NETSCOUT Systems, Inc.

nGeniusONE with InfiniStreamNG v6.3.3

Assurance Activity Report

Version 1.14

April 4, 2024

Document prepared by

www.lightshipsec.com

Table of Contents

1 INTRODUCTION 3
1.1 Evaluation Identifiers 3
1.2 Evaluation Methods 3
1.3 Reference Documents 6

2 EVALUATION ACTIVITIES FOR SFRS 7
2.1 Security Audit (FAU) 7
2.2 Cryptographic Support (FCS) 12
2.3 Identification and Authentication (FIA) 28
2.4 Security management (FMT) 34
2.5 Protection of the TSF (FPT) 40
2.6 TOE Access (FTA) 49
2.7 Trusted path/channels (FTP) 53

3 EVALUATION ACTIVITIES FOR OPTIONAL REQUIREMENTS 57
3.1 Identification and Authentication (FIA) 57
3.2 Protection of the TSF (FPT) 62
3.3 Communication (FCO) 64

4 EVALUATION ACTIVITIES FOR SELECTION-BASED REQUIREMENTS 69
4.1 Security Audit (FAU) 69
4.2 Cryptographic Support (FCS) 71
4.3 Identification and Authentication (FIA) 103
4.4 Security management (FMT) 111

5 EVALUATION ACTIVITIES FOR SECURITY ASSURANCE REQUIREMENTS 117
5.1 ASE: Security Target 117
5.2 ADV: Development 118
5.3 AGD: Guidance 119
5.4 ALC: Life-cycle Support 122
5.5 ATE: Tests 122

6 VULNERABILITY ASSESSMENT 124

1 Introduction

1 This Assurance Activity Report (AAR) documents the evaluation activities performed by Lightship Security for the evaluation identified in Table 1. The AAR is produced in accordance with National Information Assurance Program (NIAP) reporting guidelines.

1.1 Evaluation Identifiers

Table 1: Evaluation Identifiers

Scheme	Canadian Common Criteria Scheme
Evaluation Facility	Lightship Security
Developer/Sponsor	NETSCOUT Systems, Inc.
TOE	NETSCOUT nGeniusONE with InfiniStreamNG v6.3.3 Builds: • nGeniusONE v6.3.3 build 1154 • InfiniStreamNG: v6.3.3 build 863 Patches: nGeniusONE: • ATSFX64_34W_54L_53_90058_02zg_208 • nG1_6x-STIG-10JAN2022_v1
Security Target	NETSCOUT nGeniusONE with InfiniStreamNG v6.3.3 Security Target, v1.25, April 2024
Protection Profile	collaborative Protection Profile for Network Devices, Version 2.2e, 23-March-2020

1.2 Evaluation Methods

2 The evaluation was performed using the methods and standards identified in Table 2.

Table 2: Evaluation Methods

Evaluation Criteria	CC v3.1R5
Evaluation Methodology	CEM v3.1R5
Supporting Documents	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP, December-2019, Version 2.2

Interpretations	TD #	Name	Applicability	Exclusion Rationale
	TD0527	Updates to Certificate Revocation Testing (FIA_X509_EXT.1)	Yes	
	TD0528	NIT Technical Decision for Missing EA's for FCS_NTP_EXT.1.4	No	FCS_NTP_EXT.1 not claimed
	TD0536	NIT Technical Decision for Update Verification Inconsistency	Yes	
	TD0537	NIT Technical Decision for Incorrect Reference to FCS_TLSC_EXT.2.3	No	FCS_TLSC_EXT.2 not claimed
	TD0546	NIT Technical Decision for DTLS – clarification of Application Note 63	No	FCS_DTLS not claimed
	TD0547	NIT Technical Decision for Clarification on developer disclosure of AVA_VAN	Yes	
	TD0555	NIT Technical Decision for RFC Reference incorrect in TLSS Test	Yes	
	TD0556	NIT Technical Decision for RFC 5077 question	Yes	
	TD0563	NiT Technical Decision for Clarification of Audit Date Information	Yes	
	TD0564	NiT Technical Decision for Vulnerability Analysis Search Criteria	Yes	
	TD0569	NiT Technical Decision for Session ID Usage Conflict in FCS_DTLSS_EXT.1.7	Yes	
	TD0570	NiT Technical Decision for	Yes	

