPE.1) .....106
    - 3.2.2 Preparative Procedures (AGD\_PRE.1).....108
  - 3.3 Life-cycle support (ALC).....109



---

3.3.1	Labelling of the TOE (ALC_CMC.1) .....	109
3.3.2	TOE CM Coverage (ALC_CMS.1).....	110
3.4	Tests (ATE).....	110
3.4.1	Independent Testing - Conformance (ATE_IND.1).....	110
3.5	Vulnerability assessment (AVA) .....	112
3.5.1	Vulnerability Survey (AVA_VAN.1).....	112
3.5.2	Additional Flaw Hypotheses (AVA_VLA.1) .....	114



## 1. INTRODUCTION

This document presents evaluations results of the Cisco Nexus 9000 Series Switches running NX-OS 10.4 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 is the Nexus 9200, 9300, 9400, 9500, and 9800 series switches running NX-OS 10.4.

- Nexus 9200 models:
  - N9K-C92348GC-X
- Nexus 9300 models:
  - N9K-C93108TC-FX
  - N9K-C9348GC-FXP,
  - N9K-C93216TC-FX2,
  - N9K-C93180YC-FX,
  - N9K-C93240YC-FX2,
  - N9K-C93360YC-FX2,
  - N9K-C9364C,
  - N9K-C9332C,
  - **N9K-C9336C-FX2,**
  - N9K-C9364C-GX,
  - N9K-C9316D-GX,
  - N9K-C93600CD-GX,
  - N9K-C93400LD-H1
- Nexus 9400 models:
  - N9K-C9400-Sup-A
- Nexus 9500 models:
  - N9K-C9504,
  - N9K-C9508,
  - N9K-C9516,
  - N9K-SUP-A+,
  - N9K-SUP-B+,
  - N9K-SC-A
- Nexus 9800 models:
  - N9K-C9808,



- N9K-C9804

The evaluator performed full testing on the following platform (**bolded** in the lists above) and microarchitectures.

- **N9K-C9336C-FX2– Intel Xeon D-1526 (Broadwell)**

Each Nexus 9k device contains an Intel Xeon CPUs running NX-OS v10.4. The hardware models that comprise the TOE have common hardware characteristics and contain similar processors from the Intel Xeon families. The exact same software image is used on all hardware platforms and there are no security relevant functions implemented by the network drivers. Since the same software is installed on all the hardware platforms and all security functions provided by the TOE are implemented in software; the TOE security behavior is the same on all the hardware models for each of the SFRs defined by the NDcPP22E. These SFRs are instantiated by the same version of the TOE software and in the same way on every platform. Any differing characteristics in the hardware affect only non-TSF relevant functionality such as throughput, processing speed, number and type of network connections supported and amount of storage.

There is no NIAP requirement to test all microarchitectures for PP testing. All security features are implemented above the microarchitecture level except entropy and the entropy analysis has been approved. All cryptographic functions have been tested on all microarchitectures so any cryptographic differences have been tested.

### 1.1.2 CAVP EQUIVALENCE

All cryptography is implemented using the CiscoSSL FOM Cryptographic implementation version 7.3a.

The following algorithm implementations have been CAVP certified.

SFR	Selection	Algorithm	Implementation	Certificate #
FCS_CKM.1 - Cryptographic Key Generation	2048	RSA KeyGen	CiscoSSL FOM 7.3a	A4446
	P-256 P-384 P-521	ECDSA KeyGen and KeyVer	CiscoSSL FOM 7.3a	A4446
	3072	FFC KeyGen	CiscoSSL FOM 7.3a	A4446
FCS_CKM.2 – Cryptographic Key Establishment	2048	RSA key establishment	CiscoSSL FOM 7.3a	Testing using a known-good implementaiton.
	P-256 P-384 P-521	KAS-ECC-SSC SP800-56Ar3	CiscoSSL FOM 7.3a	A4446
	3072	KAS-FFC-SSC Sp800-56Ar3	CiscoSSL FOM 7.3a	A4446



SFR	Selection	Algorithm	Implementation	Certificate #
FCS_COP.1/ DataEncryption	AES-CBC-128 AES-CBC-256 AES-CTR-128 AES-CTR-256 AES-GCM-128 AES-GCM-256	AES-CBC Encrypt/Decrypt AES-CTR Encrypt/Decrypt AES-GCM Encrypt/Decrypt	CiscoSSL FOM 7.3a	A4446
FCS_COP.1/ SigGen	2048	RSA SigGen and SigVer	CiscoSSL FOM 7.3a	A4446
	P-256 P-384 P-521	ECDSA SigGen and SigVer	CiscoSSL FOM 7.3a	A4446
FCS_COP.1/ Hash	SHA1 SHA-256 SHA-384 SHA-512	SHS	CiscoSSL FOM 7.3a	A4446
FCS_COP.1/ KeyedHash	HMAC-SHA1 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	HMAC	CiscoSSL FOM 7.3a	A4446
FCS_RBG_EXT.1 – Random Bit Generation	HMAC_DRBG (any) 256 bits	DRBG	CiscoSSL FOM 7.3a	A4446

All algorithms claimed have Cryptographic Algorithm Validation Program (CAVP certificates running on the processors specified in the table below.

Hardware Model	Processor
Cisco Nexus 9200 Series, Cisco Nexus 9300 Series ( <b>93108TC-FX, 93216TC-FX2, 93240YC-FX2, 93360YC-FX2, 9364C, 9332C, 9336C-FX2, 9364C-GX, 9316D-GX, 93600CD-GX</b> ), Cisco Nexus 9500 Series( <b>Supervisor 9500-Sup-A+</b> )	Intel Xeon D-1526 (Broadwell)
Cisco Nexus 9300 Series ( <b>93180YC-FX</b> ) Cisco Nexus 9500 Series ( <b>Supervisor 9500-Sup-B+</b> )	Intel Xeon D-1528 (Broadwell)
Cisco Nexus 9300 Series ( <b>93400LD-H1</b> )	Intel Xeon D-1623N (Broadwell)
Cisco Nexus 9400 Series	Intel Xeon D-1633N (Broadwell)
Cisco Nexus 9800 Series	Intel Xeon D-1530 (Broadwell DE)

All algorithm testing has been performed on all processors with 2 minor exceptions. The HMAC-SHA1 and HMAC-SHA2-512 listing for the Intel Xeon D-1623N and Intel Xeon D-1528 do not have the claimed key size from FCS\_COP.1/KeyedHash (160 bits for HMAC-SHA1 and 512 bits for HMAC-SHA512). These is due to the range of key sizes that were included in the vector set for ACVP testing, but does not indicate that other key sizes are not





supported by the library on these Operating Environment. The 3 remaining processors from the evaluation (Intel Xeon D-1526, Intel Xeon D-1633N, and Intel Xeon D-1530), which are also listed on the algorithm cert, include the described key sizes for the HMAC algorithms. The Intel Xeon D-1526 is the same family and microarchitecture as the Intel Xeon D-1528 and the Intel Xeon D-1633N is the same family and microarchitecture as the Intel Xeon D-1623N. All operating environments are running the same crypto library and all models of the TOE leverage the same TOE image. For these reasons it can be deduced by equivalency that these HMAC key sizes (160 bits for HMAC-SHA1 and 512 bits for HMAC-SHA512) are also supported for the Intel Xeon D-1623N and Intel Xeon D-1528.

## 1.2 REFERENCES

The following evidence was used to complete the Assurance Activities:

- Cisco Nexus 9000 Series Switches running NX-OS 10.4 Security Target, Version 0.9, January 21, 2025 **(ST)**
- Cisco Nexus 9000 Series Switches Running NX-OS 10.4 Common Criteria Operational User Guidance and Preparative Procedures, Version 0.9, January 21, 2025 **(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 (FAU\_GEN.1) in the ST states the TOE generates an audit record that is stored internally within the TOE whenever an audited event occurs. The types of events that cause audit records to be generated include, cryptography related events, events related to the enforcement of information flow policies, identification and authentication related events, and administrative events. Each of the events is specified in the syslog which is stored internal to the TOE in enough detail to identify the user for which the event is associated, when the event occurred, where the event occurred, the outcome of the event, and the type of event that occurred.

For the administrative task of generating/import of, changing, or deleting of cryptographic keys a unique key name is included in the audit record.

The TOE is not distributed.

**Component Guidance Assurance Activities:** The evaluator shall check the guidance documentation and ensure that it provides an example of each auditable event required by FAU\_GEN.1 (i.e. at least one instance of each auditable event, comprising the mandatory, optional and selection-based SFR sections as applicable, shall be provided from the actual audit record).

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

Section “Security Relevant Events” in the **Admin Guide** provides a table identifying all of the required audit events consistent with the ST along with a sample of each record for each SFR where applicable.

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

The TOE is not distributed.

## 2.1.2 USER IDENTITY ASSOCIATION (NDcPP22E:FAU\_GEN.2)

### 2.1.2.1 NDcPP22E:FAU\_GEN.2.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

**Component TSS Assurance Activities:** The TSS and Guidance Documentation requirements for FAU\_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU\_GEN.1.

See FAU\_GEN.1

**Component Guidance Assurance Activities:** The TSS and Guidance Documentation requirements for FAU\_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU\_GEN.1.



See FAU\_GEN.1

**Component Testing Assurance Activities:** This activity should be accomplished in conjunction with the testing of FAU\_GEN.1.1.

