Assurance Activities Report

for

Crestron DigitalMedia NVX®AV-over-IP v7.1

Version 1.1

4 October 2024

Prepared by:

Leidos Inc.
https://www.leidos.com/CC-FIPS140
Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive
Columbia, MD 21046

Prepared for:

National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

The Developer of the TOE:

Crestron Electronics, Inc.
15 Volvo Drive
Rockleigh, NJ 07647

The TOE Evaluation was Sponsored by:

Crestron Electronics, Inc.
15 Volvo Drive
Rockleigh, NJ 07647

Evaluation Personnel:

Josh Marciante
Kevin Zhang
Pascal Patin

Common Criteria Version:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction and general model, 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 Version:

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

Protection Profile:

- *collaborative Protection Profile for Network Devices*, Version 2.2e, 23 March 2020 [NDcPP]
- *Evaluation Activities for Network Device cPP*, Version 2.2, December 2019 [SD-ND]

Revision History

Version	Date	Description
0.1	04 April 2024	Initial draft
1.0	11 September 2024	Version for NIAP submission
1.1	4 October 2024	Update in response to ECR comments

Contents

1 Introduction

This document presents results from performing assurance activities associated with the Crestron DigitalMedia NVX®AV-over-IP v7.1 product evaluation. This report contains sections documenting the performance of assurance activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in Evaluation Activities for Network Device cPP, Version 2.2, December 2019 and including the following optional and selection-based SFRs: : FAU_STG.1, FCS_HTTPS_EXT.1; FCS_NTP_EXT.1; FCS_SSHS_EXT.1; FCS_TLSC_EXT.1; FCS_TLSS_EXT.1; FIA_X509_EXT.1/Rev; FIA_X509_EXT.2; FIA_X509_EXT.3; and FMT_MTD.1/CryptoKeys.

Note that, in accordance with NIAP Policy Letter #5, all cryptography in the TOE for which NIST provides validation testing of FIPS-approved and NIST-recommended cryptographic algorithms and their individual components must be NIST validated. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity. As such, Test activities associated with functional requirements within the scope of Policy Letter #5 are performed by verification of the relevant CAVP certification and not through performance of any testing as specified in the PP or its supporting document.

1.1 Applicable Technical Decisions

The NIAP Technical Decisions referenced below apply to [NDcPP]. Rationale is included for those Technical Decisions that do not apply to this evaluation.

TD0527 – Updates to Certificate Revocation Testing (FIA_X509_EXT.1)

- This TD is applicable to the TOE.

TD0528 – NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4

- This TD is applicable to the TOE.

TD0536 – NIT Technical Decision for Update Verification Inconsistency

- This TD is applicable to the TOE.

TD0537 – NIT Technical Decision for Incorrect reference to FCS_TLSC_EXT.2.3

- This TD is applicable to the TOE but relates solely to Application Notes and thus is not applicable to this document.

TD0546 – NIT Technical Decision for DTLS - clarification of Application Note 63

- This TD is not applicable to the TOE because DTLS is not claimed.

TD0547 – NIT Technical Decision for Clarification on developer disclosure of AVA_VAN

- This TD is applicable to the TOE.

TD0555 – NIT Technical Decision for RFC Reference incorrect in TLSS Test

- This TD is applicable to the TOE.

TD0556 – NIT Technical Decision for RFC 5077 question

- This TD is applicable to the TOE.

TD0563 – NiT Technical Decision for Clarification of audit date information

- This TD is applicable to the TOE but relates solely to Application Notes and thus is not applicable to this document.

TD0564 – NiT Technical Decision for Vulnerability Analysis Search Criteria
- This TD is applicable to the TOE.

TD0569 – NIT Technical Decision for Session ID Usage Conflict in FCS_DTLSS_EXT.1.7
- This TD is applicable to the TOE.

TD0570 – NiT Technical Decision for Clarification about FIA_AFL.1
- This TD is a clarification to the requirement and therefore is generally applicable to the TOE but does not change any of the requirements or this document.

TD0571 – NiT Technical Decision for Guidance on how to handle FIA_AFL.1
- This TD is a clarification to the requirement and therefore is generally applicable to the TOE but does not change any of the requirements or this document.

TD0572 – NiT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers
- This TD is a clarification to the FCS_TLSC_EXT and FTP_ITC.1 requirements and therefore is generally applicable to the TOE but does not change any of the requirements.

TD0580 – NiT Technical Decision for clarification about use of DH14 in NDcPPv2.2e.
- This TD is applicable to the TOE.

TD0581– NiT Technical Decision for Elliptic curve-based key establishment and NIST SP 800-56Arev3
- This TD is applicable to the TOE but relates solely to ST selections so it is not applicable to this document.

TD0591– NIT Technical Decision for Virtual TOEs and hypervisors
- This TD is applicable to the TOE but does not modify any SFRs or SARs so it is not applicable to this document.

TD0592– NIT Technical Decision for Local Storage of Audit Records
- This TD is applicable to the TOE but does not modify any SFRs or SARs so it is not applicable to this document.

TD0631– NIT Technical Decision for Clarification of public key authentication for SSH Server
- This TD is applicable to the TOE.

TD0632– NIT Technical Decision for Consistency with Time Data for vNDs
- This TD is applicable to the TOE.

TD0635– NIT Technical Decision for TLS Server and Key Agreement Parameters
- This TD is applicable to the TOE.

TD0636– NIT Technical Decision for Clarification of Public Key User Authentication for SSH
- This TD is not applicable to the TOE because the ST does not claim FCS_SSHC_EXT.1.

TD0638– NIT Technical Decision for Key Pair Generation for Authentication
- This TD is a clarification to FCS_CKM.1 and therefore is generally applicable to the TOE but does not change any of the requirements.

TD0639– NIT Technical Decision for Clarification for NTP MAC Keys

- This TD is a clarification on the level of protection of NTP MAC keys and is generally applicable to the TOE but does not change any of the requirements.

TD0670– NIT Technical Decision for Mutual and Non-Mutual Auth TLSC Testing

- This TD is applicable to the TOE.

TD0738– NIT Technical Decision for Link to Allowed-With List

- This TD is a clarification to the PP Allowed-With list and is generally applicable to the TOE but does not change any of the requirements.

TD0790– NIT Technical Decision: Clarification Required for testing IPv6

- This TD is applicable to the TOE.

TD0792– NIT Technical Decision: FIA_PMG_EXT.1 - TSS EA not in line with SFR

- This TD is applicable to the TOE.

TD0800– Updated NIT Technical Decision for IPsec IKE/SA Lifetimes Tolerance

- This TD is not applicable to the TOE because FCS_IPSEC_EXT.1 is not claimed.

1.2 Evidence

[ST]	Crestron DigitalMedia NVX®AV-over-IP v7.1 Security Target, Version 1.0, 3 October 2024
[CCECG]	Crestron DigitalMedia NVX® AV-over-IP v7.1 Common Criteria Evaluated Configuration Guide (CCECG), Version 1.0
[ND_SD]	Evaluation Activities for Network Device cPP, December-2019, Version 2.2
[QS_8391C]	DM-NVX-350, DM-NVX-351, and DM-NVX-352 Quick Start
[QS_8392B]	DM-NVX-350C, DM-NVX-351C, and DM-NVX-352C Quick Start
[QS_9001A]	DM-NVX-E10 and DM-NVX-D10 Quick Start
[QS_9000A]	DM-NVX-E20 and DM-NVX-D20 Quick Start
(QS_8906B)	DM-NVX-E30 and DM-NVX-D30 Quick Start
[QS_9160A]	DM-NVX-E20-2G Quick Start
[QS_9091A]	DM-NVX-D200 Quick Start
[QS_8526A]	DM-NVX-D80-IOAV Quick Start
[QS_8634A]	DM-NVX-363 and DM-NVX-360 Quick Start
[QS_8636A]	DM-NVX-363C and DM-NVX-360C Quick Start
[QS_8646B]	DM-NVX-E760 Quick Start
[QS_8638B]	DM-NVX-E760C Quick Start
[QS_8346A]	DM-NVX-E30C/DM-NVX-D30C Quick Start

2 Security Functional Requirement Evaluation Activities

This section describes the evaluation activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The evaluation activities are derived from [ND-SD] and modified by applicable NIAP Technical Decisions. Evaluation activities for SFRs not claimed by the TOE have been omitted.

2.1 Security Audit (FAU)

2.1.1 Audit Data Generation (FAU_GEN.1)

2.1.1.1 TSS 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.

Section 5.1.1 of [ST] ("Audit Data Generation") states all actions by administrators that involve keys are performed in association with certificates. When auditing certificate operations, the TOE logs the Issuer and serial number field of the certificate.

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.

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.1.2 Guidance 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).

Section "FAU_GEN.1 Audit Generation" of [CCECG] contains two tables that together provide examples of each auditable event required by FAU_GEN.1. The first table provides examples of each of the audit records associated with the specific SFRs as identified in Table 6 of [ST] ("Security Functional Requirements and Auditable Events"). The second table provides examples of each of the audit records associated with administrative actions related to TSF data related to configuration changes. The evaluator examined the audit record tables in [CCECG] and compared the contents to the audit record requirements from Table 4 of [ST]. The evaluator confirmed an example of each required audit record was included and that each record contained the required information of date/time the event was generated, the event type, the subject identity, the outcome of the event, and the additional audit record content specified in Table 4 of [ST], where applicable.

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

The evaluator examined the supplied guidance documentation, identifying all mechanisms available to the administrator for configuring and managing the capabilities of the TOE. Those mechanisms related to the SFRs specified in the ST were identified and mapped to the applicable SFRs. In addition, the evaluator sought to confirm that all SFRs that would be expected to have a management capability related to them had appropriate management capabilities identified in the guidance documentation.

The relevant administrative actions related to TSF data related to configuration changes comprise:

- Configuring the banner displayed prior to authentication
- Specifying the session inactivity time-out period for local and remote administrative sessions
- Initiating manual updates to the TOE and verifying updates prior to installation
- Configuring parameters associated with the authentication failure mechanism (number of consecutive failures allowed, lockout duration)
- Configuration of syslog export settings
- Re-enabling an administrator account
- Management of cryptographic keys—CSRs and TLS private key for web server
- Setting the date and time
- Configuring NTP
- Managing the TOE's trust store and designating X.509v3 certificates as trust anchors
- Importing X.509 v3 certificates to the TOE's trust store
- Setting the length requirement for passwords.
- Resetting account passwords.

2.1.1.3 Test 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.

Audit logs were generated for each of the events required for FAU_GEN.1.

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 TOE is not distributed. Therefore, this activity is not applicable.

2.1.2 User Identity Association (FAU_GEN.2)

2.1.2.1 TSS & Guidance 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.

2.1.2.2 Test 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.

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.3 Protected Audit Trail Storage (FAU_STG.1)

2.1.3.1 TSS Activities

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

Section 5.1.2 of [ST] (“Audit Storage and Audit Record Export”) states the TOE is a single standalone appliance that stores audit records locally. The TOE implements a log rotation policy. The TOE rotates to a new log file when the current log file exceeds 5K in size. The TOE maintains only the most recent 20 log files, so local storage of audit records is capped at 100K.

Section 5.1.2 of [ST] states the TOE does not provide any interfaces to modify or manually delete the stored audit records.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.3.2 Guidance Activities

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

Section "FAU_STG.1 Protected Audit Trail Storage" of [CCECG] states the TOE does not provide any interfaces to modify or manually delete stored audit records. As such, no configuration is necessary to protect the locally stored audit data against unauthorized modification or deletion.

2.1.3.3 Test Activities

The evaluator shall perform the following tests:

Test 1: The evaluator shall access the audit trail without authentication as Security Administrator (either by authentication as a non-administrative user, if supported, or without authentication at all) and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to access the audit trail can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

The evaluator authenticated to the TOE as a non-administrative user. The CLEARAUDITLOG command was then run, which was rejected by the TOE because of the user's access level.

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

The evaluator authenticated to the TOE as an administrative user. The CLEARAUDITLOG command was then run which cleared the audit records on the TOE.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.4 Protected Audit Event Storage (FAU_STG_EXT.1)

2.1.4.1 TSS 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.

Section 5.1.2 of [ST] ("Audit Storage and Audit Record Export") states the administrator can configure the TOE to export audit records to an external audit server in real time. The TOE uses TLS to provide the trusted channel.

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.

Section 5.1.2 of [ST] states the TOE implements a log rotation policy. The TOE rotates to a new log file when the current log file exceeds 5K in size. The TOE maintains only the most recent 20 log files, so local storage of audit records is capped at 100K.