	Clarification about FIA_AFL.1		
TD0571	NiT Technical Decision for Guidance on How to Handle FIA_AFL.1	Yes	
TD0572	NiT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers	Yes	
TD0580	NiT Technical Decision for Clarification about use of DH14 in NDcPPv2.2e	Yes	
TD0581	NiT Technical Decision for Elliptic Curve-based key establishment and NIST SP 800-56A rev3	Yes	
TD0591	NIT Technical Decision for Virtual TOEs and hypervisors	No	TOE is not a virtual TOE
TD0592	NIT Technical Decision for Local Storage of Audit Records	Yes	
TD0631	NIT Technical Decision for Clarification of public key authentication for SSH Server	Yes	
TD0632	NIT Technical Decision for Consistency with Time Data for vNDs	No	Not a virtual Network Device
TD0635	NIT Technical Decision for TLS Server and Key Agreement Parameters	Yes	
TD0636	NIT Technical Decision for Clarification of Public Key User Authentication for SSH	Yes	

	TD0638	NIT Technical Decision for Key Pair Generation for Authentication	Yes		
	TD0639	NIT Technical Decision for Clarification for NTP MAC Keys	No	FCS_NTP_EXT.1 not claimed	
	TD0670	NIT Technical Decision for Mutual and Non-Mutual Auth TLSC Testing	Yes		
	TD0738	NIT Technical Decision for Link to Allowed-With List	yes		
	TD0790	NIT Technical Decision for Clarification Required for testing IPv6	Yes		
	TD0792	NIT Technical Decision for FIA_PMG_EXT.1 - TSS EA not in line with SFR	yes		
	TD0800	TD0800: NIT Technical Decision for IPsec IKE/SA Lifetimes Tolerance	No	The TOE does not claim IPsec.	
Tools	Refer to Test Plan				

1.3 Reference Documents

Table 3: List of Reference Documents

Ref	Document
[ST]	NETSCOUT nGeniusONE with InfiniStreamNG v6.3.3 Security Target, v1.25, April 2024
[CC_GUIDE]	NETSCOUT nGeniusOne with InfiniStreamNG v6.3.3 Common Criteria Guide, v1.20, April 2024
[ADMIN]	NETSCOUT Server Administrator Guide Release Version 6.3.3, 733-1661, Rev. I/ July 2022
[PP]	collaborative Protection Profile for Network Devices, Version 2.2e, 23-March-2020
[SD]	Supporting Document Mandatory Technical Document Evaluation Activities for Network Device cPP, December-2019, Version 2.2

2 Evaluation Activities for SFRs

2.1 Security Audit (FAU)

2.1.1 FAU_GEN.1 Audit data generation

2.1.1.1 TSS

3 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.

Findings:	[ST] Section 6.1.1 - The following information is logged as a result of the Security Administrator generating/importing or deleting cryptographic keys: a) Cryptographic key name b) Storage location c) Function performed.

4 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.

Findings:	[ST] Section 6.1.1 – Table 15 lists auditable events and the TOE component that generates said event.

2.1.1.2 Guidance Documentation

5 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).

Findings:	The [CC_GUIDE] Section 3.11 provides examples of each auditable event required by FAU_GEN.1.

6 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.

Findings:	The evaluator performed this activity as part of those AAs associated with ensuring the corresponding guidance documentation satisfied their independent requirements. However, overall, the evaluator considered the administrator guides published by the vendor. The evaluator reviewed the contents of the documentation and looked specifically for functionality related to the scope of the evaluation. Where there was missing or incomplete descriptions for the functionality such that the user could not complete the testing AAs, the evaluator requested the vendor to supply augmented guidance information.