For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated for building up a secure channel between two TOE components. If for some reason (could be e.g. TSS or Guidance Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

See FAU\_GEN.1

### **2.1.3 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 in the ST indicates in the ST indicates that the TOE is a standalone device that stores audit data locally. The TOE is not distributed. The TOE can be configured to export syslog records to a specified, external syslog server in real-time. Once the configuration is complete, the audit records are automatically sent to the external syslog server at the same time as they are written to the logging buffer. The TOE protects communications with an external syslog server via TLS. If the TLS connection fails, the TOE will store audit records on the TOE when it



discovers it can no longer communicate with its configured syslog server. When the connection is restored, the TOE will transmit the buffer contents to the syslog server.

By default, the logs are circular and once the log files reach capacity of the flash storage, they are overwritten. With NX-OS, there is logging of event-histories that run in the background by default. The event-history log size is configurable.

The TOE is not distributed.

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

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

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

Section “Remote Logging Configuration” in the **Admin Guide** provides instructions for using the “logging server” command to enable the TOE to transmit audit data to a specified external audit server. This section states that when an audit event is generated, it is simultaneously sent to the external server and the local store.

Section “Local Logging” in the **Admin Guide** provides instructions for how to configure the size of the local logging buffer. This section states that if the local storage space for audit data is full the TOE will overwrite the oldest audit record to make room for the new audit record.

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

a) Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular



software (name, version) used on the audit server during testing. The evaluator shall verify that the TOE is capable of transferring audit data to an external audit server automatically without administrator intervention.

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

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

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

3) The TOE behaves as specified (for the option 'other action' in FAU\_STG\_EXT.1.3).

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

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

Test 1: The successful establishment of the TLS syslog connection for all TOE devices was demonstrated in FTP\_ITC.1. In each case the TOE initiated the connection without administrator intervention. The use of TLS ensured that no audits were viewed in cleartext. The audits collected as part of FAU\_GEN.1 throughout testing were gathered from the remote syslog server running rsyslog version 8.16.0 thus demonstrating that audits were successfully received by the remote syslog server.

Test 2: The option “overwrite previous audit records” is selected in FAU\_STG\_EXT.1.3. The logs are stored in a local buffer. The evaluator issued the “show logging” command and viewed the contents of the TOE’s log buffer which was at full capacity. The evaluator then performed other commands to generate audit activity and issued the “show logging” command again. The evaluator observed that the new audit records were generated and verified that the oldest audit records had been overwritten.

Test 3: Not applicable. The TOE does not claim support for 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 6.1 (FCS\_CKM.1) in the ST states the following table describes the key generation algorithms the TOE implements to generate asymmetric keys used for device authentication:

Scheme	Standard	Key Size/ NIST Curve	SFR	Service
RSA	FIPS PUB 186-4	2048	FCS_SSHS_EXT.1	SSH Remote Administration
ECC	FIPS PUB 186-4	P-256 P-384 P-521	FCS_SSHS_EXT.1	SSH Remote Administration

The following table shows the key generation algorithms implemented by the TOE to generate asymmetric keys used for key establishment:



Scheme	Standard	Key Size/ NIST Curve	SFR	Service
RSA	FIPS PUB 186-4	2048	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
ECC	FIPS PUB 186-4	P-256	FCS_SSHS_EXT.1	SSH Remote Administration
		P-384 P-521	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
FFC	FIPS PUB 186-4	3072	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server

The following table shows the methods that the TOE implements for key establishment:

Scheme	Standard	Key Size/ NIST Curve	SFR	Service
RSA	FIPS PUB 186-4	2048	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
ECC	FIPS PUB 186-4	P-256	FCS_SSHS_EXT.1	SSH Remote Administration
		P-384 P-521	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server



FFC	FIPS PUB 186-4	3072	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
-----	-------------------	------	----------------	--

The TOE implements RSA key establishment schemes that are conformant to NIST SP 800-56B. The TOE complies with section 6 and all subsections regarding RSA key pair generation and key establishment in the NIST SP 800-56B. The TOE implements Elliptic Curve Diffie-Hellman (ECDH) (ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521) key establishment schemes in SSH. The ECDH key generation meets the NIST FIPS PUB 186-4 Appendix B.1. ECC schemes are used with P-256, P-384, and P-521.

The TOE also supports FCC schemes for key establishment with a minimum size of 3072 bits.

The key pair generation portions of "The RSA Validation System" for FIPS 186-4 were used as a guide in testing the FCS\_CKM.1 during the FIPS validation.

The TOE employs RSA-based key establishment used in cryptographic operations with support for 2048-bit keys.

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

Section "Enabling FIPS Mode" in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section "Remote Administration Protocols" in the **Admin Guide** provides instructions for generating an RSA or ECDSA host key for SSH, configuring the SSH server key exchange algorithm, configuring the host key and user public key algorithms and configuring ssh rekey settings.

**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  $p_1, p_2, q_1, q_2, p$  and  $q$  shall all be provable primes
- Primes  $p_1, p_2, q_1$ , and  $q_2$  shall be provable primes and  $p$  and  $q$  shall be probable primes
- Primes  $p_1, p_2, q_1, q_2, p$  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  $x$ :

- $\text{len}(q)$  bit output of RBG where  $1 \leq x \leq q-1$
- $\text{len}(q) + 64$  bit output of RBG, followed by a mod  $q-1$  operation and a  $+1$  operation, where  $1 \leq x \leq q-1$ .

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

- $g \neq 0,1$
- $q$  divides  $p-1$
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

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 6.1 (FCS\_CKM.2) in the ST states the following table describes the key generation algorithms the TOE implements to generate asymmetric keys used for device authentication:

Scheme	Standard	Key Size/ NIST Curve	SFR	Service
RSA	FIPS PUB 186-4	2048	FCS_SSHS_EXT.1	SSH Remote Administration
ECC	FIPS PUB 186-4	P-256 P-384 P-521	FCS_SSHS_EXT.1	SSH Remote Administration

The following table shows the key generation algorithms implemented by the TOE to generate asymmetric keys used for key establishment:

Scheme	Standard	Key Size/ NIST Curve	SFR	Service
RSA	FIPS PUB 186-4	2048	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
ECC	FIPS PUB 186-4	P-256 P-384 P-521	FCS_SSHS_EXT.1	SSH Remote Administration
			FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
FFC	FIPS PUB 186-4	3072	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server



The following table shows the methods that the TOE implements for key establishment:

Scheme	Standard	Key Size/ NIST Curve	SFR	Service
RSA	FIPS PUB 186-4	2048	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
ECC	FIPS PUB 186-4	P-256	FCS_SSHS_EXT.1	SSH Remote Administration
		P-384	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server
P-521				
FFC	FIPS PUB 186-4	3072	FCS_TLSC_EXT.1	Transmit audit data to an external syslog server

The TOE implements RSA key establishment schemes that are conformant to NIST SP 800-56B. The TOE complies with section 6 and all subsections regarding RSA key pair generation and key establishment in the NIST SP 800-56B. The TOE implements Elliptic Curve Diffie-Hellman (ECDH) (ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521) key establishment schemes in SSH. The ECDH key generation meets the NIST FIPS PUB 186-4 Appendix B.1. ECC schemes are used with P-256, P-384, and P-521.

The TOE also supports FCC schemes for key establishment with a minimum size of 3072 bits.

The key pair generation portions of "The RSA Validation System" for FIPS 186-4 were used as a guide in testing the FCS\_CKM.1 during the FIPS validation.

The TOE employs RSA-based key establishment used in cryptographic operations with support for 2048-bit keys.

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





Section “Enabling FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Administration Protocols” in the **Admin Guide** provides instructions for generating an RSA or ECDSA host key for SSH, configuring the SSH server key exchange algorithm, configuring the host key and user public key algorithms and configuring ssh rekey settings.

#### **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. Gossamer tested using a known implementation (stunnel leveraging OpenSSL) for RSA key establishment, to ensure correctness for the TOE's use of RSA in its TLS implementation.



## 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 for<sup>2</sup>). 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 (FCS\_CKM.4) in the ST indicates that the TOE meets all requirements specified in FIPS 140-2 for destruction of keys and Critical Security Parameters (CSPs) when no longer required for use. The TSS does not identify any configurations or circumstances that may not conform to the key destruction requirement.

Section 6.2 in the ST provides a table which further describes that the TOE zeroizes all secrets, keys, and associated values when they are no longer required. The table identifies all the relevant secret and private keys, their storage location and how and when they are zeroized.

**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 6 (FCS\_CKM.4) in the ST indicates that the TOE meets all requirements specified in FIPS 140-2 for destruction of keys and Critical Security Parameters (CSPs) when no longer required for use. The TSS does not identify any configurations or circumstances that may not conform to the key destruction requirement.

Section 6.2 in the ST provides a table which further describes that the TOE zeroizes all secrets, keys, and associated values when they are no longer required. The table identifies all the relevant secret and private keys, their storage location and how and when they are zeroized.

**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 (FCS\_COP.1/DataEncryption) in the ST states that the TOE provides symmetric encryption and decryption capabilities using AES in CBC, CTR and GCM mode (128 and 256 bits) as described in ISO/IEC 18033-3, ISO/IEC 10116, and ISO/IEC 19772. AES is implemented in the SSH and TLS protocols.

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

Section “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Logging Configuration” describes the supported TLS ciphers and states they are hard-set by FIPS mode.