Section 5.1.2 of [ST] states the TOE does not provide any interfaces to modify or manually delete the stored audit records.

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.

Section 5.1.2 of [ST] states the TOE is a standalone TOE that stores audit data locally.

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

According to section 5.1.2 of [ST], the audit service rotates to a new file when the current file exceeds 5K in size. The audit log service maintains only the most recent 20 files, so local audit log space is capped at 100K.

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.

Section 5.1.2 of [ST] states the TOE exports audit records in real time (i.e., as they are generated).

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).

The TOE is not distributed. Therefore, this activity is not applicable.

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.

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.4.2 Guidance 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.

Section "FTP_ITC.1 Inter-TSF Trusted Channel" of [CCECG] describes how the administrator establishes the trusted channel to the audit server, using the `REMOTESYSLOG` CLI command. The guidance identifies the need to configure TLS to ensure secure communication, including the need to install a trusted certificate into the TOE's trust store in order to validate the X.509 certificate presented by the external audit server when establishing the connection. It also identifies the external audit server must support TLS v1.2 in order to be able to establish the trusted channel between the TOE and 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.

Section "FAU_STG_EXT.1 Protected audit event storage" of [CCECG] states the TOE transmits audit records to the configured external audit server in real time (i.e., simultaneously to it being generated by the TOE).

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 "FAU_STG_EXT.1 Protected audit event storage" of [CCECG] states that when log space is exhausted, the oldest logs are overwritten and replaced. This does not require configuration, as it is the only option available.

Test 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 tests for this requirement:

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.

The evaluator established a secure connection between the TOE and an external audit server. Audit data was generated and transferred to the server without any explicit administrator intervention being necessary. A wire capture showed that this data was encrypted and not visible.

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 behavior 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).

The evaluator verified that when TOE audit storage space is exhausted the oldest audit records are overwritten, consistent with the selection in the [ST].

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

The TOE does not comply with FAU_STG_EXT.2/LocSpace. Therefore, this activity is not applicable.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.2 Cryptographic Support (FCS)

2.2.1 Cryptographic Key Generation (FCS_CKM.1)

2.2.1.1 TSS 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 5.2.3 of [ST] ("Cryptographic Key Generation and Establishment") identifies the key sizes supported by the TOE. The TOE supports the following key generation schemes and their usage:

- RSA schemes using cryptographic key sizes of 2048, 3072, or 4096 bits, for authentication
- FFC schemes using cryptographic key sizes of 2048 bits, for key establishment (TLS and SSH)
- ECC schemes using NIST curves P-256, P-384, and P-521, for authentication and for key establishment (TLS and SSH).

2.2.1.2 Guidance 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 "FCS_CKM.1 Cryptographic Key Generation" of [CCECG] states the `FIPSMODE ON` command limits the TOE to using only the selected key generation schemes and key sizes.

2.2.1.3 Test Activities

Modified in accordance with TD0572.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

a) Random Primes:

• Provable primes

• Probable primes

b) Primes with Conditions:

• Primes p1, p2, q1, q2, p and q shall all be provable primes

• Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

• Primes p1, p2, q1, q2, p and q shall all be probable primes

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

• Primes q and p shall both be provable primes

• Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

• Generator g constructed through a verifiable process

• Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x:

• len(q) bit output of RBG where 1 <=x <= q-1

• len(q) + 64 bit output of RBG, followed by a mod q-1 operation and a +1 operation, where 1<= x<=q-1. The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

• g != 0,1

• q divides p-1

• g^q mod p = 1

• g^x mod p = y

for each FFC parameter set and key pair.

Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups

Testing for FFC Schemes using Diffie-Hellman group 14 and/or safe-prime groups is done as part of testing in CKM.2.1.

Section 5.2 of [ST] ("Cryptographic Support"), Table 8 ("Cryptographic Functions Implemented by OpenSSL") identifies the CAVP certifications verifying asymmetric key generation, as follows.

Function	Standards	Certificates
RSA (2048 bits, 3072 bits, 4096 bits)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3	RSA #A1221
ECDSA (P-256, P-384, P-521 curves)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4	ECDSA #A1222
DSA (2048 bits)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.1	DSA #A1222

Modified in accordance with TD0580.

FFC Schemes using "safe-prime" groups

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

2.2.2 Cryptographic Key Establishment (FCS_CKM.2)

2.2.2.1 TSS Activities

Modified in accordance with TD0580.

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.

Section 5.2.3 of [ST] ("Cryptographic Key Generation and Establishment") identifies the following key establishment methods supported by the TOE:

- Elliptic curve-based key establishment schemes
- Finite field-based key establishment schemes.

These key establishment methods correspond to the key generation schemes specified in FCS_CKM.1.

Section 5.2.3 of [ST] identifies the usage for each scheme, as follows:

Scheme	SFR	Service
Elliptic curve-based	FCS_TLSC_EXT.1 FCS_TLSS_EXT.1 FCS_SSHS_EXT.1	Audit Server Administration Administration
Finite field-based	FCS_TLSC_EXT.1 FCS_TLSS_EXT.1 FCS_SSHS_EXT.1	Audit Server Administration Administration

2.2.2.2 Guidance 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 "FCS_CKM.2 Cryptographic Key Establishment" of [CCECG] states the `FIPSMODE ON` command limits the TOE to using only the selected key establishment schemes.

2.2.2.3 Test Activities

Modified in accordance with TD0572.

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.

Diffie-Hellman Group 14

The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 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 Diffie-Hellman group 14.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safeprime 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.

Section 5.2 of [ST] ("Cryptographic Support"), Table 8 ("Cryptographic Functions Implemented by OpenSSL") identifies the CAVP certifications verifying SP 800-56A key establishment schemes, as follows.

Functions	Standards	Certificates
Elliptic curve-based scheme	NIST Special Publication 800-56A Revision 2, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"	KAS-ECC #A1223
Finite field-based scheme	NIST Special Publication 800-56A Revision 2, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"	KAS-FFC #A1223

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.

The TOE does not implement any RSA-based key establishment schemes.

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.

This testing is performed in conjunction with FCS_SSHS_EXT.1.7 Test 2.

2.2.3 Cryptographic Key Destruction (FCS_CKM.4)

2.2.3.1 TSS 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. Where keys are stored encrypted or wrapped under another key then this may need to be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions). 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.

Section 5.2.4 of [ST] ("Cryptographic Key Destruction"), Table 9 ("Key Clearing") lists all relevant keys and includes the following information: storage location of the key or CSP; how the key or CSP is stored (plaintext or encrypted); the purpose of the key or CSP and when it is destroyed. The evaluator examined

the list and confirmed the description of keys and storage locations is consistent with the functions performed by the TOE.

Section 5.2.4 states the TOE clears keys from volatile memory by overwriting the memory locations with zeroes. The TOE clears plaintext keys in non-volatile memory by performing a single overwrite consisting of a new value of the key.

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).

Section 5.2.4 of [ST] identifies (in Table 9) the SSH Host Private Key as the only key stored as plaintext in non-volatile memory (Flash). This key is destroyed by being overwritten by a new value of the key, created using the `SSHSERVER GENHOSTKEY` command. The SSH Host Private Key is stored in a file in a filesystem that is located in Flash. When an administrator requests a new key, the `SSHSERVER GENHOSTKEY` invokes lower-level file system APIs to open and write to the file.

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.

The ST does not select "destruction of reference" or "invocation of an interface" in FCS_CKM.4. As such, this activity is not applicable.

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.

Section 5.2.4 of [ST] identifies (in Table 9) the Web Server Certificate Private Key as the only key stored in a non-plaintext form. It further identifies this key is encrypted using 256 bit AES in CBC mode, using a randomly generated password as the key encrypting key. This password in turn is stored encrypted using 256 bit AES in CBC mode, using a key that is embedded in the firmware source code.

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.

Section 5.2.4 of [ST] states there are no configurations or circumstances that do not conform to the key destruction requirement.

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.

The ST does not specify the use of "a value that does not contain any CSP" to overwrite keys.

2.2.3.2 Guidance 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 "FCS_CKM.4 Cryptographic Key Destruction" of [CCECG] states the TOE is not subject to situation that could prevent or delay key destruction.

2.2.3.3 Test Activities

None defined.

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

2.2.4.1 TSS 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 5.2.1 of [ST] ("Cryptographic Operations") states the TOE supports AES-CBC, AES-GCM, and AES-CTR (128 and 256 bits) for data encryption/decryption.

2.2.4.2 Guidance 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 "FCS_COP.1/DataEncryption Cryptographic Operation (AES Data Encryption/Decryption)" of [CCECG] states no additional configuration is required other than executing the `FIPSMODE ON` command to ensure the TOE uses only the selected modes and key sizes for data encryption/decryption.

2.2.4.3 Test 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 AESCBC decryption.

KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N].

To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of keys and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256- bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation.

The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation.

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1: CT[1] = AES-CBC-Encrypt(Key, IV, PT)

PT = IV

else:

CT[i] = AES-CBC-Encrypt(Key, PT)

PT = CT[i-1]

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AESCBC-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 nonzero 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. Since the 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, AESGCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

There are four Known Answer Tests (KATs) described below to test a basic AES encryption operation (AES-ECB mode). For all KATs, the plaintext, IV, and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 5 plaintext values for each selected keysize and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros.

KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 5 key values for each selected keysize and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value.

KAT-3 To test the encrypt functionality, the evaluator shall supply a set of key values for each selected keysize as described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values. A set of 128 128-bit keys, a set of 192 192-bit keys, and/or a set of 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N].

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384 ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128].

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10 (test shall be performed using AES-ECB mode). For each i the evaluator shall choose a key and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key using a known good implementation. The evaluator shall perform this test using each selected keysize.

AES-CTR Monte-Carlo Test

The evaluator shall test the encrypt functionality using 100 plaintext/key pairs. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

Input: PT, Key

for i = 1 to 1000:

CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation. The evaluator shall perform this test using each selected keysize.

There is no need to test the decryption engine.

Section 5.2 of [ST] (“Cryptographic Support”), Table 8 (“Cryptographic Functions Implemented by OpenSSL”) identifies the CAVP certifications verifying AES encryption and decryption, as follows.

Functions	Standards	Certificates
AES in CBC mode (128, 256 bits) AES in GCM mode (128, 256 bits) AES in CTR mode (128, 256 bits)	ISO 18033-3 (AES) ISO 10116 (CBC mode) ISO 10116 (CTR mode) ISO 19772 (GCM mode)	AES #A1222

2.2.5 Cryptographic Operation (Signature Generation and Verification (FCS_COP.1/SigGen)

2.2.5.1 TSS 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 5.2.1 of [ST] ("Cryptographic Operation") states the TOE supports RSA with key sizes of 2048, 3072, and 4096 bits, and ECDSA with elliptical curves P-256, P-384 and P-521 for digital signature generation. Additionally, the TOE verifies 2048-bit RSA signatures on TOE update packages.

2.2.5.2 Guidance 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 "FCS_COP.1/SigGen Cryptographic Operation (Signature Generation and Verification)" of [CCECG] states no additional configuration is required other than executing the `FIPSMODE ON` command to ensure the TOE uses only the selected algorithms and key sizes for signature services.

2.2.5.3 Test 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.

Section 5.2 of [ST] ("Cryptographic Support"), Table 8 ("Cryptographic Functions Implemented by OpenSSL") identifies the CAVP certifications verifying digital signature services, as follows.

Functions	Standards	Certificates
RSA Digital Signature Algorithm (2048 bit, 3072 bit, and 4096 bit modulus)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSAPKCS1v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3	RSA #A1222
ECDSA Elliptic Curve Digital Signature Algorithm (P-256, P-384, P-521 curves)	FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 6 and Appendix D, Implementing "NIST curves" [P-256, P-384, P-521]; ISO/IEC 14888-3, Section 6.4	ECDSA #A1222

2.2.6 Cryptographic Operation (Hash Algorithm) (FCS_COP.1/Hash)

2.2.6.1 TSS 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 5.2.1 of [ST] ("Cryptographic Operations") states the TOE uses SHA hashing in conjunction with the following cryptographic operations: during digital signature calculation (hashing of the message); for integrity as part of HMAC-SHA operations within TLS; and also for authentication of NTP servers.

2.2.6.2 Guidance Activities

The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

Section "FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)" of [CCECG] states no additional configuration is required other than executing the `FIPSMODE ON` command to ensure the TOE uses only the required hash sizes.