2.1.1.3 Tests

7 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.

8 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.

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

Findings:	These tests are conducted throughout the test plan and includes testing for distributed TOEs.

2.1.2 FAU_GEN.2 User identity association

2.1.2.1 TSS & Guidance Documentation

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 Tests

10 This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

11 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.

Findings:	N/A – Auditable events in the distributed TOE are not instigated by another component. All TOE components independently audit their own events. This is described in section 3.6 of the [CC_GUIDE]: "*Once these steps for both the nGeniusONE and InfiniStream devices has been performed restart the syslog service and check that the ngp-audit-tunnel is running…*" and "*The audit records are securely sent to a remote audit server in the operational environment using SSH. Both TOE components transmit audit data to the remote audit server in real time.*"

2.1.3 FAU_STG_EXT.1 Protected audit event storage

2.1.3.1 TSS

12 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.

Findings:	[ST] Section 6.1.3 - The audit records are securely sent to a remote audit server in the operational environment using SSH. This prevents the audit records from unauthorized viewing and modification during transmission.

13 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.

Findings:	[ST] Section 6.1.3 - The TOE logs all events related to startup/shutdown, external communications, user authentication, and user management (user creation/deletion, password changes, role changes) and administrative commands in the audit log. The TOE is a distributed TOE with both components storing audit data locally. Local audit data is rotated daily. The local audit record will drop new audit data if it exceeds the storage capacity. Only authorized administrators may view audit records and no capability to modify the audit records is provided.

14 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.

Findings:	[ST] Section 6.1.3 - The TOE is a distributed TOE with both components storing audit data locally.

15 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.

Findings:	[ST] Section 6.1.3 - When the local audit data store is full, the TOE will drop audit data.

16 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 as well as acceptable frequency for the transfer of audit data.

Findings:	[ST] Section 6.1.3 - Both TOE components transmit audit data to the remote audit server in real time using SSH.

17 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).

Findings:	[ST] Section 6.1.3 - Both TOE components transmit audit data to the remote audit server in real time using SSH.

18 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.

Findings:	[ST] Section 6.1.3 - The TOE is a distributed TOE with both components storing audit data locally. The local audit record will drop new audit data when space is exceeded.

2.1.3.2 Guidance Documentation

19 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.

Findings:	[CC_GUIDE] Section 3.6 contains instructions on setting up the audit log server.

20 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.

Findings:	Section 3.6 of the [CC GUIDE] states that the audit data is stored locally and is as sent to the remote log server in real time.

21 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.

Findings:	The options for managing how the TOE reacts when the local storage is exhausted are not configurable.

2.1.3.3 Tests

22 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:

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

Findings:	Verification that the data is encrypted is satisfied by FTP_ITC.1 for the logging channel. The logging server is a CentOS 7 virtual machine running rsyslogd v8.24.0-34.el7 and OpenSSH_7.4p1, OpenSSL 1.0.2k-fips 26 Jan 2017. It requires authentication over an SSH tunnel. The evaluator observed and confirmed that the audit records were successfully received by the audit server when executing the test cases.

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

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

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

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

High-Level Test Description
The evaluator filled the storage location by copying a large file to /var/, and the /opt/NetScout/rtm/log directory The evaluator then observed the TOE disabled audit logging when the storage reached capacity.
Findings: PASS

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

Test Not Applicable	The ST does not claim this functionality.

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

Findings:	All TOE components were considered and tested in Test 1 and test 2 above.

2.2 Cryptographic Support (FCS)

2.2.1 FCS_CKM.1 Cryptographic Key Generation

2.2.1.1 TSS

23 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.

Findings:	[ST] Section 6.3.1 - The TOE supports key generation for the following asymmetric schemes:
a)	ECC P-256, P-384, P-521. Used in TLS and SSH. FFC Safe Prime(NIST Special Publication 800-56A Revision 3), used for SSH.