Section “Remote Administration Protocols” in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH.

**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  $1 < i \leq 10$ . 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  $1 < i \leq 10$ . 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<sub>i</sub> 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 key sizes 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  $1 \text{ less-than } i \text{ less-than-or-equal to } 10$  (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:

$CT[i] = \text{AES-ECB-Encrypt}(\text{Key}, \text{PT})$  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 (FCS\_COP.1/Hash) if the ST states the TOE provides cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512 as specified in ISO/IEC 10118-3:2004.

Through the implementation of the FIPS validated cryptographic module, the TOE provides Secure Hash Standard (SHS) hashing in support of TLS and SSH for secure communications. Management of the cryptographic algorithms is provided through the CLI with auditing of those commands.

**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 “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Logging Configuration” describes the supported TLS ciphers and states they are hard-set by FIPS mode.

Section “Remote Administration Protocols” in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH.

**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  $i$ th message is  $m + 99*i$ , where  $1 \leq i \leq m$ . 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  $i$ th message is  $m + 8*99*i$ , where  $1 \leq i \leq m/8$ . 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 6.1 (FCS\_COP.1/KeyedHash) in the ST states the TOE provides keyed-hashing message authentication services using HMAC-SHA-1, HMAC-SHA-256 that operates on 512-bit blocks, HMAC-SHA2-384 that operates on 1024-bit blocks, and HMAC-SHA-512 that operates on 1024-bit blocks of data with key sizes and message digest sizes of 160, 256, 384, 512 bits, respectively as specified in ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”

Through the implementation of the FIPS validated cryptographic module, the TOE provides SHS hashing and HMAC message authentication in support of SSHv2, and TLSv1.2 for secure communications. Management of the cryptographic algorithms is provided through the CLI with auditing of those commands.

SHS hashing and HMAC message authentication is used in the establishment of TLS, and SSHv2 sessions.

Section 6 (FCS\_SSHS\_EXT.1) in the ST indicates the TOE implementation of SSHv2 supports the following:

- public key algorithms for authentication as described in RFC 4252
- password-based authentication for administrative users exporting data through SSHv2.
- encryption algorithms, aes128-cbc, aes256-cbc, aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com to ensure confidentiality of the session.
- hashing algorithms hmac-sha1, hmac-sha2-256, hmac-sha2-384, and hmac-sha2-512 to ensure the integrity of the session.
- SSH transport implementation public key algorithms: ssh-rsa, ecdsa-sha2-nistp256, and ecdsa-sha2-nistp384.

**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 “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Logging Configuration” describes the supported TLS ciphers and states they are hard-set by FIPS mode.

Section “Remote Administration Protocols” in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH.



**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 6.1 (FCS\_COP.1/SigGen) in the ST states The TOE provides cryptographic signature services using the following:

- RSA Digital Signature Algorithm with key size of 2048 and greater as specified in FIPS PUB 186-4, “Digital Signature Standard”.
- Elliptic Curve Digital Signature Algorithm and cryptographic key sizes 256 bits, 384 bits, 521 bits.

Through the implementation of the FIPS validated cryptographic module, the TOE provides cryptographic signatures in support of SSHv2 for secure communications. Management of the cryptographic algorithms is provided through the CLI with auditing of those commands. For SSHv2, RSA and ECDSA host keys are supported.

In addition, the TOE provides cryptographic signature verification of RSA and X509 certificates used for TLS authentication by the external syslog server.

**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 “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Logging Configuration” describes the supported TLS ciphers and states they are hard-set by FIPS mode.

Section “Remote Administration Protocols” in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH.

#### **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 RANDOM BIT GENERATION (NDcPP22E:FCS\_RBG\_EXT.1)

### 2.2.8.1 NDcPP22E:FCS\_RBG\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.2.8.2 NDcPP22E:FCS\_RBG\_EXT.1.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

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

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

Section 6 (FCS\_RBG\_EXT.1) of the ST states 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 (FCS\_RBG\_EXT.1) in the ST states that the TOE implements an HMAC\_DRBG (as defined in section 10.1.2 of NIST SP 800-90A (6), using HMAC w/SHA-256) from CiscoSSL's FIPS Object Module (FOM) to generate its cryptographic keys.

**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) document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section "FIPS Mode" in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

**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  $\leq$  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.

## 2.2.9 SSH SERVER PROTOCOL - PER TD0631 (NDcPP22E:FCS\_SSHS\_EXT.1)

### 2.2.9.1 NDcPP22E:FCS\_SSHS\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.2.9.2 NDcPP22E:FCS\_SSHS\_EXT.1.2

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

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

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

Section 6 (FCS\_SSHS\_EXT.1) of the ST states the TOE implements SSHv2 (telnet is disabled in the evaluated configuration). SSHv2 is implemented according to the following RFCs: 4251, 4252, 4253, 4254, 4344, 5656, 6668, 8308 section 3.1, and 8332. The TOE supports both public key-based and password-based authentication.

SSHv2 connections will be dropped if the TOE receives a packet larger than 262,149 bytes. Large packets are detected by the SSH implementation and dropped internal to the SSH process. The key exchange methods used by the TOE is a configurable option but rsa-sha2-256, ecdh-sha2-nistp256, ecdh-sha2-nistp384, and ecdh-sha2-nistp521 are the only allowed methods within the evaluated configuration. Any session where the SSH client offers only non-compliant algorithms or key sizes per the NDcPPv2.2e will be rejected by the SSH server. SSH sessions can only be established when compliant algorithms and key sizes can be negotiated.



The TOE implementation of SSHv2 supports the following:

- public key algorithms for authentication as described in RFC 4252
- password-based authentication for administrative users accessing the TOE's CLI through SSHv2.
- encryption algorithms aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com to ensure confidentiality of the session.
- hashing algorithms hmac-sha2-256 and hmac-sha2-512 to ensure the integrity of the session.
- SSH transport implementation public key algorithms: rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521.

**Guidance Assurance Activities:** None Defined

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

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

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

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

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

(TD0631 applied)

Test 1: The TOE supports rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 for client public key authentication. The evaluator generated each of the key pairs on the test server and then configured an admin user on the TOE with the corresponding public key. The evaluator then attempted to login to the TOE using the supported public keys and observed that the login was successful.



Test 2: The evaluator next demonstrated an unsuccessful login using an unrecognized public key. The TOE claims support for rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 public keys only.

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 was performed as part of Test 3.

### 2.2.9.3 NDcPP22E:FCS\_SSHS\_EXT.1.3

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

Section 6. (FCS\_SSHS\_EXT.1) in the ST states that SSHv2 connections will be dropped if the TOE receives a packet larger than 262,149 bytes. Large packets are detected by the SSHv2 implementation and dropped internal to the SSH process.

**Guidance Assurance Activities:** None Defined

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

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

### 2.2.9.4 NDcPP22E:FCS\_SSHS\_EXT.1.4

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

Section 6 (FCS\_SSHS\_EXT.1) of the ST states that the TOE implementation of SSHv2 supports the following:

- public key algorithms for authentication as described in RFC 4252
- password-based authentication for administrative users accessing the TOE's CLI through SSHv2.



- encryption algorithms aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com to ensure confidentiality of the session.
- hashing algorithms hmac-sha2-256 and hmac-sha2-512 to ensure the integrity of the session.
- SSH transport implementation public key algorithms: rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521.

The key exchange methods used by the TOE is a configurable option but ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521 are the only allowed methods within the evaluated configuration.

**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 “Remote Administration Protocols” in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH.

**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 connected to the TOE using an SSH client alternately using the claimed encryption algorithms (aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com). The evaluator confirmed that the supported algorithms resulted in successful connections. In each case the evaluator viewed the Server: Key Exchange Init packet and saw that no additional ciphers beyond those that were claimed were seen as supported.

#### **2.2.9.5 NDcPP22E:FCS\_SSHS\_EXT.1.5**

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



Section 6 (FCS\_SSHS\_EXT.1) of the ST states that the TOE implementation of SSHv2 supports the following:

- public key algorithms for authentication as described in RFC 4252
- password-based authentication for administrative users accessing the TOE's CLI through SSHv2.
- encryption algorithms aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com to ensure confidentiality of the session.
- hashing algorithms hmac-sha2-256 and hmac-sha2-512 to ensure the integrity of the session.
- SSH transport implementation public key algorithms: rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521.

This aligns with the SFR claims.

**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 “Remote Administration Protocols” in the **Admin Guide** provides instructions for generating an RSA or ECDSA host key for SSH, configuring the SSH server key exchange algorithm, configuring the host key and user public key algorithms and configuring ssh rekey settings.

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

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

Has effectively been moved to FCS\_SSHS\_EXT.1.2.

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

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

(TD0631 applied)



Test 1: The evaluator established an SSH connection with the TOE using each claimed host key algorithm (rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521). The connection was successful.

Test 2: The evaluator attempted to connect to the TOE using a host public key algorithm that is not included in the ST selection and observed that the connection failed.

### 2.2.9.6 NDcPP22E:FCS\_SSHS\_EXT.1.6

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

Section 6 (FCS\_SSHS\_EXT.1) of the ST states that the TOE implementation of SSHv2 supports the following:

- public key algorithms for authentication as described in RFC 4252
- password-based authentication for administrative users accessing the TOE's CLI through SSHv2.
- encryption algorithms aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com to ensure confidentiality of the session.
- hashing algorithms hmac-sha2-256 and hmac-sha2-512 to ensure the integrity of the session.
- SSH transport implementation public key algorithms: rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521.