2.2.6.3 Test Activities

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the

length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

Section 5.2 of [ST] (“Cryptographic Support”), Table 8 (“Cryptographic Functions Implemented by OpenSSL”) identifies the CAVP certifications verifying cryptographic hashing, as follows.

Algorithm	Tested Capabilities	Certificates
SHA-1 (digest size 160 bits) SHA-256 (digest size 256 bits) SHA-384 (digest size 384 bits) SHA-512 (digest size 512 bits)	ISO/IEC 10118-3:2004	SHS #A1222

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

2.2.7.1 TSS 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 5.2.1 of [ST] ("Cryptographic Operations") states the HMAC function implemented by the TOE uses key lengths, hash function, block size, and output MAC length as summarized in the following table:

Algorithm	Key Size	Block Size	Message Digest Size
SHA-1	160 bits	512	160
SHA-256	256 bits	512	256
SHA-384	384 bits	1024	384
SHA-512	512 bits	1024	512

2.2.7.2 Guidance 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 "FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithm)" of [CCECG] states no additional configuration is required other than executing the `FIPSMODE ON` command to ensure the TOE uses only the specified keyed-hash algorithms.

2.2.7.3 Test 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.

Section 5.2 of [ST] ("Cryptographic Support"), Table 8 ("Cryptographic Functions Implemented by OpenSSL") identifies the CAVP certifications verifying cryptographic keyed hashing, as follows.

Algorithm	Tested Capabilities	Certificates
HMAC-SHA-1 (key size 160 bits, digest size 160 bits) HMAC-SHA-256 (key size 256 bits, digest size 256 bits) HMAC-SHA-384 (key size 384 bits, digest size 384 bits) HMAC-SHA-512 (key size 512 bits, digest size 512 bits)	ISO/IEC 9797-2:2011, Section 7 "MAC Algorithm 2"	HMAC #A1222

2.2.8 HTTPS Protocol (FCS_HTTPS_EXT.1/Client; FCS_HTTPS_EXT.1/Server))

2.2.8.1 TSS Activities

The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Section 5.2.5.3 of [ST] ("TLS Server Protocol") states the TOE implements HTTPS in compliance with RFC 2818 using TLS.

2.2.8.2 Guidance Activities

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

Section "FCS_HTTPS_EXT.1" of [CCECG] states executing the `FIPSMODE ON` command will limit the cryptographic key generation algorithms, cryptographic key establishment methods, data encryption/ decryption algorithms, signature generation and verification services, cryptographic hashing services, and keyed-hash message authentication to those specified in [ST].

2.2.8.3 Test Activities

This test is now performed as part of FIA_X509_EXT.1/Rev testing. Tests are performed in conjunction with the TLS evaluation activities. If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

Refer to the Test Activities for FIA_X509_EXT.1/Rev.

2.2.9 NTP Protocol (FCS_NTP_EXT.1)

2.2.9.1 TSS Activities

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

Section 5.2.5.4 of [ST] ("NTP Protocol") states the TOE supports NTP v4. It can use SHA-1 or SHA-256 as its means of ensuring the timestamps it receives are from an authenticated source and their integrity has been maintained. This description is consistent with the specification of FCS_NTP_EXT.1 and the selections made in the ST.

2.2.9.2 Guidance Activities

FCS_NTP_EXT.1.1

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

Section "FCS_NTP_EXT.1 NTP Protocol" of [CCECG] provides instructions to the Security Administrator as to how to configure NTP using the `SNTP` CLI command. The instructions cover how to configure multiple NTP servers and how to configure the authentication method consistent with the selections in the ST. The guidance states the TOE supports NTP v4 only and no configuration of NTP version is necessary—the TOE will use the version supported by the NTP server. If the server supports a version other than v4, the TOE will not establish a connection.

FCS_NTP_EXT.1.2

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

Assurance Activity Note:

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

Section "FCS_NTP_EXT.1 NTP Protocol" of [CCECG] provides instructions to the Security Administrator as to how to configure the authentication method consistent with the selections in the ST.

FCS_NTP_EXT.1.3

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

Section "FCS_NTP_EXT.1 NTP Protocol" of [CCECG] states after performing the identified configuration steps, no additional configuration is required to ensure the device will not accept broadcast and multicast NTP packets.

2.2.9.3 Test Activities

FCS_NTP_EXT.1.1

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

The TOE was observed to connect successfully to three NTPv4 servers in a wire capture captured as part of FCS_NTP_EXT.1.4 test 1.

FCS_NTP_EXT.1.2

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

Per the EA, the cryptographic implementation of the protocol is tested as part of an FCS_COP SFR.

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

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

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

The evaluator configured the TOE with an incorrect NTP key to use for message authentication and observed that the TOE did not update its time in response.

The positive test case was seen in FCS_NTP_EXT.1.4, where a connection to three NTP servers with correct MAC values resulted in the TOE updating its timestamp and logging the change.

FCS_NTP_EXT.1.3

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

A server was setup to transmit NTP packets over broadcast and multicast. This was verified by wire capture. The TOE was then observed not to change its time in response to these packets.

Modified in accordance with TD0528.

FCS_NTP_EXT.1.4

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

The TOE was configured to receive time from three NTP time sources while utilizing MAC authentication. Time on the TOE was successfully updated in response to the NTP updates.

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

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

The evaluator configured a custom NTP implementation to send an NTP server message without first receiving an NTP client message. The evaluator verified the timestamp was not updated in response to such messages.

2.2.10 Cryptographic Operation (Random Bit Generation) (FCS_RBG_EXT.1)

2.2.10.1 TSS Activities

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 5.2.2 of [ST] ("Random Bit Generation") states the TOE uses a CTR_DRBG (AES) conformant DRBG, which is seeded from `/dev/random`. This entropy source is assumed to provide full entropy to the caller (i.e., a request for 256 bits to seed the DRBG is assumed to provide 256 bits min-entropy).

2.2.10.2 Guidance Activities

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

Section "FCS_RBG_EXT.1 Random Bit Generation" states no administrator configuration is required for the RNG functionality.

2.2.10.3 Test 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.

Performed in accordance with NIAP Policy Letter #5.

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

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

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

Section 5.2 of [ST] ("Cryptographic Support"), Table 8 ("Cryptographic Functions Implemented by OpenSSL") identifies the CAVP certification verifying deterministic random bit generation, as follows.

Functions	Standards	Certificates
AES-256 CTR_DRBG	ISO/IEC 18031:2011	DRBG #A1222

2.2.11 SSH Server Protocol (FCS_SSHS_EXT.1)

2.2.11.1 TSS Activities

FCS_SSHS_EXT.1.2

Modified in accordance with TD0631.

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

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

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

Section 5.2.5.1 of [ST] ("SSH Server Protocol") states the TOE supports public key-based and password-based authentication methods. The use of password-based authentication is consistent with the selection in FCS_SSHS_EXT.1.2. The SSH public key-based authentication implementation uses the following public key algorithms and rejects all others: ssh-rsa and ecdsa-sha2-nistp256. This description is consistent with the selections made in FCS_SSHS_EXT.1.5, and the corresponding algorithms are selected in FCS_COP.1/SigGen.

The TOE does not support public key algorithms that require X.509v3 certificates.

Section 5.3.1 of [ST] ("User Identification and Authentication") describes how the TOE establishes a user identity when an SSH client presents a public key. A key presented for SSH login must match that for the defined account in the TOE's database. If the asserted identity and password or key cannot be verified then the login fails and an audit record is generated.

FCS_SSHS_EXT.1.3

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

Section 5.2.5.1 of [ST] states the TOE drops packets greater than 256 kilobytes in SSH transport connections.

FCS_SSHS_EXT.1.4

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 5.2.5.1 of [ST] states the TOE does not support optional characteristics. The TOE uses the following encryption algorithms and rejects all others: aes128-ctr; aes256-ctr; aes128-gcm@openssh.com; and aes256-gcm@openssh.com. This description is consistent with the selections made in FCS_SSHS_EXT.1.4.

FCS_SSHS_EXT.1.5

Modified by TD0631

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

Section 5.2.5.1 of [ST] states the TOE supports the following public algorithms and rejects all others: ssh-rsa and ecdsa-sha2-nistp256. This list is consistent with the selections made in FCS_SSHS_EXT.1.5.

FCS_SSHS_EXT.1.6

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

Section 5.2.5.1 of [ST] states the TOE supports the following data integrity algorithms and rejects all others: hmac-sha2-256; hmac-sha2-512; and implicit GCM. This list is consistent with the selections made in FCS_SSHS_EXT.1.6.

FCS_SSHS_EXT.1.7

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

Section 5.2.5.1 of [ST] states the TOE supports the following key exchange algorithms and rejects all others: diffie-hellman-group14-sha1; ecdh-sha2-nistp256; ecdh-sha2-nistp384; and ecdh-sha2-nistp521. This list is consistent with the selections made in FCS_SSHS_EXT.1.7.

FCS_SSHS_EXT.1.8

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 5.2.5.1 of [ST] states within SSH connections, the same session keys are used for a threshold of no longer than one hour, and each encryption key is used to protect no more than one gigabyte of data. After any of the thresholds are reached, the TOE performs a rekey.

2.2.11.2 Guidance Activities

FCS_SSHS_EXT.1.4, FCS_SSHS_EXT.1.5

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 "FCS_SSHS_EXT.1 SSH Server Protocol" of [CCECG] contains instructions for configuring the TOE so that SSH conforms to the description in the TSS. No configuration is required to ensure that only the allowed encryption and public key algorithms are used in SSH connections with the TOE.

FCS_SSHS_EXT.1.6

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 "FCS_SSHS_EXT.1 SSH Server Protocol" of [CCECG] contains instructions for configuring the TOE so that SSH conforms to the description in the TSS. The administrator executes the `SSHSERVER HMAC` command to disable unapproved HMAC algorithms. No other configuration is required to ensure that only the allowed HMAC algorithms are used in SSH connections with the TOE.

FCS_SSHS_EXT.1.7

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 "FCS_SSHS_EXT.1 SSH Server Protocol" of [CCECG] describes use of the `SSHSERVER KEYEXCHANGE` command to disable the diffie-hellman-group-exchange-sha256 key exchange algorithm. No other configuration is required to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

FCS_SSHS_EXT.1.8

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 "FCS_SSHS_EXT.1 SSH Server Protocol" of [CCECG] states within SSH connections the same session keys are used for a threshold of no longer than one hour, and each encryption key is used to protect no more than one gigabyte of data. After either of the thresholds is reached, the TOE performs a rekey. No configuration is necessary or permitted to enforce this behavior.

2.2.11.3 Test Activities

FCS_SSHS_EXT.1.2

Modified in accordance with TD0631

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.

The evaluator demonstrated the successful use of all supported SSH public key algorithms (ssh-rsa, ecdsa-sha2-nistp256).

Modified in accordance with TD0631.

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.

The evaluator demonstrated that utilizing an unrecognized public key resulted in failed authentication.

Added in accordance with TD0631.

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.

The evaluator demonstrated that utilizing a correct password resulted in access being granted.

Added in accordance with TD0631.

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.

The evaluator demonstrated that utilizing an incorrect password resulted in access being denied.

FCS_SSHS_EXT.1.3

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

An SSH testing tool was used to send an improperly large packet to the TOE. It was demonstrated via wire capture that this packet and the connection were then dropped.

FCS_SSHS_EXT.1.4

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 verified that the TOE accepts SSH connections using all of the ciphers defined in the [ST]. The evaluator additionally verified that the TOE only advertised the list of ciphers defined in the [ST].

FCS_SSHS_EXT.1.5

Modified by TD0631

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.

The evaluator verified that the server supported all claimed host key algorithms by iteration.

Modified by TD0631

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.

The evaluator attempted to connect to the TOE while specifying an unsupported host key type. The evaluator verified that the connection failed.

FCS_SSHS_EXT.1.6

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.

The evaluator established a connection to the TOE using each of the MAC algorithms claimed in the [ST].

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 in 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.

The evaluator attempted to establish an SSH connection to the TOE using HMAC-MD5 and verified that the connection attempt was rejected.

FCS_SSHS_EXT.1.7

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.

The evaluator initiated an SSH connection using diffie-hellman-group1-sha1 key exchange algorithm and observed that the connection failed.

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.

The evaluator verified that an SSH connection to the TOE could be established using each of the key exchange methods claimed in the [ST].

FCS_SSHS_EXT.1.8

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.