2.2.1.2 Guidance Documentation

24 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.

Findings:	[CC_GUIDE] Sections 3.1, 3.7, and 3.8 provide instructions on configuring the TOE to use the selected algorithms.

2.2.1.3 Tests

25 Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products. Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

26 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*.

27 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

28 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

29 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

30 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)

31 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.

32 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

33 and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

34 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.

35 The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

36 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.

37 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

38 for each FFC parameter set and key pair.

[Modified by TD0580].***FFC Schemes using "safe-prime" groups***

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

Findings:	ECC key generation is covered by the CAVP certificates listed in Table 4 of the [ST] which claims ECDSA KeyGen. They are consistent with FCS_CKM.1 and FCS_CKM.2 in the [ST] section 5.3.3. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms and certificate number A3740 (for NG1), A3739 (for InfiniStream + NG1) and A5165 (for NG1).

2.2.2 FCS_CKM.2 Cryptographic Key Establishment

2.2.2.1 TSS

40 **[Modified by 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.

41 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

42 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.

Findings:	[ST] Section 6.3.2 – Contains a table that maps the schemes to the SFRs that claim them. It also outlines the usage for each scheme.

2.2.2.2 Guidance Documentation

43 The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

Findings:	[CC_GUIDE] Sections 3.1, 3.7, and 3.8 provide instructions on configuring the TOE to use the selected schemes.

2.2.2.3 Tests

Key Establishment Schemes

44 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

45 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

46 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.

47 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.

48 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.

49 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.

50 If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

51 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.

52 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).

53 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.

Findings:	Key agreement schemes are covered by the CAVP certificates listed in Table 4 of the [ST]. They are consistent with FCS_CKM.1 and FCS_CKM.2 in the [ST] section 5.3.3. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms and certificate number A3740 (for NG1), A3739 (for InfiniStream + NG1) and A5165 (for NG1).

RSA-based key establishment schemes

54 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.

Findings:	RSA-based key establishment schemes are not claimed.

The Tests section for FCS_CKM.2 is **[Modified by TD0580]**.

FFC Schemes using "safe-prime" groups

55 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.

Findings:	Please see FCS_SSHC_EXT.1 and FCS_SSHS_EXT.1 for confirmation.

2.2.3 FCS_CKM.4 Cryptographic Key Destruction

2.2.3.1 TSS

56 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[1]). 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.

Findings:	[ST] Section 6.3.3 – Table 17 contains a list of the keys, the algorithm used for those keys, the storage mechanism for the keys and how each key is zeroized after it is no longer needed.

57 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).

Findings:	[ST] Section 6.3.3 – SSH private keys stored in non-volatile memory are destroyed by using the Linux shred utility where the data is overwritten with pseudo-random numbers 15 times and a final overwrite of zeroes before removing the file. Long-term

[1] 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.

	TLS server private keys are stored in non-volatile memory and are destroyed using a Java-specific interface.

58 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.

Findings:	[ST] Section 6.3.3 of the ST describes how keys held in volatile memory, as well as How plain-text keys in volatile memory are being zeroized.

59 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.

Findings:	All relevant keys are described in [ST] section 6.3.3 . The ST does not claim any keys that are stored in non-plaintext format.

60 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.

Findings:	No such information is provided in the ST. There are no obvious circumstances that would prevent conformance to the requirements.

61 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.

Findings:	The TOE does not claim this selection.

2.2.3.2 Guidance Documentation

62 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.

63 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[2] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

Findings:	The guidance does not provide any evidence to suggest there are circumstances where keys are prevented or delayed from being cleared.

2.2.3.3 Tests

64 None

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

2.2.4.1 TSS

65 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.

Findings:	[ST] Section 6.3.4 - All TOE components perform encryption and decryption using the AES algorithm with key sizes of 128 and 256 bits in CBC and GCM modes for TLS communication support. The implementation performs encryption and decryption using the AES algorithm with key sizes of 128 and 256 bits in CBC mode for SSH communication support. Additionally, the AES algorithm with key size of 256 bits in CTR mode is implemented to support DRBG functionality (CTR_DRBG 256 bit).