**Guidance Assurance Activities:** The evaluator shall also check the guidance documentation to ensure that it contains instructions 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 "Remote Administration Protocols" in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH.

**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 established an SSH connection with the TOE using each of the claimed integrity algorithms (hmac-sha2-256 and hmac-sha2-512 and implicit from GCM algorithms). The evaluator observed a successful connection using each claimed integrity algorithm.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using the HMAC-MD5 algorithm. The connection attempt failed.

### **2.2.9.7 NDcPP22E:FCS\_SSHS\_EXT.1.7**

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

Section 6 (FCS\_SSHS\_EXT.1) of the ST states that the TOE implementation of SSHv2 supports the following:

- public key algorithms for authentication as described in RFC 4252
- password-based authentication for administrative users accessing the TOE's CLI through SSHv2.
- encryption algorithms aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com to ensure confidentiality of the session.
- hashing algorithms hmac-sha2-256 and hmac-sha2-512 to ensure the integrity of the session.
- SSH transport implementation public key algorithms: rsa-sha2-256, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521.

The key exchange methods used by the TOE is a configurable option but ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521 are the only allowed methods within the evaluated configuration.

**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 “Remote Administration Protocols” in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH.



**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: Performed as part of test 2 where the evaluator attempted to establish an SSH connection with the TOE using diffiehellman-group1-sha1 key exchange. The connection attempt failed.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using each allowed key exchange method: ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521. The connection succeeded.

### 2.2.9.8 NDcPP22E:FCS\_SSHS\_EXT.1.8

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

- a) Both thresholds are checked by the TOE.
- b) Rekeying is performed upon reaching the threshold that is hit first.

Section 6 (FCS\_SSHS\_EXT.1) of the ST states that Remote CLI SSHv2 sessions are limited to an administrator configurable session timeout period and used for a threshold of no longer than one hour, and no more than one gigabyte of transmitted data. After either of the thresholds are reached a rekey needs to be performed.

**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 “Remote Administration Protocols” in the **Admin Guide** provides instructions for configuring SSH Rekey Limits. The TOE will react to the first threshold reached.

**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 100 MB. The evaluator attempted to connect to the TOE using a SSH client sending data and confirmed that a rekey happened at the 100 MB threshold. Next the evaluator configured the rekey time limit to 1 hour. 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



## 2.2.10 TLS CLIENT PROTOCOL WITHOUT MUTUAL AUTHENTICATION - PER TD0670 & TD0790 (NDcPP22E:FCS\_TLSC\_EXT.1)

### 2.2.10.1 NDcPP22E:FCS\_TLSC\_EXT.1.1

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

Section 6.1 (FCS\_TLSC\_EXT.1) of the ST states, the supported ciphersuites including the following:

- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA as defined in RFC 3268
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 3268
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA as defined in RFC 4492,
- TLS\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246
- TLS\_RSA\_WITH\_AES\_256\_CBC\_SHA256 as defined in RFC 5246
- TLS\_RSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5288
- TLS\_RSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5288
- TLS\_DHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5246
- TLS\_DHE\_RSA\_WITH\_AES\_256\_CBC\_SHA256 as defined in RFC 5246
- TLS\_DHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5288
- TLS\_DHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5288
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289
- TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289,



TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289,

TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 as defined in RFC 5289

TLS\_ECDHE\_RSA\_WITH\_AES\_256\_GCM\_SHA384 as defined in RFC 5289

TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256 as defined in RFC 5289,

TLS\_ECDHE\_RSA\_WITH\_AES\_256\_CBC\_SHA384 as defined in RFC 5289

The following NIST curves are supported by default on the TOE: secp256r1, secp384r1, and secp521r1. No administrator configuration is required in order to use these curves.

The TOE will not establish SSL 2.0, SSL 3.0 and TLS v1.0 or TLS v1.1 connections if offered by the server. In addition, the TOE will only establish a connection if the peer presents a valid certificate during the handshake.

Any session where the client offers the following in the client hello: SSL 2.0, SSL 3.0, TLS 1.0 and TLS 1.1 will be rejected by the TLS server.

**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 “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section “Remote Logging Configuration” in the **Admin Guide** provides instructions for using the “logging server” command to enable the TOE to transmit audit data to a specified external audit server using TLS.

**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 only one of the claimed ciphersuites at a time. The evaluator used a network sniffer to



capture the TLS session negotiation. The evaluator examined each traffic capture and observed that the expected TLS cipher was negotiated.

Test 2: The evaluator first connected the TOE to a TLS server with a valid certificate. The evaluator observed via packet capture that the connection was successful. Next, the evaluator connected the TOE to a TLS server that had an invalid certificate missing the serverAuth key usage. The packet capture indicated that the connection was rejected by the TOE.

Test 3: The evaluator sent a server certificate in the TLS connection that did not match the server-selected ciphersuite. The evaluator observed via packet capture that the connection was rejected.

Test 4a: The evaluator configured a test server to accept only the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite. The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the ciphersuite and disconnected the session.

Test 4b: The evaluator configured a test server to send a ciphersuite not presented in the client hello. The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid ciphersuite and disconnected the session.

Test 4c: As the TOE supports DHE ciphers the evaluator configured a test server to perform a DHE key exchange in the TLS connection using a non-supported group (dh1096). The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid DHE key exchange and disconnected the session.

Test 5a: The evaluator configured a test server to send an invalid TLS version (1.4). The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid TLS version and disconnected the session.

Test 5b: As the TOE supports DHE ciphers the evaluator configured a test server to modify the signature block in the Server's Key Exchange handshake message. The evaluator then attempted to establish a TLS session from the TOE to that test server. The packet capture indicated that the TOE rejected the invalid handshake and disconnected the session.

Test 6: The evaluator connected the TOE to a TLS server which modified traffic according to the scenarios identified above for this test case. In each case, the TOE successfully detected the modifications and rejected the connection to the TOE.

### 2.2.10.2 NDcPP22E:FCS\_TLSC\_EXT.1.2

**TSS Assurance Activities:** The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of



reference identifiers are supported (e.g. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported.

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

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

Section 6.1 (FCS\_TLSC\_EXT.1) of the ST states When establishing a TLS connection, the TOE supports reference identifiers of type DNS-ID and IP address and will seek a match to the DNS domain name or IP address respectively in the subjectAltName extension. If the TOE determines there is a mismatch in the presented identifier, it will not establish the TLS trusted channel connection. The TOE does not support the use of wildcards within certificates and does not support certificate pinning.

**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 "Remote Logging Configuration" describes how to configure the remote syslog server. The "name" field of the "logging server" command is used to define the syslog server which is also the expected reference identifier. When establishing a TLS connection, the TOE supports reference identifiers of type DNS-ID and IP address and will seek a match to the DNS domain name or IP address respectively in the subjectAltName extension. If the TOE determines there is a mismatch in the presented identifier, it will not establish the TLS trusted channel connection. The TOE does not support the use of wildcards within certificates and does not support certificate pinning.



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

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

or

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

or

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

Note that for some tests additional conditions apply.

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

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

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

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

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

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

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



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

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

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

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

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

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

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

Test 6: [conditional] If IP address identifiers are supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (\*) (e.g. CN=\*.168.0.1 when connecting to 192.168.1.20, CN=2001:0DB8:0000:0000:0008:0800:200C:\* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

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





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

(TD0790 applied, supersedes TD0634)

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

- 1) The evaluator shall present a server certificate that does not contain an identifier in the Subject (DN) attribute type(s) that matches the reference identifier. The evaluator shall verify that the connection fails.
- 2) The evaluator shall present a server certificate that contains a valid identifier as an attribute type other than the expected attribute type (e.g. if the TOE is configured to expect id-atserialNumber=correct\_identifier, the certificate could instead include id-at-name=correct\_identifier), and does not contain the SAN extension. The evaluator shall verify that the connection fails. Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass this test.
- 3) The evaluator shall present a server certificate that contains a Subject attribute type that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.
- 4) The evaluator shall confirm that all use of wildcards results in connection failure regardless of whether the wildcards are used in the left or right side of the presented identifier. (Remark: Use of wildcards is not addressed within RFC 5280.)

The test results show TOE support for each reference identifier type. The TOE can be configured to identify the following:

- An IP value that must be in a certificate's SAN IP field CN field,
- a DNS-id value that must be in the certificate's SAN DNS field or CN field

The TOE accepts certificates using wildcards for DNS names that may be contained either in a SAN DNS field or a CN field. The TOE does not match wildcards in certificates to IP address values.

Tests 1-4 below were repeated for both DNS and IP values.

Test 1: The evaluator established a TLS session from the TOE targeting a server using a valid certificate with a CN matching the domain name used by the client. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed a successful connection. The evaluator then established a TLS



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

Test 2: The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

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

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

Test 5: The evaluator tested hostname wildcards by configuring an expected DNS on the TOE with the test server’s certificates configured with wildcard DNS names. The TOE does not support the use of wildcards in certificates. The TOE rejected all certificates that included hostname wildcards.

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

Test 6: The TOE does not support IP addresses in the CN. The evaluator established a TLS session from the TOE targeting a server using a server certificate that contains a CN that contains an IP address with a wildcard. Using a network sniffer to capture the TLS session negotiation, the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 7: The TOE does not support FPT\_ITT.1



### 2.2.10.3 NDCPP22E:FCS\_TLSC\_EXT.1.3

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

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

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

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

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

Test 1 and 2 were performed as part of testing of FIA\_X509\_EXT.1.1/Rev Test 1. Test 3 is not applicable as the TOE does not offer administrative override of certificate validation failures.