The evaluator used an SSH test tool to send 1 GB of data. The TOE rekeyed the SSH connection prior to reaching the 1 GB threshold. Additionally, the TOE was shown to rekey an SSH connection after it had been kept open for 1 hour.

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.

The evaluator used an SSH test tool to send 1 GB of data. The TOE rekeyed the SSH connection prior to reaching the 1 GB threshold.

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/Functions).

The evaluator used an SSH test tool to hold an SSH connection open for 1 hour. The evaluator observed that the TOE rekeyed an SSH connection after it had been kept open for 1 hour. The rekey thresholds are not configurable.

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.

These cases did not occur during testing.

2.2.12 TLS Client Protocol without Mutual Authentication (FCS_TLSC_EXT.1)

2.2.12.1 TSS Activities

FCS_TLSC_EXT.1.1

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 5.2.5.2 of [ST] ("TLS Client Protocol") states the TOE's TLS client implementation supports the following ciphersuites:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492
- 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.

This list is identical to the ciphersuites selected in the SFR.

FCS_TLSC_EXT.1.2

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.

Section 5.2.5.2 of [ST] states the TOE's TLS client implementation establishes its reference identifiers from the administrator-configured reference identifiers per Section 6 of RFC 6125, using the hostname as a reference identifier and checking that the certificate presented by the external TLS server includes the specified identifier. The TOE does not support IP addresses but does support wildcards.

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.

The TOE is not distributed. Therefore, this activity is not applicable.

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 (RFC 5952 for IPv6, RFC 3986 for IPv4) is enforced.

Section 5.2.5.2 of [ST] states the TOE does not support IP addresses.

FCS_TLSC_EXT.1.4

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

Section 5.2.5.2 of [ST] states the TOE supports the Elliptic Curves Extension (specifying only P-256, P-384, and P-521) in its Client Hello, and the TOE does not require (nor allow) administrative configuration of this extension; the TOE always sends it.

2.2.12.2 Guidance Activities

FCS_TLSC_EXT.1.1

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 "FCS_TLSC_EXT.1 TLS Client Protocol without Mutual Authentication" of [CCECG] describes use of the `FIPSMODE ON` command, which limits the TOE to the set of ciphersuites specified in [ST].

FCS_TLSC_EXT.1.2

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.

Section "FCS_TLSC_EXT.1 TLS Client Protocol without Mutual Authentication" of [CCECG] states the TOE establishes its reference identifiers from the administrator-configured reference identifiers, using the hostname as the reference identifier. The reference identifier is configured using the `REMOTESYSLOG` CLI command, which is fully described in Section "FTP_ITC.1 Inter-TSF Trusted Channel" of [CCECG].

Section “FCS_TLSC_EXT.1 TLS Client Protocol without Mutual Authentication” of [CCECG] also states the TOE supports the SAN extension, but that the use of IP addresses is not supported.

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

The TOE is not distributed.

FCS_TLSC_EXT.1.4

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 5.2.5.2 of [ST] states the TOE supports the Elliptic Curves Extension (specifying only P-256, P-384, and P-521) in its Client Hello, and the TOE does not require (nor allow) administrative configuration of this extension; the TOE always sends it.

2.2.12.3 Test Activities

Modified in accordance with TD0670.

FCS_TLSC_EXT.1.1

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).

The evaluator verified via wire capture that a TLS connection could be established to the TOE using each of the ciphersuites claimed in the [ST].

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 field 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.

The evaluator verified via wire capture that the TOE successfully connects to a TLS server whose certificate contained the Server Authentication purpose in the extendedKeyUsage field, and that it would not connect to a TLS server whose certificate lacked the Server Authentication purpose.

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.

The evaluator configured a server to present an ECDSA ciphersuite while presenting an RSA certificate. The evaluator verified that the TOE rejected the connection to the server.

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.

a) The evaluator verified via wire capture that the TOE does not connect to a TLS server that selects the TLS_NULL_WITH_NULL_NULL ciphersuite.

b) The evaluator verified via wire capture that the TOE does not connect to a TLS server whose Server Hello message does not have a ciphersuite from the Client Hello message.

c) The evaluator verified via wire capture that the TOE does not connect to a TLS server that attempts to use the secp192r1 curve.

Test 5: The evaluator performs 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.

a) The evaluator verified that the TOE would not connect to a TLS server that attempted to connect using TLS 1.1.

b) The evaluator verified via wire capture that the TOE terminated a TLS connection after receiving a Server Key Exchange with a modified signature block.

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.

a) The evaluator verified via wire capture that the TOE rejected a TLS connection with a modified Server Finished handshake message.

b) The evaluator verified via wire capture that the TOE terminates a TLS connection if it receives a garbled message after the ChangeCipherSpec message.

c) The evaluator verified via wire capture that the TOE terminates a TLS connection after the Server Key Exchange if the server nonce in the Server Hello is modified.

FCS_TLSC_EXT.1.2

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:

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.

The evaluator verified via wire capture that the TOE rejects a TLS connection attempt if the TLS server's certificate has an incorrect CN and no SAN extension.

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.

The evaluator verified via wire capture that the TOE rejects a TLS connection attempt if the TLS server's certificate has a correct CN but an incorrect SAN value.

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.

The evaluator verified via wire capture that the TOE will connect to a TLS server whose certificate has a valid CN and no SAN extension.

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).

The evaluator verified via wire capture that the TOE will connect to a TLS server whose certificate has an incorrect CN but a valid value in the SAN extension.

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, URIID):

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.)

1) The evaluator verified via wire capture that the TOE rejects a server certificate that has a wildcard that is not in the leftmost identifier.

2) The evaluator verified via wire capture that the TOE accepts a server certificate with a wildcard in the leftmost identifier if the configured reference identifier has a single leftmost label. If the reference identifier has two left labels or none the TOE rejects the certificate.

Modified in accordance with TD0790.

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 addresses 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 an wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.0.1, CN=2001:0DB8:0000:0000:0008:0800:200C:* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

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.

The TOE does not claim to support the use of IP address identifiers in the CN or SAN, thus this test is not applicable.

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

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

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

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

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

The TOE does not claim to use the secure channel for FPT_ITT, thus this test is not applicable.

FCS_TLSC_EXT.1.3

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.

The evaluator verified via wire capture in FIA_X509_EXT.1.1 Test 1a that the TOE would accept a TLS server certificate that had been issued by an intermediate certificate that chained back to a root certificate in

the TOE's trust store, when all required intermediate CA certificates were presented to the TOE with the server certificate.

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.

The evaluator verified via wire capture in FIA_X509_EXT.1.1 Test 1b that the TOE rejects a server certificate when a complete chain back to a trusted root certificate cannot be completed. The TOE claims no override mechanisms and so no alternative behavior is expected or was observed throughout the evaluation for any other failures in certificate validation.

Test 3 [conditional]: 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.

The TOE does not claim any override mechanisms, thus this test is not applicable.

FCS_TLSC_EXT.1.4

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.

The evaluator verified via wire capture that the TOE would connect to a TLS server using each of the curves claimed in the [ST].

2.2.13 TLS Server Protocol without Mutual Authentication (FCS_TLSS_EXT.1)

2.2.13.1 TSS Activities

FCS_TLSS_EXT.1.1

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

Section 5.2.5.3 of [ST] ("TLS Server Protocol") states the TOE's TLS server implementation supports the same set of ciphersuites supported by the TOE's TLS client implementation (listed in section 5.2.5.2 of [ST], "TLS Client Protocol"). This list is identical to the set of ciphersuites specified in FCS_TLSS_EXT.1.1.

FCS_TLSS_EXT.1.2

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

Section 5.2.5.3 of [ST] states the TOE supports TLS 1.2 and denies requests to use all older TLS and SSL versions.

FCS_TLSS_EXT.1.3

Modified by TD0635

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

Section 5.2.5.3 of [ST] describes the following key agreement parameters of the server Key Exchange message: 2048 bit DHE; P-256, P-384, or P-521 elliptic curves.

FCS_TLSS_EXT.1.4

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

Section 5.2.5.3 of [ST] states the TOE does not support session resumption or session tickets.

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

The TOE's TLS implementation does not support session resumption based on session tickets.

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

The TOE's TLS implementation does not support session resumption based on session tickets.

Added in accordance with TD0569.

If the TOE claims a TLS server capable of session resumption (as a single context, or across multiple contexts), the evaluator verifies that the TSS describes how session resumption operates (i.e. what would trigger a full handshake, e.g. checking session status, checking Session ID, etc.). If multiple contexts are used the TSS describes how session resumption is coordinated across those contexts. In case session establishment and session resumption are always using a separate context, the TSS shall describe how the contexts interact with respect to session resumption (in particular regarding the session ID). It is acceptable for sessions established in one context to be resumable in another context.

Section 5.2.5.3 of [ST] states the TOE does not support session resumption.

2.2.13.2 Guidance Activities

FCS_TLSS_EXT.1.1

The evaluator shall check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

Section "FCS_TLSS_EXT.1 TLS Server Protocol without Mutual Authentication" of [CCECG] describes use of the `FIPSMODE ON` command, which limits the TOE to the set of ciphersuites specified in [ST].

FCS_TLSS_EXT.1.2

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

Section "FCS_TLSS_EXT.1 TLS Server Protocol without Mutual Authentication" of [CCECG] states the TOE supports TLS 1.2 server protocol without mutual authentication. Execution of the `FIPSMODE ON` command is the only administrative action required to configure the TOE to meet the requirement.

FCS_TLSS_EXT.1.3

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

Section "FCS_TLSS_EXT.1 TLS Server Protocol without Mutual Authentication" of [CCECG] states execution of the `FIPSMODE ON` command limits the key establishment for TLS to using 2048-bit DHE, and secp256r1, secp384r1, secp521r1 ECDHE curves and is the only administrative action required to configure the TOE to meet the requirement.

Added in accordance with TD0569.

FCS_TLSS_EXT.1.4

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

Section "FCS_TLSS_EXT.1 TLS Server Protocol without Mutual Authentication" of [CCECG] states the TOE's TLS server implementation does not support session resumption or session tickets.

2.2.13.3 Test Activities

FCS_TLSS_EXT.1.1

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).

The evaluator verified via the sslyze tool that the TOE would establish a connection using each of the ciphersuites claimed in the [ST].

Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

The evaluator verified via wire capture that the TOE would not open a TLS connection to a client that only had unsupported ciphersuites in the Client Hello and that it would not open a TLS connection to a client that only specified the TLS_NULL_WITH_NULL_NULL ciphersuite.

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

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

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

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

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

a) The evaluator verified via wire capture that the TOE rejects a TLS connection after receiving a Client Finished message with a modified byte.

b) The evaluator established a TLS connection with the server and verified via wire capture that the TOE indeed encrypted the TLS Finished record.

FCS_TLSS_EXT.1.2

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

The evaluator verified that the TOE rejects SSL 2.0, SSL 3.0, TLS 1.0 and TLS 1.1 connection attempts.

FCS_TLSS_EXT.1.3

Test 1 [conditional]: If ECDHE ciphersuites are supported:

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

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

a) The evaluator verified via wire capture that the TOE could establish a connection using each of the claimed elliptic curves and that it selected the same curve in the Server Key Exchange.

b) The evaluator verified via wire capture that the TOE would not accept a connection attempt using the secp192r1 curve.

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

The evaluator verified via wire capture that the TOE supports 2048 bit DH parameters.

Test 3 [conditional]: If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size.

The TOE does not claim support for any RSA key establishment ciphersuites, thus this test is not applicable.

Modified in accordance with TD0569.

FCS_TLSS_EXT.1.4

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

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

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

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

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps:

Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

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

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

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

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

The evaluator verified via wire capture and the sslyze tool that the TOE does not send session ID or session tickets to a client that requests them.

Modified in accordance with TD0569.

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

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

b) The evaluator shall initiate a handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then, within the same handshake, generate or force an unencrypted fatal Alert message immediately before the client would otherwise send its ChangeCipherSpec message thereby disrupting the handshake. The evaluator shall then initiate a new Client Hello using the previously captured session ID, and verify that the server (1) implicitly rejects the session ID by sending a ServerHello containing a different SessionID and performing a full handshake (as shown in figure 1 of RFC 4346 or RFC 5246), or (2) terminates the connection in some way that prevents the flow of application data.

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

The TOE does not claim to support session resumption using Session IDs, thus this test is not applicable.

Modified in accordance with TD0556 and TD0569.