2.2.4.2 Guidance Documentation

66 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.

Findings:	[CC_GUIDE] Section 3.1 provides instructions on initially configuring the TOE to use the selected algorithms.

2.2.4.3 Tests

AES-CBC Known Answer Tests

67 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.

[2] 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).

68 **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.

69 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.

70 **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.

71 To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

72 **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].

73 To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key 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.

74 **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].

75 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

76 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.

77 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

78 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]
```

79 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.

80 The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Test

81 The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

a) **Two plaintext lengths**. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

a) **Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

b) **Two IV lengths**. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

82 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.

83 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.

84 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

85 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, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

86 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.

87 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.

88 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.

89 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].

90 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

91 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

92 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]
```

93 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.

94 There is no need to test the decryption engine.

Findings:	AES encryption and decryption is covered by the following CAVP certificates for 128 and 256-bit AES in CBC, GCM and CTR: A3740, A3739, and A5165. These claims are consistent with FCS_COP.1/DataEncryption in the [ST] section 5.3.3. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

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

2.2.5.1 TSS

95 The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and key size supported by the TOE for signature services.

Findings:	[ST] Section 6.3.5 - The TOE provides cryptographic signature generation and verification services using: Elliptic Curve Digital Signature Algorithm (ECDSA) with 256, 384, and 521-bit key sizes using NIST curves of P-256, P-384 and P-521 for TLS communications in accordance with ISO/IEC 14888-3

2.2.5.2 Guidance Documentation

96 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.

Findings:	[CC_GUIDE] Section 3.1 provides instructions on configuring the TOE to use the selected algorithms.

2.2.5.3 Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

97 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

98 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

99 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.

100 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

101 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.

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

Findings:	ECDSA signature generation and verification is covered by the following CAVP certificate which claim ECDSA Signature Algorithm with NIST curves P-256, P-384 and P-521: A3740, A3739, and A5165. These claims are consistent with FCS_COP.1/SigGen in the [ST] section 5.3.3. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

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

2.2.6.1 TSS

103 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.

Findings:	[ST] Section 6.3.6 - The TOE provides cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512. SHA is implemented in the following parts of the TSF:
a)	SHA-1 for Diffie-Hellman group 14 SSH key agreement scheme
b)	SHA-256 for SSH HMAC message authentication support
c)	SHA-1 and SHA-256 for TLS algorithm support (AES 128 and AES 256)
d)	SHA-256, SHA-384, SHA-512 for ECDSA algorithm support (P-256, P384, P521)
e)	SHA-256 for password hashing

2.2.6.2 Guidance Documentation

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

Findings:	[CC_GUIDE] Section 3.1 provides instructions on configuring the TOE to use the selected algorithms.

2.2.6.3 Tests

105 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.

106 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

107 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

108 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

109 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

110 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

111 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.

Findings:	Hashing is covered by the following CAVP certificates which claim SHA-1, SHA-256, SHA-384, SHA-512: A3740, A3739, and A5165. These claims are consistent with FCS_COP.1/Hash in the [ST] section 5.3.3. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP validated cryptographic algorithms.

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

2.2.7.1 TSS

112 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.

Findings:	[ST] Section 6.3.7 – Table 18 contains all the values used by the HMAC functions including; the algorithm; block size; key size; and digest size.

2.2.7.2 Guidance Documentation

113 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.

Findings:	[CC_GUIDE] Section 3.1 provides instructions on configuring the TOE to use the selected algorithms.

2.2.7.3 Tests

114 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.

Findings:	HMAC is covered by the following CAVP certificates which claim HMAC-SHA-256, and HMAC-SHA-512: A3740, A3739, and A5165. These claims are consistent with FCS_COP.1/KeyedHash in the [ST] section 5.3.3 and Table 18: HMAC Characteristics. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms..