### 2.2.10.4 NDCPP22E:FCS\_TLSC\_EXT.1.4

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

Section 6.1 (FCS\_TLSC\_EXT.1) of the ST states the following NIST curves are supported by default on the TOE: secp256r1, secp384r1, and secp521r1. No administrator configuration is required in order to use these curves.

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



Section 6.1 (FCS\_TLSC\_EXT.1) of the ST indicated that no administrator configuration is required in order to use the supported curves.

**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/DHE 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 supported curves.

- 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

## 2.3 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).

The TOE enforces a timed lockout after an Administrator defined number of unsuccessful password attempts is exceeded. TOE supports setting the limit to any number in the range from 1-65535 failed login attempts within an admin-defined time period of 1-65535 seconds, and to lock the account for an admin-defined number of period of 1-65535 seconds.

For example, an authorized administrator can be blocked from a range of 1 to 65535 seconds when 3 login attempts fail within a period of 60 seconds.

Additionally, a locked account can be unlocked by another authorized administrator using the command:

```
clear aaa local user blocked {username username | all}
```

**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 "Authentication failure" of the Admin Guide how to configure a lockout policy. This policy will lockout users for an administrator configured amount of time after an administrator defined number of failed consecutive login attempts. Only SSH protocol is supported for configuration of user lockout attributes. Administrator lockouts are not applicable to the local console. Local administrators cannot be locked out and have the ability to unlock other users by using the local console.



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

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

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

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

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

The evaluator configured the number of unsuccessful authentication attempts to three with a lockout time of 1 minute. The evaluator performed three login attempts with an incorrect password and verified the attempts failed and verified the account was locked. After 1 minute the evaluator attempted to authenticate with the correct password and was able to log in successfully. The evaluator then repeated the test with the device configured for 5 unsuccessful authentication attempts and a lockout timer of 3 minutes. The evaluator verified the account was locked after the configured limit of failed attempts and after the lockout timer expired the evaluator verified the account was able to log in successfully. The evaluator then issued three incorrect passwords once again, verified the account was locked, and unlocked the offending user through a CLI command on an administrator account. The evaluator performed a login to verify that this administrative action unlocked the account.

### **2.3.2 PASSWORD MANAGEMENT - PER TD0792 (NDcPP22E:FIA\_PMG\_EXT.1)**

#### **2.3.2.1 NDcPP22E:FIA\_PMG\_EXT.1.1**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined



**Testing Assurance Activities:** None Defined

**Component TSS Assurance Activities:** The evaluator shall check that the TSS lists the supported special character(s) for the composition of administrator passwords.

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

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

(TD0792 applied)

Section 6 (FIA\_PMG\_EXT.1) of the ST states the TOE supports the configuration of passwords to be composed of any combination of upper- and lower-case letters, numbers, and special characters that include: “@”, “\$”, “%”, “^”, “&”, “\*”, “(”, “)”, “+”, “=”, “\_”, “-”, “\”, and “.”. Minimum password length is settable by the Authorized Administrator and can be configured for minimum password lengths of 15 characters.

**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 “Password Length” of the Admin Guide describes how to configure passwords and the password length. To prevent administrators from choosing insecure passwords, each password must be at least 8 characters long and can be a maximum of 127 characters. The TOE allows for passwords to be composed of any combination of upper- and lower-case letters, numbers, and the following special characters: “@”, “\$”, “%”, “^”, “&”, “\*”, “(”, “)”, “+”, “=”, “\_”, “-”, “\”, and “.”. A password strength check can be enabled as well as a minimum password length.

A strong password has the following characteristics:

- Is at least eight characters long
- Does not contain many consecutive characters (such as abcd)
- Does not contain many repeating characters (such as aaabbb)
- Does not contain dictionary words
- Does not contain proper names



- Contains both uppercase and lowercase characters
- Contains numbers

**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 set of characters. The testing included testing a password of at least 15 characters in length.

### **2.3.3 PROTECTED AUTHENTICATION FEEDBACK (NDcPP22E:FIA\_UAU.7)**

#### **2.3.3.1 NDcPP22E:FIA\_UAU.7.1**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

**Component TSS Assurance Activities:** None Defined

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

There are no necessary preparatory steps to ensure authentication data is not revealed while entering as this occurs by default.





**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- This test was performed as part of the tests for FIA\_UIA\_EXT.1 where the evaluator observed that passwords are obscured on the console logins.

### **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 NDcPP22e:FIA\_UIA\_EXT.1.

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

See NDcPP22e:FIA\_UIA\_EXT.1.

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

See NDcPP22e: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 (FIA\_UIA\_EXT.1) of the ST states the TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed. Administrative access to the TOE is facilitated through the TOE's CLI. The TOE mediates all administrative actions through the CLI. Once a potential administrative user attempts to access the CLI of the TOE through either a directly connected console or remotely through an SSHv2 connection, the TOE prompts the user for a user name and password. SSHv2 connections also allow for public key authentication. Only after the administrative user presents the correct authentication credentials will access to the



TOE administrative functionality be granted. No access is allowed to the administrative functionality of the TOE until an administrator is successfully identified and authenticated.

Authentication may be provided by:

- Authentication against a local database.

**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 “Initial Setup via Direct Console Connection” of the admin guide describes that The console port is an asynchronous serial port that allows you to connect to the device for initial configuration through a standard RS-232 port with an RJ-45 connector. Any device connected to this port must be capable of asynchronous transmission. Default parameters for your terminal emulator must be configured with the following:

*9600 baud, 8 data bits, 1 stop bit, and no parity.*

Section “Administrator Configuration and Credentials” provides details on how to assign credentials while creating new users.

Section “Remote Administration Protocols” describes how to configure and enable SSH for remote authentication. It describes how to configure a public key for a user for SSH public key authentication. Both local password-based and public key-based authentication may be enabled on the TOE at the same time.

Section “Identification and Authentication” provides examples to demonstrate how to successfully log on to the TOE using the created credentials.

**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. The evaluator also performed an nmap port scan and verified that there were no other services available, nor any configuration options offered to administrators otherwise, 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 TOE's 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 & 1b -- For this test, the evaluator alternately configured the TOE to have and then not have the trusted root CA used by stunnel on the test server to anchor all of its certificates. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server expecting the connection to succeed only when the trusted root CA was properly configured forming a valid certificate chain.

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 using TLS. The test server then presented a certificate during the protocol negotiation where the certificate was valid. A packet capture was obtained of these protocol negotiations 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 - The evaluator configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where the ECDSA certificates have a supported named curve. The TOE accepted the certificate as valid.

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

Test 8c - The use of ECDSA certificates in the truststore is outside the scope of the evaluated configuration, thus the test is not applicable. This is backed by the ST and AGD.

### **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: For this test, the evaluator alternately configured a test peer to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and observed that the connection was rejected in each case.

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

**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.





The TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS connections. The certificate validation checking takes place during the TLS session setup.

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

- Manual cut-and-paste - ESA displays the Certificate Request Message via the GUI or CLI interface. This allows the administrator to copy the certificate request and in a secure offline manner send the request to a Certification Authority to be transformed into an X.509v3 public-key certificate.
- Both the certificate request message and the certificates themselves provide protection in that they are digitally signed. If a certificate is modified in any way, it would be invalidated. The digital signature verification process would show that the certificate had been tampered with when the hash value would be invalid.
- The certificate chain establishes a sequence of trusted certificates, from a peer certificate to the root CA certificate. Within the PKI hierarchy, all enrolled peers can validate the certificate of one another if the peers share a trusted root CA certificate or a common subordinate CA. When a certificate chain is received from a peer, the default processing of a certificate chain path continues until the first trusted certificate is reached.
- The authorized administrator can also configure one or more certificate fields together with the  
ir matching criteria to match. Such as:
  - o alt-subject-name
  - o issuer-name
  - o name
  - o serial-number
  - o subject-name

The physical security of the TOE (A.PHYSICAL\_PROTECTION) protects the switch and the certificates from being tampered with or deleted. In addition, the TOE identification and authentication security functions protect an unauthorized user from gaining access to the TOE.

OCSP is used for certificate revocation. The TOE will act accordingly based on the Authority Info Access extension in the CA or Peer certificate.

Checking is also done for the basicConstraints extension and the CA flag to determine whether they are present and set to TRUE. The local certificate that was imported must contain the basic constraints extension with the CA flag set to true, the check also ensure that the key usage extension is present, and the keyEncipherment bit or the keyAgreement bit or both are set. If they are not, the certificate is not accepted.



If the connection to determine the certificate validity cannot be established, the certificate is not accepted.

**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 Certificates” of the Admin Guide states X.509v3 certificates are being used for authentication of the TLS channel between the Nexus 9K Switch and the syslog server.

Section “Remote Logging Configuration” The TOE supports peer certificate verification against a locally trusted CA and uses OCSP for certificate revocation checking to validate certificates. CA certificates must have the CA flag set to “True” in the basicConstraints extension to be accepted by the TOE. Server certificates must have the “Server Authentication” purpose in the extendedKeyUsage field, and OCSP Responders must have the “OCSP Signing” purpose.