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

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with an abbreviated handshake described in section 3.1 of RFC 5077 and illustrated with an example in figure 2. Of particular note: if the server successfully verifies the client's ticket, then it may renew the ticket by including a NewSessionTicket handshake message after the ServerHello in the abbreviated handshake (which is shown in figure 2). This is not required, however as further clarified in section 3.3 of RFC 5077.

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

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

The TOE does not claim to support session tickets, thus this test is not applicable.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication Failure Management (FIA_AFL.1)

2.3.1.1 TSS 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.t

Section 5.3.2 of [ST] (“Authentication Failure Management”) states, for password-based authentication, the administrator can configure the number of incorrect remote authentication attempts enforced on user accounts to a value between 1 and 65534 (with the default being 5). The TOE maintains a counter for each user account that it increments for each unsuccessful password-based authentication attempt. The counter is reset to zero on a successful authentication attempt. If an authentication attempt causes the counter to reach the configured threshold, for any supported mode of remote administration, the TOE will enforce an administrator-configurable (between 1 and 255 hours) lockout of the remote administrator. The admin can also unlock a user with the `REMLOCKEDUSER` CLI command.

The evaluator shall examine the TSS to confirm that the TOE ensures that authentication failures by remote administrators cannot lead to a situation where no administrator access is available, either permanently or temporarily (e.g. by providing local logon which is not subject to blocking).

Section 5.3.2 of [ST] states there can never be a case where all administrators are locked out, because the TOE does not subject SSH public key authentication to lockouts. The ST goes on to state that to ensure there is never a case where all administrators are locked out due to the session locking mechanism, at least one admin account must be set up to use SSH public key authentication.

2.3.1.2 Guidance 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.

Section "FIA_AFL.1 Authentication failure management" of [CCECG] provides instructions for configuring the number successive unsuccessful authentication attempts (using the `SETUSERLOGINATTEMPTS` CLI command) and lockout duration (using the `SETUSERLOCKOUTTIME` CLI command). It also provides instructions for the administrator to unlock a user account, using the `REMLOCKEDUSER` CLI command.

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 "FIA_AFL.1 Authentication failure management" of [CCECG] states there can never be a case where all administrators are locked out because authentication failure handling is only enforced on password credentials. User authentication based on SSH private key do not enforce a lockout mechanism. The guidance also states that at least one admin account must be set up to use SSH public key authentication.

2.3.1.3 Test 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):

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.

A lockout limit of three unsuccessful connection attempts was configured on the TOE. The evaluator verified that this resulted in a user account being blocked after three incorrect password entries, and valid passwords no longer worked.

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 verified that a different administrative account could be used to unlock the locked account and that valid credentials worked again. The evaluator additionally verified that the TOE will automatically unlock the account when a Security Administrator-configured time period has passed and that valid credentials worked again.

2.3.2 Password Management (FIA_PMG_EXT.1)

2.3.2.1 TSS Activities

Modified in accordance with TD0792.

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.

Section 5.3.1 of [ST] ("User Identification and Authentication") states the TOE supports the use in passwords of upper and lower case letters, numbers, symbols, punctuation, and space characters as defined by the following regular expression: ^[\p{L}\p{N}\p{Zs}\p{S}\p{P}]*$. The minimum length allowed for passwords is configurable in the range 6 to 128, with a default of 8.

2.3.2.2 Guidance 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 "FIA_PMG_EXT.1 Password Management") of [CCECG] states the TOE supports the use in passwords of upper and lower case letters, numbers, symbols, punctuation, and space characters as defined by the following regular expression: ^[\p{L}\p{N}\p{Zs}\p{S}\p{P}]*$. It provides the following guidance to administrators on the composition of strong passwords: the password must be a mix of upper- and lower-case characters, numbers, and special characters.

Section "FIA_PMG_EXT.1 Password Management") of [CCECG] provides instructions for setting the minimum password length using the `SETPASSWORDRULE` command, and states the minimum length allowed for passwords is configurable in the range 6 to 128, with a default of 8.

2.3.2.3 Test 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.

The evaluator verified that the TOE supported passwords with lengths of 8 and 15 characters, and passwords with all supported special characters.

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

The evaluator verified that the TOE enforced the configured length requirements and password complexity rules requiring at least one upper case character, one lower case letter, one number and one special character.

2.3.3 Protected Authentication Feedback (FIA_UAU.7)

2.3.3.1 TSS Activities

None defined.

2.3.3.2 Guidance 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.

Section “FIA_UAU.7 Protected Authentication Feedback” of [CCECG] states password data is obfuscated while it is being entered and only generic success/failure messages are provided. There are no preparatory steps required to ensure authentication data is not revealed while entering login information.

2.3.3.3 Test Activities

The evaluator shall perform the following test for each method of local login allowed:

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.

Because local access to the TOE is accomplished by connecting directly to a network port and using SSH, the tests for FCS_SSHS_EXT.1.2 tests 3 and 4 showed that no authentication feedback is provided when entering authentication information.

2.3.4 Password-based Authentication Mechanism (FIA_UAU_EXT.2)

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.

2.3.5 User Identification and Authentication (FIA_UIA_EXT.1)

2.3.5.1 TSS 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".

Section 5.3.1 of [ST] ("User Identification and Authentication") describes the logon process for each of the methods supported by the TOE: username and password to login via the local console connection and the remote web GUI; and username with password or public key for remote SSH. A successful logon occurs if the TOE can verify the credential (password or public key) provided with the username matches the credential held for that username in the TOE's database.

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.

Section 5.3.1 of [ST] states the only action the TOE allows before user identification and authentication is display of the warning banner.

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.

The TOE is not distributed. Therefore, this activity is not applicable.

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.

The TOE is not distributed. Therefore, this activity is not applicable.

2.3.5.2 Guidance 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 "FIA_UIA_EXT.1 User Identification and Authentication" of [CCECG] describes the process for logging in to the TOE remotely via the Web UI, and locally or remotely via the CLI.

No configuration is necessary to limit the services provided prior to logging in to the TOE.

2.3.5.3 Test 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:

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.

The following types of credentials and access methods are supported by the TOE:

- SSH Key (SSH) – The testing activity for FCS_SSHS_EXT.1.2 test 1 and 2 show that correct keys are accepted, while incorrect keys are rejected.
- Password (SSH) – The testing activity for FCS_SSHS_EXT.1.2 test 3 and 4 show that correct passwords are accepted, while incorrect passwords are rejected.
- Password (HTTPS) – The testing activity for FIA_AFL.1 tests 1 and 2 show that correct passwords are accepted (when an account lockout is not in place), while incorrect passwords are rejected.

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.

The evaluator verified that only the TOE access banner was available prior to authentication.

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.

Because the TOE accomplishes local login by connecting directly to a network port and using SSH, the results for this test are shown in the results for FIA_UIA_EXT.1 test 2.

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 is not distributed. Therefore, this activity is not applicable.

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

2.3.6.1 TSS 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).

Section 5.3.3 of [ST] ("X.509 Certificate Validation") states the TOE performs validation and revocation checks on certificates during TLS client connection establishment (i.e., when the TOE acts as a TLS client connecting to a remote syslog server). The TOE checks the validity of the server certificate and its chain, conformant to RFC 5280. This includes supporting a minimum path length of three certificates and the certification path terminating with a trusted CA certificate designated as a trust anchor. The TOE supports rules for extendedKeyUsage fields for TLS Server certificates (Sever Authentication purpose), TLS Client certificates (Client Authentication purpose), and OCSP certificates presented for OCSP responses (OCSP Signing purpose). The TOE does not use certificates for trusted updates or executable code integrity verification and therefore does not support rules for extendedKeyUsage fields of certificates used for trusted update or executable code integrity verification (Code Signing purpose).

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

Section 5.3.3 of [ST] states the TOE performs revocation checks on certificates presented to the TOE by the external syslog server and external authentication server, using OCSP as specified in RFC 6960.

2.3.6.2 Guidance 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 "FIA_X509_EXT.1/Rev X.509 Certificate Validation" of [CCECG] states the TOE performs validation and revocation checks on certificates during TLS client connection establishment (i.e., when the TOE acts as a TLS client connecting to a remote syslog server). It states the TOE does not use certificates for trusted updates or executable code integrity verification and therefore does not support rules for extendedKeyUsage fields of certificates used for trusted update or executable code integrity verification (Code Signing purpose).

Section "FIA_X509_EXT.1/Rev X.509 Certificate Validation" of [CCECG] additionally states the TOE performs OCSP revocation checking on certificates for the TOE's web server and for server certificates presented by remote syslog servers.

2.3.6.3 Test 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/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:

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

The evaluator verified the TOE successfully validated a certificate chain when all required certificates were present.

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.

The evaluator verified via wire capture that the TOE would not connect to an external TLS server whose certificate could not be validated due to a missing intermediate CA certificate.

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

The evaluator verified via wire capture that the TOE would not connect to an external TLS server whose certificate was expired.

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.

The evaluator established a certificate chain with OCSP responders defined and verified that the TOE accepted the chain when all certificates returned a valid result. The evaluator additionally verified that the certificate chain was rejected when either the server certificate or an intermediate CA was revoked.

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.

The evaluator verified via wire capture that the TOE would not accept an OCSP response from a responder whose certificate lacked the OCSP signing purpose.

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.)

The evaluator verified via wire capture that the TOE rejected a TLS server certificate whose first eight bytes had been modified.

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.)

The evaluator verified via wire capture that the TOE rejected a TLS server certificate whose last byte was modified.

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.)

The evaluator verified via wire capture that the TOE rejected a TLS server certificate whose public key had been modified.

Tests added in accordance with TD0527.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen):

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.

The evaluator loaded an EC root certificate into the TOE's trust store and verified that an intermediate EC CA certificate with a named curve was accepted.

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.

The evaluator verified that an intermediate EC CA certificate with explicitly specified curve parameters was rejected by the TOE.

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.

The evaluator attempted to import the intermediate CA certificates used in tests 8a and 8b to the TOE's trust store. The evaluator verified that the intermediate CA specified as a named curve could be added to the TOE's trust store and that the intermediate CA specified with explicit parameters could not be added to the TOE's trust store.

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).

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).

The evaluator verified via wire capture that the TOE would not connect to a TLS server whose certificate was issued by a CA that lacked the basicConstraints extension.

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 verified via wire capture that the TOE would not connect to a TLS server whose certificate was issued by a CA where the basicConstraints CA flag was set to false.

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).

The TOE only validates certificates for use in TLS.

2.3.7 X.509 Certificate Authentication (FIA_X509_EXT.2)

2.3.7.1 TSS 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.

Section 5.3.4 of [ST] ("X.509 Certificate Authentication") states the TOE uses X.509 certificates for authentication of TLS and HTTPS trusted channels. The TOE relies upon the administrator to load the CA certificates (root CA and any needed intermediate 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 5.3.4 of [ST] states during revocation checking, if the TOE cannot establish a connection to determine revocation status of the certificate being validated, the trusted channel will not be established. There is no administrator override.

2.3.7.2 Guidance 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 "FIA_X509_EXT.2 X.509 Certificate Authentication" of [CCECG] describes the configuration required on the TOE in order to validate server certificates presented by the external audit server. Section "FTP_ITC.1 Inter-TSF Trusted Channel" of [CCECG] describes the configuration required in the operating environment to enable the TOE to use certificates to authenticate the external audit server.

Section "FIA_X509_EXT.2 X.509 Certificate Authentication" of [CCECG] also states there is no administrative option when a connection cannot be established during the certificate validity check. In this circumstance, the TOE will not accept the certificate and the connection will not be established.

2.3.7.3 Test 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 verified via wire capture that the TOE will not connect to a TLS server when the OCSP responder cannot be contacted. There is no administrator configuration.

2.3.8 Certificate Requests (FIA_X509_EXT.3 X.509)

2.3.8.1 TSS Activities

If the ST author selects "device-specific information", the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

The ST does not select “device-specific information” in FIA_X509_EXT.3.1. Therefore, this activity is not applicable.

2.3.8.2 Guidance Activities

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

Section “FIA_X509_EXT.3 X.509 Certificate Requests” of [CCECG] provides instructions on requesting certificates from a CA, including generation of a certificate request using the `CREATECSR` command. The guidance includes instructions for establishing the Common Name, Organization, Organizational Unit, and Country fields as part of the `CREATECSR` command.

2.3.8.3 Test Activities

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

The evaluator verified that the TOE could generated a valid CSR.

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

The evaluator verified that the TOE would not import a certicate signed by a CA it did not recognize and that it would import a certificate signed by a CA that was in its trust store.

2.4 Security Management (FMT)

General requirements for distributed TOEs.

TSS

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.

Guidance Documentation

For distributed TOEs it is required to verify the Guidance Documentation to describe management of each TOE component. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

Tests

Tests defined to verify the correct implementation of security management functions shall be performed for every TOE component. For security management functions that are implemented centrally, sampling should be applied when defining the evaluator's tests (ensuring that all components are covered by the sample).

The TOE is not distributed.

2.4.1 Management of Security Functions Behavior (FMT_MOF.1/ManualUpdate)

2.4.1.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1 [of [SD-ND]]. There are no specific requirements for non-distributed TOEs.

The TOE is not distributed. Therefore, this activity is not applicable.

2.4.1.2 Guidance 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).

Section "FPT_TUD_EXT.1 Trusted Update" of [CCECG] describes the steps necessary to perform a manual update of the TOE. It also warns the administrator that all functions of the TOE will temporarily cease to operate, either shortly before the reboot caused by the update process, or because of the reboot. The TOE resumes full operation automatically once the install is completed.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.4.1.3 Test 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 shall fail.

The evaluator verified that a user without administrative privileges could not initiate a manual update.

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.

Per the EA, this test case is covered by FPT_TUD_EXT.1.

2.4.2 Management of Security Functions Behaviour (FMT_MOF.1/Services)

2.4.2.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1. For non-distributed TOEs, the evaluator shall ensure the TSS lists the services the Security Administrator is able to start and stop and how that how that operation is performed.

The TOE is not distributed.

Section 5.4.2 of [ST] ("Management of Security Functions Behavior") states that the ability to start and stop services, specifically to enable and disable export of audit records to an external server, is restricted to the role of Security Administrator. The TSS also states that the REMOTESYSLOG CLI command is used to enable and disable the export of audit records.

2.4.2.2 Guidance Activities

For distributed TOEs see chapter 2.4.1.2. For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the TSS lists the services the Security Administrator is able to start and stop and how that how that operation is performed.

The TOE is not distributed.

Section "FMT_MOF.1/Services Management of Security Functions Behavior" of [CCECG] states that the ability to start and stop services (specifically, to enable and disable export of audit records to an external syslog server using the REMOTESYSLOG CLI command) is restricted to users with the Administrator role. Section "FTP_ITC.1 Inter-TSF Trusted Channel" of [CCECG] provides guidance on the use of the REMOTESYSLOG CLI command.

2.4.2.3 Test Activities

The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) without prior authentication as Security Administrator (either by authenticating as a user with no administrator privileges, if possible, or without prior authentication at all). The attempt to enable/disable this service/these services 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 enable/disable this service/these services 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 enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) with prior authentication as Security Administrator. The attempt to enable/disable this service/these services should be successful.

The TOE is not distributed.

The evaluator verified that a non-Security Administrator could not start or stop the TOE's syslog export service and that a Security Administrator could start and stop the TOE's syslog export service.

2.4.3 Management of TSF Data (FMT_MTD.1/CoreData)

2.4.3.1 TSS 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.

Section 5.4.3 of [ST] ("Management of TSF Data") states the TOE does not provide any access to management functions prior to authentication.

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 5.4.3 of [ST] states the TOE restricts the ability to manage cryptographic keys to Security Administrators using role-based access control methods. Specifically, the Security Administrator may load certificates into the TOE's trust store.

2.4.3.2 Guidance 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.

Section "FMT_SMF.1: Specification of Management Functions" of [CCECG] identifies the TSF-data-manipulating functions implemented in response to the requirements of [NDcPP]. The TOE restricts access to these functions to users assigned to the 'Administrators' role defined by the TOE that corresponds to the Security Administrator role defined in [NDcPP].

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.

Section "FTP_ITC.1 Inter-TSF Trusted Channel" of [CCECG] provides information to the administrator to configure and maintain the TOE's trust store in a secure way, including commands to securely load CA certificates (`CERTIFICATE ADD`) and to designate a CA certificate as a trust anchor (`CERTIFICATE ADD ROOT`).

2.4.3.3 Test 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.

Per the EA, no separate testing is required because all management functions have been tested elsewhere.

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

2.4.4.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1 [of [SD-ND]].

The TOE is not distributed. Therefore, this activity is not applicable.

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 5.4.3 of [ST] ("Management of TSF Data") states the TOE provides the ability to manage cryptographic keys to Security Administrators. Specifically, the Security Administrator may use the TOE to generate CSRs (which contain key pairs) and load certificates (whether it is a certificate for the TOE generated by an external CA, or a certificate used to validate a presented TLS client or server certificate) into the TOE's trust store. The Security Administrator may additionally cause the TOE to generate SSH Host keys to be used by the TOE and import SSH public keys to associate with TOE user accounts.

2.4.4.2 Guidance Activities

For distributed TOEs see chapter 2.4.1.2 [of [SD-ND]].

The TOE is not distributed. Therefore, this activity is not applicable.

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

Section "FMT_MTD.1/CryptoKeys Management of TSF Data" of [CCECG] states the Security Administrator may use the TOE to generate Certificate Signing Requests (CSRs) and load certificates into the TOE's trust store.

2.4.4.3 Test 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 verified that a non-administrative user could not create a CSR or add certificates to the TOE's trust store.

The evaluator shall try to perform at least one of the related actions with prior authentication as Security Administrator. This attempt should be successful.

FIA_X509 tests demonstrated that an administrative user could add certificates to the TOE's trust store.

2.4.5 Specification of Management Functions (FMT_SMF.1)

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/ManualUpdate, FMT_MOF.1/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/Services, and FMT_MOF.1/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.

2.4.5.1 TSS Activities (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).

Section 5.4.1 of [ST] ("Security Roles and Specification of Management Functions") lists the management functions provided by the TOE, as specified in FMT_SMF.1. It states the TOE is administered via the Web UI and the CLI, and identifies through which of these interfaces each management function can be accessed. It also states the CLI can be accessed locally or remotely.

Section "FMT_SMF.1: Specification of Management Functions" of [CCECG] lists the management functions provided by the TOE and provides a reference to where the function can be exercised. The evaluator compared this list with the list of functions in FMT_SMF.1 and with the details provided in the TSS. The evaluator confirmed the guidance covers all the management functions specified in FMT_SMF.1 and described in the TSS.

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.

Section 5.4.1 of [ST] states the TOE provides administrative access through its HTTPS server and via a CLI. The CLI can be accessed remotely over SSH or locally by directing connecting a user laptop to the TOE's network port and using SSH. The web-based interface is accessed remotely from a web browser.

Section "FIA_UIA_EXT.1 User Identification and Authentication" of [CCECG] states the TOE provides the Security Administrator administrative access through its HTTPS server and via a CLI. The CLI can be accessed remotely over SSH or locally by directing connecting a user laptop to a network port and using an SSH client. The web-based interface is accessed remotely from a web browser.

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 behavior observed during testing of the configured SFRs is as described in the TSS and Guidance Documentation.

The TOE is not distributed. Therefore, this activity is not applicable.

2.4.5.2 Guidance Activities

See section 2.4.4.1 (of [SD-ND]).

The evaluation activities for the guidance are included in the above section.

2.4.5.3 Test Activities

The evaluator tests management functions as part of testing the SFRs identified in section 2.4.5. 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.

The evaluator verified that all management functions were tested elsewhere in the evaluation, thus no separate testing for this SFR was required.

2.4.6 Restrictions on Security Roles (FMT_SMR.2)

2.4.6.1 TSS 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 5.4.1 of [ST] ("Security Roles and Specification of Management Functions") states the TOE provides a number of roles for local and remote management of the TOE, of which only the 'Administrator' role corresponds to the Security Administrator as defined in [NDcPP].

2.4.6.2 Guidance 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.

Section "FIA_UIA_EXT.1 User Identification and Authentication" of [CCECG] describes how the administrator can connect to the local management interface and how the administrator can access the management console and Web UI remotely. This includes the need to use an SSH client to access the console and how to connect to the Web UI using a supported browser on a client workstation.

2.4.6.3 Test 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.

During the course of the evaluation, the evaluator utilized both the HTTPS and the SSH interface.

2.5 Protection of the TSF (FPT)

2.5.1 Protection of Administrator Passwords (FPT_APW_EXT.1)

2.5.1.1 TSS 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 5.5.1 of [ST] ("Protection of Administrator Passwords") states the TOE stores administrator passwords in a salted SHA-512 hash within an internal configuration file. The TOE does not provide interfaces for an administrator to view, extract, or read the password. The TOE only accepts passwords during authentication attempts (or during administrative changing of the password). The TOE salts and hashes a provided password and either compares it to the stored value as part of authentication, or stores the new value if the administrator is changing the administrative password.

2.5.1.2 Guidance Activities

None defined.

2.5.1.3 Test Activities

None defined.

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

2.5.2.1 TSS Activities

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

Section 5.2.4 of [ST] ("Cryptographic Key Destruction"), Table 9 ("Key Clearing") lists the private and symmetric keys used by the TOE and how they are stored. Section 5.2.4 states all keys, key material, and authentication credentials are protected from unauthorized disclosure. Keys stored in flash memory are stored encrypted using 256 bit AES in CBC mode. Section 5.5.2 of [ST] ("Protection of TSF Data (for reading of all pre-shared, symmetric and private keys") states that the TOE does not provide any interfaces to view

the SSH private host key and that it does not provide any commands to access the Web Server Certificate private key.

2.5.2.2 Guidance Activities

None defined.

2.5.2.3 Test Activities

None defined.

2.5.3 Reliable Time Stamps (FPT_STM_EXT.1)

2.5.3.1 TSS Activities

Modified in accordance with TD0632.

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 5.5.5 of [ST] ("Reliable Time Stamps") states the TOE utilizes time when creating audit records and when checking syslog and administrative client certificates (for expiration and revocation), for session inactivity timeout, and for administrator lock out periods. The TOE obtains and maintains time by allowing the administrator to both manually specify the time as well as configure up to three NTP servers with which the TOE synchronizes its clock. The TOE also has a battery-backed real-time clock that is used to guarantee the availability of time data. The TOE is not a virtualized TOE so it does not obtain time from any virtualization system.

2.5.3.2 Guidance Activities

Modified in accordance with TD0632.

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 "FCS_NTP_EXT.1 NTP Protocol" instructs the administrator how to set the time, either manually via the `TIMEDATE` CLI command, or by configuring the TOE to synchronize its time with up to three NTP servers. The guidance describes how the administrator establishes the communication path between the TOE and NTP servers, including the commands to configure the NTP client on the TOE. The guidance states the TOE supports NTP v4 and will use the version supported by the configured NTP servers. If the NTP server supports a version other than v4, the TOE will not establish a connection.

2.5.3.3 Test Activities

The evaluator shall perform the following tests:

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.

The evaluator verified that it was possible for an administrative user to set the time on the TOE.

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.

Testing for this requirement was performed in conjunction with testing for FCS_NTP_EXT.1.4 Test 1.

Added in accordance with TD0632.

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.

The TOE is not a virtualized TOE, therefore it does not obtain time from an underlying VS and this test is not applicable.

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.

The TOE is not distributed, thus this test is not applicable.

2.5.4 TSF Testing (FPT_TST_EXT.1)

2.5.4.1 TSS 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.

Section 5.5.3 of [ST] (“TSF Testing”) states the TOE runs cryptographic primitive Known Answer Tests and a software and firmware integrity verification test during initial start-up. Section 5.5.3 of [ST] describes how these tests are performed and what they actually do. These self-tests are sufficient to demonstrate correct operation of the TSF since they encompass the cryptographic functionality and the integrity of the entire TSF firmware executable code.

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.

The TOE is not distributed. Therefore, this activity is not applicable.

2.5.4.2 Guidance 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.

Section "FPT_TST_EXT.1 TSF Testing" of [CCECG] describes the TOE's power-up self-tests and the possible errors that may result from such tests. It states a failure of a power-up self-test causes the TOE to halt its boot and to reboot. The description in the guidance corresponds to the description of possible errors in the TSS. If the TOE reboots continually, instructions are provided to factory reset the device or the administrator can contact Crestron Support.

For distributed TOEs the evaluator shall ensure that the guidance documentation describes how to determine from an error message returned which TOE component has failed the self-test.

The TOE is not distributed. Therefore, this activity is not applicable.

2.5.4.3 Test 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 fulfil 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.

The evaluator found audit records verifying that the TOE performs appropriate self-tests upon startup.

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 is not distributed. Therefore, this activity is not applicable.