**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 (FIA\_X509\_EXT.1) of the ST states the TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS connections. The certificate validation checking takes place during the TLS session setup.

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

- Manual cut-and-paste - ESA displays the Certificate Request Message via the GUI or CLI interface. This allows the administrator to copy the certificate request and in a secure offline manner send the request to a Certification Authority to be transformed into an X.509v3 public-key certificate.
- Both the certificate request message and the certificates themselves provide protection in that they are digitally signed. If a certificate is modified in any way, it would be invalidated. The digital signature verification process would show that the certificate had been tampered with when the hash value would be invalid.
- The certificate chain establishes a sequence of trusted certificates, from a peer certificate to the root CA certificate. Within the PKI hierarchy, all enrolled peers can validate the certificate of one another if the peers share a trusted root CA certificate or a common subordinate CA. When a certificate chain is received from a peer, the default processing of a certificate chain path continues until the first trusted certificate is reached.
- The authorized administrator can also configure one or more certificate fields together with the
  - ir matching criteria to match. Such as:
    - o alt-subject-name
    - o issuer-name
    - o name
    - o serial-number
    - o subject-name

The physical security of the TOE (A.PHYSICAL\_PROTECTION) protects the switch and the certificates from being tampered with or deleted. In addition, the TOE identification and authentication security functions protect an unauthorized user from gaining access to the TOE.

OCSP is used for certificate revocation. The TOE will act accordingly based on the Authority Info Access extension in the CA or Peer certificate.



Checking is also done for the basicConstraints extension and the CA flag to determine whether they are present and set to TRUE. The local certificate that was imported must contain the basic constraints extension with the CA flag set to true, the check also ensure that the key usage extension is present, and the keyEncipherment bit or the keyAgreement bit or both are set. If they are not, the certificate is not accepted.

If the connection to determine the certificate validity cannot be established, the certificate is not accepted.

**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 Certificates” of the Admin Guide describes how to configure a trustpoint to be used to verify the certificates during TLS authentication.

Section “Remote Logging Configuration” describes how to restrict certificate validation to a single trustpoint if desired.

If the TOE determines that a certificate is invalid or is unable to reach the OCSP Responder, it will not establish the TLS trusted channel connection. Administrators may then choose to:

- Confirm that the OCSP Responder is up and reachable,
- Confirm all CA certificates have a CA flag set to “True”,
- Confirm peer certificates are valid and signed by a trusted CA.

**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 alternately configured a test peer to send an authentication certificate with valid/accessible revocation servers and an authentication certificate with revocation information referring to an inaccessible revocation server. In each case, the evaluator then attempted to make a connection between the test peer and the TOE. The connection was successful when the revocation server is accessible and not successful when the revocation server is not accessible only if that behavior is claimed for the TOE.



## 2.4 SECURITY MANAGEMENT (FMT)

### 2.4.1 MANAGEMENT OF SECURITY FUNCTIONS BEHAVIOUR (NDcPP22E:FMT\_MOF.1/MANUALUPDATE)

#### 2.4.1.1 NDcPP22E:FMT\_MOF.1.1/MANUALUPDATE

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

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

There are no specific requirements for non-distributed TOEs.

The TOE is not distributed.

**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 “Secure Acceptance of the TOE” in the Admin Guide describes the process to install and update the TOE. The installation requires a restart of the TOE.

The TOE is not distributed.

**Component Testing Assurance Activities:** The evaluator shall try to perform the update using a legitimate update image without prior authentication as Security Administrator (either by authentication as a user with no



administrator privileges or without user authentication at all - depending on the configuration of the TOE). The attempt to update the TOE should fail.

The evaluator shall try to perform the update with prior authentication as Security Administrator using a legitimate update image. This attempt should be successful. This test case should be covered by the tests for FPT\_TUD\_EXT.1 already.

The set of functions available to administrators prior to login do not include TOE update as specified in NDcPP22e:FIA\_UIA\_EXT.1-t2. The successful update of the TOE by an authorized administrator was demonstrated in NDcPP22e:FPT\_TUD\_EXT.1-t1.

## 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 (FMT\_MTD.1/CoreData) of the ST states that the warning and access banner may be displayed prior to the identification and authentication of an Authorized Administrator. No administrative functionality is available prior to administrative login.

The inverse of this means that all administrative functions require administrative login (which would include handling of its X509 certificates for the truststore).

**Component Guidance Assurance Activities:** The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is



identified, and that configuration information is provided to ensure that only administrators have access to the functions.

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

The section “X.509 Certificates” in the Admin Guide describes how to install/configure the CA certificate to be used for TLS.

Refer to the guidance activities throughout the AAR which address all TSF-data-manipulating functions.

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

No separate testing for FMT\_MTD.1/CoreData is required as the management functions have been exercised alongside their relevant tests.

### **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 (FMT\_MTD.1/CryptoKeys) of the ST states the TOE provides the ability for Authorized Administrators to access TOE data. This includes cryptographic keys.



This access takes the form of management activity that will create keys, import keys, configuring the use of keys and destroy (zeroize) keys. The keys that can be managed x509v3 certificates for the truststore, and SSH host and user public keys.

Section 6 (FIA\_X509\_EXT.2) of the ST describes how X509 certificates are managed via trustpoints.

Section 6 (FCS\_SSHS\_EXT.1) of the ST describes how SSH host keys and public keys can be managed.

Section 6.2 of the ST describes how these keys can be destroyed.

See the AGD activity below for more information on how the keys are managed.

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

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

The TOE is not distributed.

Section "Remote Administration Protocols" in the **Admin Guide** provides instructions for generating an SSH host key. It describes how to configure a public key for a user for SSH public key authentication.

Section "X.509 Certificates" in the **Admin Guide** provides details for managing the X509 certificates used for trustpoints.

**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 set of functions available to administrators prior to login do not include performing any cryptographic functions as specified by NDcPP22e:FIA\_UIA\_EXT.1-t2. The successful configuration of SSH key by an authorized administrator is demonstrated in NDcPP22E:FCS\_SSHS\_EXT.1.2 test 1.





## 2.4.4 SPECIFICATION OF MANAGEMENT FUNCTIONS - PER TD063 1 (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 (FMT\_SMF.1) of the TOE provides all the capabilities necessary to securely manage the TOE. The Authorized Administrators user can connect to the TOE using the CLI to perform these functions via SSHv2 secured connection, a terminal server, or at the local console.

The specific management capabilities available from the TOE include:



- Local and remote administration of the TOE and the services provided by the TOE via the TOE CLI, as described above;
- The ability to manage the warning banner message and content which allows the Authorized Administrator the ability to define warning banner that is displayed prior to establishing a session (note this applies to the interactive (human) users; e.g. administrative users);
- The ability to update the NX-OS software. The validity of the image is provided using SHA-512 and/or digital signature prior to installing the update;
- The ability to manage the time limits of session inactivity which allows the Authorized Administrator the ability to set and modify the inactivity time threshold;
- The ability to manage termination of a local session due to exceeding the threshold of authentication failure attempts. The account is locked until the Authorized Administrator unlocks the account;
- The ability to manage audit behavior and the audit logs which allows the Authorized Administrator to view the audit logs and to clear the audit logs.
- The ability to manage the cryptographic functionality which allows the Authorized Administrator the ability to identify and configure the algorithms used to provide protection of the data, such as generating the RSA keys to enable SSHv2 and to configure thresholds for SSH rekeying;
- The ability to manage the trusted public key database for SSH user public key authentication
- The ability to manage the cryptographic functionality for TLS remote syslog communication. This includes configuring the reference identifier of the peer.
- The ability to configure and set the time clock;
- The ability to manage the TOE's trust store and designate X.509v3 certificates as trust anchors;
- The ability to import X.509v3 certificates to the TOE's trust store.

All of the management functions above are addressed in the TSS and Guidance activities throughout the AAR as part of the SFR they pertain to.

**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 of the management functions were demonstrated throughout the course of testing.

## 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 (FMT\_SMR.2) in the ST states the TOE platform maintains both privileged and semi-privileged administrator roles. The terms "Authorized Administrator" and "Security Administrator" are used interchangeable in this ST to refer to any user that has been assigned to a privilege level that is permitted to perform the relevant action; therefore, has the appropriate privileges to perform the requested functions. The assigned role determines the functions the user can perform; hence the authorized administrator with the appropriate privileges.



**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.

See FIA\_UIA\_EXT.1 which identifies the instructions in the **Admin Guide** for administering the TOE both locally and remotely.

**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.

Throughout the course of testing, the TOE was administered both via local CLI and SSH.

## 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 (FPT\_APW\_EXT.1) of the ST states the TOE prevents reading of cryptographic passwords. AES password encryption can be configured by an authorized administrator to encrypt stored passwords using the "feature password encryption aes" command.

In this manner, the TOE ensures that plaintext user passwords will not be disclosed even to administrators.

**Component Guidance Assurance Activities:** None Defined

**Component Testing Assurance Activities:** None Defined

## **2.5.2 PROTECTION OF TSF DATA (FOR READING OF ALL PRE-SHARED, SYMMETRIC AND PRIVATE KEYS) (NDcPP22E:FPT\_SKP\_EXT.1)**

### **2.5.2.1 NDcPP22E:FPT\_SKP\_EXT.1.1**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

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

The TOE stores all private keys in a secure directory that is not readily accessible to administrators. All pre-shared and symmetric keys are stored in encrypted form using AES encryption to additionally obscure access. This functionality is configured on the TOE using the 'feature password encryption aes' command.