2.5.5 Trusted Update (FPT_TUD_EXT.1)

2.5.5.1 TSS 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.

Section 5.5.4 of [ST] ("Trusted Update") states the TOE displays the currently executing version of firmware on the **STATUS** tab in the navigation bar of the web GUI. The TOE does not provide the capability to install a trusted update on the TOE with a delayed activation.

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.

Section 5.5.4 of [ST] describes mechanisms for updating the system firmware using either the web GUI or the CLI.

The administrator can manually initiate updates via the **Upload Firmware File** radio button of the web GUI. To initiate the firmware update, the administrator first locates the firmware file on the vendor support site and downloads it to the `/firmware` directory on the TOE appliance. The TOE automatically verifies the digital signature on the update files during the download process and only installs the updates if the signatures verify. The web UI does not provide a means to download an update for installation at a later date (i.e., a delayed activation).

The administrator can also update the TOE manually from the CLI. The administrator first obtains the update file from the vendor's website using SFTP and downloads it to the `/firmware` directory on the TOE appliance. It is possible to upload a file and install it at a later date. The file is not active until it is installed by the administrator and, once installed, it is immediately activated. Updates installed via this mechanism are verified at installation. If the digital signatures cannot be verified, then the TOE will not perform the update and the failure will be audited. There are no specific commands for viewing the version of the downloaded but uninstalled firmware, but the version of the firmware is identified in the filename and can be queried by navigating to the firmware directory and viewing the filename.

When an update fails, the device remains in the current version and this can be verified by running the `ver -v` CLI command. All failures are audited.

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.

FPT_TUD_EXT.1.2 does not include 'support automatic checking for updates' or 'support automatic updates'. Therefore, this assurance activity is not applicable.

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.

The TOE is not distributed. Therefore, this assurance activity is not applicable.

If a published hash is used to protect the trusted update mechanism, then the evaluator shall verify that the trusted update mechanism does involve an active authorization step of the Security Administrator, and that download of the published hash value, hash comparison and update is not a fully automated process involving no active authorization by the Security Administrator. In particular, authentication as Security Administration according to FMT_MOF.1/ManualUpdate needs to be part of the update process when using published hashes.

The TOE uses a digital signature mechanism to protect the TOE update, not a published hash. Therefore, this assurance activity is not applicable.

2.5.5.2 Guidance 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.

Section "FPT_TUD_EXT.1 Trusted Update" of [CCECG] states the administrator can query the currently executing version of the TOE from the **STATUS** tab in the navigation bar of the web GUI or by executing the `version` CLI command.

Section "FPT_TUD_EXT.1 Trusted Update" of [CCECG] states the administrator can update the TOE manually from the CLI, in which case the administrator first obtains the update file from the vendor's website using SFTP and downloads it to the `/firmware` directory on the TOE appliance. It is possible to upload a file and install it at a later date. The file is not active until it is installed by the administrator and, once installed, it is immediately activated. There are no specific commands for viewing the version of the downloaded but uninstalled firmware, but the version of the firmware is identified in the filename and can be queried by navigating to the firmware directory and viewing the filename.

Section "FPT_TUD_EXT.1 Trusted Update" of [CCECG] states the administrator can also update the TOE manually from the GUI. The administrator can manually initiate updates to the TOE firmware through the GUI **Upload Firmware File** radio button. To initiate the firmware upgrade, the administrator must first obtain the firmware file by loading it onto the device and clicking OK. From the web interface, it is not possible to download an update and then install it at a later date.

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.

Section "FPT_TUD_EXT.1 Trusted Update" of [CCECG] states authenticity of TOE updates is verified using a digital signature mechanism. For trusted updates performed using the web GUI, the update files are verified as they are loaded onto the TOE and are installed only if the authenticity of the update is verified via its digital signature. For updates performed using the CLI, the TOE verifies the authenticity of the update files at installation time (rather than when they are first loaded onto the TOE).

If the digital signature cannot be verified in any update operation (Web GUI or CLI), then the TOE will not perform the update and will log the failure. The description in [CCECG] corresponds with the description in [ST].

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.

The TOE uses a digital signature mechanism to protect the TOE update, not a published hash. Therefore, this assurance activity is not applicable.

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.

The TOE is not distributed. Therefore, this assurance activity is not applicable.

If this was information 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.

The TOE is not distributed. Therefore, this assurance activity is not applicable.

If this was information was not provided in the TSS: If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the Guidance Documentation contains a description of how the certificates are contained on the device. The evaluator also ensures that the Guidance Documentation describes how the certificates are installed/updated/selected, if necessary.

The TOE does not use a certificate-based mechanism for software update digital signature verification. Therefore, this assurance activity is not applicable.

2.5.5.3 Test Activities

The evaluator shall perform the following tests:

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.

The evaluator checked the TOE's version, ran an update with a legitimate update file, and then checked the version again. The TOE was verified to have been updated to the new version.

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.

The evaluator modified a legitimately signed update and verified that the update was rejected by the TOE.

The evaluator removed the signature file from a TOE update package and verified that the update was rejected.

The evaluator modified the signature file in a TOE update package and verified that the update was rejected.

The TOE does not allow delayed activation of updates so that portion of the test is not applicable.

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.

The TOE uses digital signatures and not published hashes to verify updates, thus this test is not applicable

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 TOE uses digital signatures and not published hashes to verify updates, thus this test was 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).

The TOE only claims support for manual updates in the evaluated configuration, thus this test was performed against the manual update interface.

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

The TOE is not distributed. Therefore, this test activity is not applicable.

2.6 TOE Access (FTA)

2.6.1 TSF-initiated Termination (FTA_SSL.3)

2.6.1.1 TSS Activities

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

Section 5.6.2 of [ST] (“Session Termination”) states the TOE terminates remote interactive sessions when an administrator-configurable inactive timeout value is reached. For remote access to the CLI, the default

inactivity timeout is 20 minutes, and is configurable to values between 1 and 60 minutes. For remote access to the Web UI, the default inactivity timeout is 1200 seconds (20 minutes) and is configurable to values between 600 and 3600 seconds (10 minutes and 60 minutes respectively).

2.6.1.2 Guidance Activities

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

Section "FTA_SSL.3 TSF-Initiated Termination" of [CCECG] includes instructions for configuring the inactivity time period for remote administrative session termination. The instructions cover remote access to the CLI via SSH, using the `SETLOGOFFIDLETIME` CLI command, and access to the Web UI, using the `WEBSERVER TIMEOUT` CLI command.

2.6.1.3 Test Activities

For each method of remote administration, the evaluator shall perform the following test: **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 verified that the TOE enforced 10 and 20 minute timeout periods.

2.6.2 User-initiated Termination (FTA_SSL.4)

2.6.2.1 TSS Activities

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

Section 5.6.2 of [ST] ("Session Termination") states the administrator can terminate their administrative session from the CLI (locally or remotely) using the '`BYE`' command. From the Web UI, the administrator can click on the profile icon/image in the upper right and select "Sign Out".

2.6.2.2 Guidance Activities

The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Section "FTA_SSL.4 User-Initiated Termination" of [CCECG] states how to terminate both local and remote interactive sessions. From the CLI (local or remote), the administrator enters the `BYE` command. From the Web UI (remote only), the administrator can click on the profile icon/image in the upper right and select "Sign Out".

2.6.2.3 Test Activities

For each method of remote administration, the evaluator shall perform the following tests: **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.

The TOE accomplishes local access by connecting directly to a network port and utilizing SSH. Thus, the evidence for this test is found in the evidence for FTA_SSL.4 test 2.

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.

The evaluator verified that the BYE command would terminate an SSH session.

The evaluator verified that the GUI's logout button terminated a session in the management GUI.

2.6.3 TSF-initiated Session Locking (FTA_SSL_EXT.1)

2.6.3.1 TSS 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 5.6.2 of [ST] ("Session Termination") states the TOE terminates local interactive sessions when an administrator-configurable inactive timeout value is reached. The default inactivity timeout is 20 minutes, and is configurable to values between 1 and 60 minutes.

2.6.3.2 Guidance 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 "FTA_SSL_EXT.1 TSF-Initiated Session Locking" of [CCECG] states the TOE terminates local interactive sessions when the administrator-configurable inactive timeout value is reached. It also provides instructions for configuring the inactivity timeout value, using the `SETLOGOFFIDLETIME` CLI command.

2.6.3.3 Test Activities

The evaluator shall perform the following test.

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 re-authentication is needed when trying to unlock the session.

The TOE accomplishes local access by connecting directly to a network port and utilizing SSH. Thus, the evidence for this test is found in the evidence for FTA_SSL.3 Test 1. In that test the evaluator verified that a session with the CLI is terminated after the Security Administrator-determined period of inactivity.

2.6.4 Default TOE Access Banners (FTA_TAB.1)

2.6.4.1 TSS Evaluation Activity

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 5.4.1 of [ST] ("Security Roles and Specification of Management Functions") states the TOE provides administrative access through its HTTPS server and via a CLI. The CLI can be accessed remotely over SSH or locally by directing connecting a user laptop to a network port and using SSH. The web-based interface is accessed remotely from a web browser.

Section 5.6.1 of [ST] ("Access Banner") states the TOE provides the administrator the ability to set a banner that the TOE displays before each login on all interfaces.

2.6.4.2 Guidance Activities

The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Section "FTA_TAB.1 Default TOE Access Banners" of [CCECG] describes how to configure the banner message. The guidance instructs the administrator to use an SFTP client to copy a text file containing the text the administrator wishes to display onto the TOE appliance. The text file must be called `banner.txt` and placed in the `/SSHBanner` directory.

2.6.4.3 Test Activities

The evaluator shall also perform the following test:

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 message and verified that it was displayed by the TOE.

2.7 Trusted Path/Channels (FTP)

2.7.1 Inter-TSF Trusted Channel (FTP_ITC.1)

2.7.1.1 TSS 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 5.7 of [ST] ("Trusted Path/Channels") states the TOE can be configured to export audit records to an external syslog server over TLS.

Section 5.2.5 of [ST] ("Cryptographic Protocols") states the TOE acts as a TLS client when exporting audit records to the external audit server. Section 5.2.5.2 of [ST] ("TLS Client Protocol") specifies the supported cipher suites, elliptic curves, and reference identifiers that can be used for the audit server connection. The TOE does not support mutual authentication or IP addresses for audit server connections. Section 5.3.3 of [ST] ("X.509 Certificate Validation") specifies how the TOE validates certificate presented by the audit server when it is presented as part of the TLS connection used for audit record export.

Section 5.7 of [ST] states the administrator can specify the root CA and any needed intermediate CA certificates for the TOE to use when validating the certificate presented by the external audit server.

2.7.1.2 Guidance 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 ("FTP_ITC.1 Inter-TSF Trusted Channel") of [CCECG] provides instructions to the administrator for establishing TLS communications between the TOE and an external syslog server. It also provides detailed recovery instructions should the connection to the external syslog server be unintentionally broken.

2.7.1.3 Test 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:

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.

Evidence for this can be seen in the results for FCS_TLSC_EXT.1.1 test 1. The channel covered by this requirement is TLS, used to connect to an external audit server. In that test, the evaluator verified that the syslog over TLS connection could be successfully established to a TLS server.

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.

Evidence for this can be seen in the results for FTP_ITC.1 test 4. In that test, the evaluator verified that the syslog over TLS connection could be successfully established to a TLS server. The communication was in fact initiated by the TOE.

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

Evidence for this can be seen in the results for FTP_ITC.1 test 4. In that test, the evaluator verified that the syslog over TLS connection did not transmit its data in plaintext.

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.

The TOE connected over TLS to an audit server. The evaluator physically interrupted the network communication (at an intermediate switch level, so that the TOE could not detect a cable being unplugged) between the devices for a short period of time (sufficient to interrupt the network link layer, but not the application layer timeout) and observed that the connection was resumed when the connection was re-established. The evaluator physically interrupted the network connection for a long period of time (sufficient to both interrupt the network layer and exceed the application layer timeout). The evaluator observed that a new TLS session was initiated when the connection was re-established and that no data was 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 external secure channels to TOE components in the Security Target.

The TOE is not distributed. Therefore, this activity is not applicable.

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.

2.7.2 Trusted Path (FTP_TRP.1/Admin)

2.7.2.1 TSS 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 5.7 of [ST] (“Trusted Path/Channels”) states the administrator can connect to the TOE using HTTPS or SSH protected administration channels. This is consistent with the selections made in FTP_TRP.1/Admin. Furthermore, the corresponding protocol requirements (FCS_HTTPS_EXT.1, FCS_SSHS_EXT.1) are included in the ST.

2.7.2.2 Guidance Activities

The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Section “FTP_TRP.1/Admin Trusted Path” of [CCECG] provides instructions for establishing a remote administrative session over HTTPS by opening a browser on the administrator’s workstation or laptop and navigating to the applicable URL to access the TOE’s web server GUI.

Section “FTP_TRP.1/Admin Trusted Path” of [CCECG] also provides instructions for establishing a remote administrative session over SSH by starting up an SSH client on the administrator’s workstation or laptop and connecting to the TOE using a configured administrative account and the TOE’s hostname or IP address.

Additionally, section “FIA_UIA_EXT.1 User Identification and Authentication” of [CCECG] describes how the administrator can access the management console and Web UI remotely. This includes the need to use an SSH client to remotely access the console using a configured administration account and the TOE’s hostname or IP address, and how to connect to the Web UI using a supported browser on a client workstation and entering the TOE’s IP address.

2.7.2.3 Test Activities

The evaluated shall perform the following tests.

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.

Evidence for this can be seen from FCS_NTP_EXT.1.4 test 1 and FCS_SSHS_EXT.1.2 Test 1. In those tests the evaluator accessed the HTTPS interface (for the NTP test) and the SSH interface (for the SSHS test). In both cases, the interfaces could be used successfully.

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

The evaluator accessed the HTTPS interface while capturing packets and verified that no data was sent in plaintext.

Evidence for the SSH interface can be seen in the evidence for FCS_SSHS_EXT.1.4. In that test, a connection was established with the TOE and no data was seen to be sent in plaintext.

Further assurance activities are associated with the specific protocols.

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

The TOE is not distributed. Therefore, this activity is not applicable.

3 Security Assurance Requirements

3.1 Class ASE: Security Targeted Evaluation

General ASE

When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2 (Evaluation Activities for SFRs).

3.1.1 ASE_TSS.1 TOE Summary Specification for Distributed TOEs

For distributed TOEs only the SFRs classified as 'all' have to be fulfilled by all TOE parts. The SFRs classified as 'One' or 'Feature Dependent' only have to be fulfilled by either one or some TOE parts, respectively. To make sure that the distributed TOE as a whole fulfills all the SFRs the following actions for ASE_TSS.1 have to be performed as part of ASE_TSS.1.1E.

Note that additional Evaluation Activities for the TSS in the case of a distributed TOE are defined in section A.9.1.1 in the SD.

The TOE is not a distributed TOE. Therefore, this activity is not applicable.

3.2 Class ADV: Development

3.2.1 ADV_FSP.1 Basic Functional Specification

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 3.

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.

3.2.1.1 ADV_FSP.1 Evaluation Activity

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.

Through review of [ST] and [CCECG], the evaluation team identified that the following external interfaces are security relevant:

- Web UI
- CLI
- TLS logical interface
- SSH logical interface
- Syslog interface.

The evaluation team determined the interface documentation described the purpose and method of use for each TSFI identified as being security relevant, sufficient to enable each of the evaluation activities to be completed satisfactorily. The evaluation team's results from performing the evaluation activities are documented in Section 2 of this AAR.

3.2.1.2 ADV_FSP.1 Evaluation Activity

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 evaluation team determined the interface documentation identified and described the parameters for each TSFI identified as being security relevant, sufficient to enable each of the evaluation activities to be completed satisfactorily. The evaluation team's results from performing the evaluation activities are documented in Section 2 of this AAR.

3.2.1.3 ADV_FSP.1 Evaluation Activity

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

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

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

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

The evaluation team examined the interface documentation and was able to map interfaces to SFRs, sufficient to enable each of the evaluation activities to be completed satisfactorily. The evaluation team's results from performing the evaluation activities are documented in Section 2 of this AAR.

3.3 Class AGD: Guidance Documents

It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to Security Administrators and users (as appropriate) as part of the TOE.

Note that additional Evaluation Activities for the guidance documentation in the case of a distributed TOE are defined in section A.9.1.1 in the SD.

3.3.1 AGD_OPE.1 Operational User Guidance

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.

3.3.1.1 AGD_OPE.1 Evaluation Activity

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

The [CCECG] is published with the Security Target at the https://www.niap-ccevs.org/ website. The distribution of the documentation provides a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

3.3.1.2 AGD_OPE.1 Evaluation Activity

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.

Section “Introduction” of [CCECG] identifies all of the platforms claimed for the TOE in section 1.1 of [ST] (“Security Target, Target of Evaluation, and Common Criteria Identification”).

The [CCECG] provides the following information regarding the Operational Environment for all platforms claimed in the Security Target in the identified document sections: scope of evaluation (“Scope of Evaluation”); required configuration (“Configuration Prerequisites”); assumptions (“Evaluation Assumptions”); and operational environment that the system must be in to ensure a secure deployment (“Introduction”).

3.3.1.3 AGD_OPE.1 Evaluation Activity

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.

Section “Configuration Prerequisites” of [CCECG] provides instructions for configuring the Crestron Crypto Kernel for Open SSL cryptographic engine included with the TOE. The `FIPSMODE ON` command ensures

the TOE runs in a FIPS-compliant mode, using only the approved cryptographic algorithms and ciphersuites identified in [ST]. This section of [CCECG] also states use of other cryptographic engines was not evaluated nor tested during the CC evaluation of Crestron Digital Media NVX Series v7.1.

3.3.1.4 AGD_OPE.1 Evaluation Activity

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.

Section "Scope of Evaluation" of [CCECG] lists the functionality that was excluded from evaluation and makes it explicitly clear that only the functionality claimed in [ST] was evaluated.

3.3.1.5 AGD_OPE.1 Evaluation Activity

Modified in accordance with TD0536.

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 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:

   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.

Part a) is addressed by section 3.3.1.3 above.

For part b), section "FPT_TUD_EXT.1 Trusted Update" of [CCECG] describes the update process, including the instructions for obtaining TOE updates and for making updates available to the TOE, and instructions for initiating the update process and for discerning whether the process was successful or unsuccessful.

Part c) is addressed by section 3.3.1.4 above.

3.3.2 AGD_PRE.1 Preparative Procedures

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 Security Administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that Security 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.

3.3.2.1 AGD_PRE.1 Evaluation Activity

The evaluator shall examine the Preparative procedures to ensure they include a description of how the Security 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).

Section "Evaluation Assumptions" of [CCECG] identifies the assumptions that state the specific conditions that are expected to be met by the operational environment and/or administrators.

The [CCECG] provides the following information regarding the Operational Environment for all platforms claimed in the Security Target in the identified document sections: scope of evaluation ("Scope of Evaluation"); required configuration ("Configuration Prerequisites"); assumptions ("Evaluation Assumptions"); and operational environment that the system must be in to ensure a secure deployment ("Introduction").

3.3.2.2 AGD_PRE.1 Evaluation Activity

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.

Section "Introduction" of [CCECG] identifies all of the platforms claimed for the TOE in section 1.1 of [ST] ("Security Target, Target of Evaluation, and Common Criteria Identification"). The instructions and guidance contained in the [CCECG] are applicable to all TOE platforms.

3.3.2.3 AGD_PRE.1 Evaluation Activity

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

Section "About this Guide" of [CCECG] states it is intended for administrators responsible for installing, configuring and operating the TOE, and provides guidance allowing an administrator to deploy the product in an environment consistent with the evaluated configuration. Section "Installation Guidance" of [CCECG] identifies the applicable Quick Start Guide for each appliance model included in the TOE. Each Quick Start Guide provides instructions enabling the administrator to successfully install the appliance, comprising: mounting the device on a flat surface or an equipment rack, or installing the device in a card chassis for models with a chassis card form factor; connecting the device; and configuring the device.

3.3.2.4 AGD_PRE.1 Evaluation Activity

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.

[CCECG] provides instructions for managing the security of the TOE both as a product and as a component of the larger operational environment.

3.3.2.5 AGD_PRE.1 Evaluation Activity

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.

Sections "FMT_MTD.1/CoreData Management of TSF Data" and "FMT_SMF.1: Specification of Management Functions" of [CCECG] include instructions to provide a protected administrative capability. Section "FIA_UIA_EXT.1 User Identification and Authentication" states that there are no default passwords on the TOE. Instead, when the device is powered up for the first time, it requires that an admin account be created. The admin can choose this account name and password.

3.4 Class ALC: Life-Cycle Support

3.4.1 ALC_CMC.1 Labelling of the TOE

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 evaluation team verified this through the completion of the ALC_CMC.1 work units described in the CEM. The results of this analysis are included in the proprietary ETR produced by the laboratory.

3.4.2 ALC_CMS.1 TOE CM Coverage

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

The evaluation team verified this through the completion of the ALC_CMS.1 work units described in the CEM. The results of this analysis are included in the proprietary ETR produced by the laboratory.

3.5 Class ATE: Tests

3.5.1 ATE_IND.1 Independent Testing – Conformance

3.5.1.1 ATE_IND.1 Assurance Activity

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 A [in the SD] 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 A.9.3.1 in the SD.

The evaluator developed a test plan to list all individual test activities for the TOE based on claimed SFRs and subsequently exercised all the test cases. The tests were selected in order to ensure that each of the test assertions specified in *Evaluation Activities for Network Device cPP* were covered. All tests passed.

The TOE consists of multiple variants. The evaluation lab provided detailed equivalency arguments in the test report that discusses which TOE variant was tested and why this subset of TOE models can reasonably be expected to cover the full set of variants covered by the evaluation.

Testing was accomplished using a variety of tools. OpenSSL's s_server and s_client tools, sslyze analyzer, and nmap were utilized for TLS testing along with Leidos proprietary testing tools for packet modification. OpenSSH's ssh client and WinSCP were utilized for SSH testing along with Leidos proprietary tools for rekey testing.

The TOE is not distributed, so additional evaluation activities related to those TOEs are not applicable.

3.6 Class AVA: Vulnerability Assessment

3.6.1 AVA_VAN.1 Vulnerability Survey

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 in the SD, while an "outline" of the assurance activity is provided below.

3.6.1.1 AVA_VAN.1 Evaluation Activity (Documentation)

Modified in accordance with TD0547.

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[1] that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE), 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.

Section 1.6.1 of [ST] ("Physical Scope") identifies the components that compose the TOE. The information lists the processor used by the TOE, the operating system environment in which the TOE application executes, and the independently identifiable and reusable (outside the TOE) third party components included in the TOE. Additionally, a list of third-party software components was provided in Table 2 of that section.

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 3.4.1.2 and 3.5.1.2.

The TOE is not distributed.

3.6.1.2 AVA_VAN.1 Evaluation Activity

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

The evaluation team performed a search of the following public vulnerability database:

- National Vulnerability Database (https://nvd.nist.gov/).
- Crestron Security Advisories (https://www.crestron.com/Security#securityAdvisoriesTab).

Searches were performed on 15 August 2024 and 3 October 2024, using the following search terms:

- Crestron
- Crestron DM-NVX
- DM-NVX-350
- DM-NVX-350C

[1] In this sub-section the term "components" refers to parts that make up the TOE. It is therefore distinguished from the term "distributed TOE components", which refers to the parts of a TOE that are present in one physical part of a distributed TOE. Each distributed TOE component will therefore generally include a number of the hardware and software components that are referred to in this sub-section: for example, each distributed TOE component will generally include hardware components such as processors and software components such as an operating system and libraries.

- DM-NVX-351
- DM-NVX-351C
- DM-NVX-352
- DM-NVX-352C
- DM-NVX-360
- DM-NVX-360C
- DM-NVX-363
- DM-NVX-363C
- DM-NVX-E10
- DM-NVX-E20
- DM-NVX-E20-2G
- DM-NVX-E30C
- DM-NVX-E30
- DM-NVX-D10
- DM-NVX-D20
- DM-NVX-D30
- DM-NVX-D30C
- DM-NVX-D80-IOAV
- DM-NVX-D200
- DM-NVX-E760
- DM-NVX-E760C
- Intel Arria 10 SX SoC FPGA
- ARM Cortex-A9 MPCore
- Lighttpd 1.4.52
- Redis v5.0.14
- openbsd openssh 9.8p1
- Net-SNMP 5.9.4
- OpenSSL 1.0.2zd
- NTPSec 1.2.3
- Angstrom Linux
- Crestron AV Router.

The evaluators additionally performed testing as specified in sections A.1.2 through A.1.4 of [ND_SD].

The evaluation team determined that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.