The TOE is configured to not display configured keys as part of configuration files using the 'hidekeys' command.

**Component Guidance Assurance Activities:** None Defined

**Component Testing Assurance Activities:** None Defined

## **2.5.3 RELIABLE TIME STAMPS - PER TD0632 (NDcPP22E:FPT\_STM\_EXT.1)**



### 2.5.3.1 NDcPP22E:FPT\_STM\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.5.3.2 NDcPP22E:FPT\_STM\_EXT.1.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

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

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

Section 6 (FPT\_STM\_EXT.1) of the ST states the Nexus 9000 Series switch can provide hardware-based timestamp that are used to provide that timestamp in audit records. The TOE provides the use the of internally generated time stamps.

The date and time are used as the time stamp that is applied to TOE generated audit records and used to track inactivity of administrative sessions. This function can only be accessed from within the configuration exec mode via the privileged mode of operation of the TOE. The clock function is reliant on the system clock provided by the underlying hardware.

**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 “Clock Management” in the **Admin Guide** provides instructions for the administrator to manually set the time on the TOE.

Setting the time via NTP or an underlying VS is not supported.

**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 then observed via the console display and the audit records that the time was set as intended.

Test 2: Not applicable. The TOE does not support the use of an NTP server.

Test 3: Not applicable. The TOE does not obtain its time from the 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 (FPT\_TST\_EXT.1) of the ST states the TOE runs a suite of self-tests during initial start-up to verify its correct operation. Refer to the FIPS Security Policy for available options and management of the cryptographic self-test.

For testing of the TSF, the TOE automatically runs checks and tests at startup and during resets to ensure the TOE is operating correctly. Refer to the Guidance documentation for installation configuration settings and information and troubleshooting if issues are identified.

When the system is booted up in FIPS mode, the FIPS power-up self-tests run on the supervisor and line card modules. If any of these FIPS self-tests fail, the whole system is moved to the FIPS error state. In this state, as per the FIPS requirement, all cryptographic keys are deleted, and all line cards are shut down. This mode is exclusively meant for debugging purposes.

Once the switch is in the FIPS error state, any reload of a line card moves it to the failure state. To move the switch back to FIPS mode, it has to be rebooted. However, once the switch is in FIPS mode, any power-up self-test failure on a subsequent line card reload or insertion affects only that line card, and only the corresponding line card is moved to the failure state.

If any of the self-tests fail, the TOE transitions into an error state. In the error state, all secure data transmission is halted and the TOE outputs status information indicating the failure.

The following cryptographically self-tests tests are run:

- AES Known Answer Test
- HMAC Known Answer Test
- RNG/DRBG Known Answer Test
- SHA-1/SHA-256/SHA-384/SHA-512 Known Answer Test





- RSA Signature Known Answer Test (both signature/verification)
- Software Integrity Test

**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-Tests” states if any of these bootup tests fail, the whole system is moved to the error state. In this state, all cryptographic keys are deleted, and all line cards are shut down. This mode is exclusively meant for debugging purposes.

Once the switch is in the error state, any reload of a line card moves it to the failure state. To move the switch back to operational mode, it must be rebooted. However, once the switch is in operational mode, any power-up self-test failure on a subsequent line card reload or insertion affects only that line card, and only the corresponding line card is moved to the failure state.

All ports are blocked from moving to forwarding state during the POST. Only when all components of all modules pass the POST is the system placed in an operational state and ports are allowed to forward data traffic.

If any of the POST fail:

- Restart the TOE to perform POST and determine if normal operation can be resumed.

If the problem persists, contact Cisco Technical Assistance via:

<https://mycase.cloudapps.cisco.com/case> or 1 800 553-2447

**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 TOE provides self-tests consistent with the FIPS 140-2 requirements. When the system is booted up in FIPS mode, the FIPS power-up self-tests run as part of the Power on Startup Test (POST) on the supervisor and line card modules. These self-tests include the following:

- AES Known Answer Test
- HMAC Known Answer Test
- RNG/DRBG Known Answer Test
- SHA-1/SHA-256/SHA-384/SHA-512 Known Answer Test
- RSA Signature Known Answer Test (both signature/verification)
- Software Integrity Test

During the system bootup process (power on or reboot), all the Power on Startup Test (POST) components for all the cryptographic modules perform the POST for the corresponding component (hardware or software). Also, during the initialization and self-tests, the module inhibits all access to the cryptographic algorithms. Additionally, the power-on self-tests are performed after the cryptographic systems are initialized but prior to the underlying OS initialization of external interfaces; this prevents the security appliances from passing any data before completing self-tests and entering FIPS mode. In the event of a power-on self-test failure, the cryptographic module will force the IOS platform to reload and reinitialize the operating system and cryptographic module. This operation ensures no cryptographic algorithms can be accessed unless all power on self-tests are successful.

The evaluator rebooted the device in fips mode and confirmed that the TOE powered on 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 6.1 (FPT\_TUD\_EXT.1) of the ST states an Authorized Administrator can query the software version running on the TOE and can initiate updates to (replacements of) software images.

When software updates are made available by Cisco, an administrator can obtain, verify the integrity of, and install those updates. The updates can be downloaded from the software.cisco.com.

The validity of the image is verified using SHA-512 published hash and digital signature prior to installing the update.

The TOE image files are digitally signed so their integrity can be verified during the boot process, and an image that fails an integrity check will not be loaded. The digital certificates used by the update verification mechanism are contained on the TOE. Detailed instructions for how to do this verification are provided in the administrator guidance for this evaluation.

The TOE does not support automatic updates or checking of updates. The TOE is not distributed.

**Component Guidance Assurance Activities:** The evaluator shall verify that the guidance documentation describes how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the guidance documentation needs to describe how to query the loaded but inactive version.

The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

If a published hash is used to protect the trusted update mechanism, the evaluator shall verify that the guidance documentation describes how the Security Administrator can obtain authentic published hash values for the updates.

For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT\_TUD\_EXT.1, how all TOE components are updated, and the



error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. The guidance documentation only has to describe the procedures relevant for the Security Administrator; it does not need to give information about the internal communication that takes place when applying updates.

If this 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 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 “Secure Acceptance of the TOE” in the **Admin Guide** details the steps for downloading and applying a TOE image.

Approved methods for obtaining a Common Criteria evaluated software images:

Download the Common Criteria evaluated software image file from Cisco.com onto a trusted computer system. The reason to download to a trusted system within your organization, such as the management workstation, is to ensure the file has not been tampered with prior to securely copying to the TOE for installation.

Software images are available from Cisco.com at the following:

<https://software.cisco.com>

The TOE ships with the correct software images installed, however this may not be the evaluated version.

The "show software authenticity file *filename*" command on the Cisco Nexus 9K Series can be used to verify the digital signature of the image.

The TOE implements digital signature verification on the image being installed and will cancel the update process if the signature fails validation. If the update process fails, contact Cisco Technical Assistance Center (TAC).

The administrator must use the “show version” command to display the currently running system image filename and the system software release version.

**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: The evaluator verified the current TOE version and then followed guidance procedures to install a new image to the TOE. The evaluator then verified the TOE version again and confirmed that it was updated successfully.



Test 2: For each device, the evaluator used legitimate updates that were 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

Attempts to update with each of these modified images failed as expected.

Test 3: Not applicable. The does not verify the integrity of updates using published hashes.

## 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 (FTA\_SSL.3) in the ST states an administrator can configure maximum inactivity times individually for remote administrative sessions through the use of the “session-timeout” setting applied to the console. These settings are not immediately activated for the current session. The current line console session must be exited. When the user logs back in, the inactivity timer will be activated for the new session. If a remote user session is inactive for a configured period of time, the session will be terminated and will require authentication to establish a new session.

The allowable inactivity timeout range is from 0 to 525600 minutes, although a value of 0 should not be used to ensure an inactivity timeout is enforced. Administratively configurable timeouts are also available for the EXEC level access (access above level 1) through use of the “exec-timeout” setting.

**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 “Session Termination” in the **Admin Guide** provides instructions for configuring the inactivity time period for remote administrative session termination.

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

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

The evaluator performed a test that demonstrates operation of inactivity timeouts over SSH with timeout values of 1 minute, 3 minutes and 5 minutes. For each time period, the evaluator recorded the time immediately prior to login, logged in, and performed no further actions. When the TOE terminated the SSH session for inactivity the evaluator recorded the time again. The period of time observed matched the configured timeout values.

## 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 (FTA\_SSL.4) in the ST states an administrator is able to exit out of both local and remote administrative sessions by using the ‘exit’ command.

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

Section “Session Termination” of the **Admin Guide** states to manually log out of a CLI session (console or SSH), use the “exit” command once in EXEC mode. The “end” command will return the user back to EXEC mode from any configuration mode that they may be in.

**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 performed a logout for each interactive console session by typing “exit” after a successful authentication attempt.

Test 2: The evaluator performed a logout for each remote interactive session by typing “exit” after a successful authentication attempt.

### 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 (FTA\_SSL\_EXT.1) in the ST states an administrator can configure maximum inactivity times individually for local administrative sessions through the use of the “session-timeout” setting applied to the console. These settings are not immediately activated for the current session. The current line console session must be exited. When the user logs back in, the inactivity timer will be activated for the new session. If a local user session is inactive for a configured period of time, the session will be terminated. The local user will need to re-authentication to start a new session.

The allowable inactivity timeout range is from 0 to 525600 minutes, although a value of 0 should not be used to ensure an inactivity timeout is enforced. Administratively configurable timeouts are also available for the EXEC level access (access above level 1) through use of the “exec-timeout” setting.

**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 “Session Termination” in the **Admin Guide** provides instructions for configuring local admin session termination after a specified time period of inactivity by using the exec-timeout command applied to the console lines.

**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 performed a test that demonstrates operation of inactivity timeouts on the local console with timeout values of 1 minute, 3 minutes and 5 minutes. For each time period, the evaluator recorded the time immediately prior to login, logged in, and performed no further actions. When the TOE terminated the local console session for inactivity the evaluator recorded the time again. The period of time observed matched the configured timeout values.

## **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 6.1 (FTA\_TAB.1) in the ST states the TOE displays a customizable login banner on the local and remote CLI management interface prior to allowing any administrative access to the TOE.



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

Section “Login Banners” in the **Admin Guide** provides instructions for configuring the banner 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.

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

## 2.7 TRUSTED PATH/CHANNELS (FTP)

### 2.7.1 INTER-TSF TRUSTED CHANNEL - PER TD0639 (NDcPP22E:FTP\_ITC.1)

#### 2.7.1.1 NDcPP22E:FTP\_ITC.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.7.1.2 NDcPP22E:FTP\_ITC.1.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.7.1.3 NDcPP22E:FTP\_ITC.1.3

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined



**Testing Assurance Activities:** None Defined

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

Section 6 (FTP\_ITC.1) states the TOE protects communications between the TOE and the remote audit server using TLS. This provide secure channels to transfer audit logs and data.

**Component Guidance Assurance Activities:** The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

Section “Remote Logging Configuration” in the **Admin Guide** provides instructions for configuring and establishing TLS connections between the TOE and a remote audit server. The **Admin Guide** describes that the TOE is constantly renegotiating the connection to push syslog data as needed. If connection to the remote logging server is interrupted, the TOE will continue to save logs locally while attempting to renegotiate with the server. The renegotiation attempts by the TOE will continue until the connection is restored, no matter the length of time.

**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: The TOE support for TLS protected syslog was demonstrated and the evaluator viewed that the TOE initiated the connection.

Test 2: As demonstrated in NDcPP22e:FTP\_ITC.1-t1, the TOE is able to initiate communication via the trusted channel for transmitting audit records to a secure audit server and for communication with an external authentication server.

Test 3: As demonstrated in NDcPP22e:FTP\_ITC.1-t1, the TOE is able to initiate communication via the trusted channel for transmitting audit records to a secure audit server. The packet captures confirm that the channel data is encrypted and is not in plain text.

Test 4: The evaluator started a packet capture and established a TLS connection between the TOE and the external test server. The evaluator observed encrypted traffic between the TOE and the test server carrying syslog traffic.

The evaluator then initiated the physical interruption of the connection for roughly 30 seconds and then restored the connection. The packet capture showed no traffic between the TOE and external server during the disruption. The evaluator also observed that no new audit records appeared on the syslog server during the disruption.



Upon reconnection the TOE continued establishing TLS tunnels and sending encrypted syslog data between the TOE and the test server. The evaluator also observed audit records appearing on the syslog server after the connection was repaired.

The evaluator repeated the above steps using a duration of roughly 10 minutes. The evaluator observed that no new audit records appeared on the syslog server during the disruption. Upon reconnection, the evaluator observed that the TOE initiated a new TLS tunnel between the TOE and the test server. The evaluator also observed audit records appearing on the syslog server after the connection was repaired.

## 2.7.2 TRUSTED PATH - PER TD0639 (NDCPP22E:FTP\_TRP.1/ADMIN)

### 2.7.2.1 NDCPP22E:FTP\_TRP.1.1/ADMIN

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.7.2.2 NDCPP22E:FTP\_TRP.1.2/ADMIN

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.7.2.3 NDCPP22E:FTP\_TRP.1.3/ADMIN

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

**Component TSS Assurance Activities:** The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.



Section 6 (FTP\_TRP.1/Admin) in the ST states that all remote administrative communications take place over a secure encrypted SSH session. The SSH session is encrypted using AES encryption. A remote authorized administrator is able to initiate SSH communications with the TOE

**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 “Remote Administration Protocols” provides instructions for establishing the remote administrative sessions using SSH.

**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 and Test 2: Remote SSH administration was tested throughout the course of the evaluation. The successful testing of the SSH channel and demonstration of its encryption was performed in FCS\_SSHS\_EXT.1 on all devices.





### 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 programming 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'.

For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS 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.



Section “Enabling FIPS Mode” in the **Admin Guide** describes how to enable FIPS mode to ensure proper cryptographic functionality in the evaluated configuration. Section “Excluded Functionality” describes what protocols that are not included in the evaluated configuration need to be disabled. All sections in the **Admin Guide** which describe configuring TOE security functions include only those settings that should be enabled and can be used in the evaluated configuration.

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 evaluator had the Admin Guide to use when configuring the TOE. The completeness of the Admin Guide is addressed by its use in the AA's carried out in the evaluation.

### 3.3 LIFE-CYCLE SUPPORT (ALC)

#### 3.3.1 LABELLING OF THE TOE (ALC\_CMC.1)

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

The evaluator performs the CEM work units associated with ALC\_CMC.1.

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



The evaluator verified that the ST, TOE, and Admin Guide 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 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.

The following diagram indicates the test environment (TLS/SSH configurations):

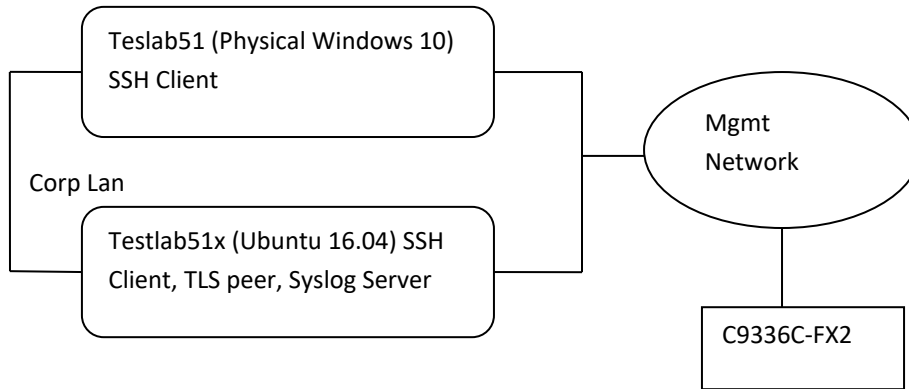


Figure 1 Test Setup

Test Servers	IP addresses and MAC addresses used by this Test Server
Windows 10 Pro Physical Test Platform (Testlab16)	192.168.144.250, 2001:192:168:144::250 (00-15-17-BE-0D-F8) 172.16.16.250, 2001:172:16:16::250 (00-15-17-BE-0D-F9)
Ubuntu 16.04 Test Server (TL16-16x)	Eth1:0: 172.16.16.151, 2001:172:16:16::254 (00:15:5d:00:da:01) Eth2: 172.16.16.51, 2001:172:16:16::239 (00:15:5d:00:da:02) Eth4: 172.16.16.52, 2001:172:16:16::244 (00:15:5d:00:da:04)



TOE Device	IP addresses and MAC addresses used by this TOE Device
C9336C-FX2	172.16.16.133 (Physical a0:bc:6f:ff:b5:44 VLAN a0:bc:6f:ff:b5:4b)

**TOE Platform:**

- C9336C-FX2 model running nxos version 10.4.4.M

Management functions and Audit data collection were performed via the direct console connection or via SSH over the 172.16.16.x network.

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

- Openssl version 1.0.2g
- Openssh client version 7.2p2
- Big Packet Putty, Openssh-client version 6.8p1
- Rsyslog version 8.16.0
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Nmap version 7.01
- Stunnel 5.30
- Strongswan v5.3.5
- FreeRADIUS Version 3.0.15
- Scapy version 2.4.5

### 3.5 VULNERABILITY ASSESSMENT (AVA)

#### 3.5.1 VULNERABILITY SURVEY (AVA\_VAN.1)

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





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

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

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

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

The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic libraries, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis. (TD0547 applied)

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

- a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]
- b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]
- c) additional information in the Preparative Procedures as identified in the refinement of AGD\_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

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

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



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

- National Vulnerability Database (<https://web.nvd.nist.gov/vuln/search>),
- Vulnerability Notes Database (<http://www.kb.cert.org/vuls/>),
- Rapid7 Vulnerability Database (<https://www.rapid7.com/db/vulnerabilities>),
- Tipping Point Zero Day Initiative (<http://www.zerodayinitiative.com/advisories>),
- 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 January 7, 2024. The search was conducted with the following search terms: "Cisco Nexus 9000", "Cisco Nexus 9300", "Cisco Nexus", "Cisco", "Nexus", "NX-OS", "N9K", "Intel Xeon D-1633N", "Intel Xeon D-1526", "CiscoSSL", "Intel Xeon D-1528", "Intel Xeon D-1623N", "Intel Xeon D-1530", "Cisco Nexus 9400", "Cisco Nexus 9500", "Cisco Nexus 9800".

### 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

The TOE does not support a TLS server implementation and therefore is not subject to Bleichenbacher attacks.