2.2.8 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

115 Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [NDcPP].

2.2.8.1 TSS

116 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.

Findings:	[ST] Section 6.3.9 - All TOE components implement a NIST-approved counter deterministic random bit generator (CTR DRBG). All TOE components provide the same software-based entropy source as described in the proprietary entropy specification. The DRBG is seeded with a minimum of 256 bits of entropy so that it is sufficient to ensure full entropy for 256-bit keys, which are the largest keys generated by the TSF.

2.2.8.2 Guidance Documentation

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

Findings:	[CC_GUIDE] Section 3.1 provides instructions on configuring the TOE to use the selected schemes.

2.2.8.3 Tests

118 The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

119 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).

120 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.

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

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

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

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

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

Findings:	CTR_DRBG (AES) is covered by the following CAVP certificates which claims the use of a 256-bit key: A3740, A3739, and A5165. These claims are consistent with FCS_RBG_EXT.1 in the [ST] section 5.3.3. The evaluator verified that the crypto modules used in the TOE correspond to the crypto modules which have CAVP-validated cryptographic algorithms.

2.3 Identification and Authentication (FIA)

2.3.1 FIA_AFL.1 Authentication Failure Management

2.3.1.1 TSS

122 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.

Findings:	[ST] Section 6.4.1 – When a user account has sequentially failed authentication the configured number of times (default 3), the account will be locked until the Security Administrator manually unlocks the account or a configurable time period has elapsed.

123 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).

Findings:	[ST] Section 6.4.1 – When a user account is locked out, it is only locked out remotely.

2.3.1.2 Guidance Documentation

124 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.

Findings:	The [CC_GUIDE], Section 3.7 describes how to configure the number of unsuccessful authentication attempts as well as the lockout time-period. When an administrator is locked out, section 3.7 of the [CC_GUIDE] provides instructions on making use of the "unlocker" local-console account to manually unlock any locked accounts to maintain positive control of the TOE.

125 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.

Findings:	The [CC_GUIDE], Section 3.7 states that any interface can be locked out except for the serial console. This ensures that administrator access can always be maintained.

2.3.1.3 Tests

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

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

High-Level Test Description
Using the GUI, set the login threshold to 3 attempts. Using the GUI, log into the TOE twice using an incorrect password. On the third attempt, log in correctly and verify that the threshold has not been reached. Using the GUI, log into the TOE three times using an incorrect password. On the fourth attempt, log in correctly and verify that the threshold has been reached and that the user cannot log in. Using a secondary workstation with a distinct IP, log into the TOE using the GUI with the correct password. The attempt should fail. Wait for the lockup period to expire. Attempt to login using the locked out username and correct password. The attempt should succeed. Repeat the above test using SSH instead of the GUI. For administrator action and to ensure that the TOE does not lockout the Security Administration; the evaluator log into the Serial interface as the user ‘Unlocker’ and unlocked the locked out Administrator account.
Findings: PASS

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

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

Note:	In Test 1 above, the evaluator performed the action specified in the ST to re-enable the remote administrator's access and observed that it resulted in successful access.

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.

Note:	See Test 1 above.

2.3.2 FIA_PMG_EXT.1 Password Management

2.3.2.1 TSS

127 **[Modified by TD0792]** The evaluator shall check that the TSS lists the supported special character(s) for the composition of administrator passwords.

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

129 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..

Findings:	[ST] Section 6.4.2 - The passwords can be composed of any combination of upper and lower case letters, numbers, and special characters "!", "@", "#", "$", "%", "^", "&", "*", "(", ")". The minimum password length is settable by the Administrator and can range from 5 to 255 characters.

2.3.2.2 Guidance Documentation

130 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.

Findings:	[CC_GUIDE] Section 3.7 - passwords can be composed of any combination of upper and lower case letters, numbers, and special characters “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)”. Instructions for configuring the minimum password length is also provided.

2.3.2.3 Tests

131 The evaluator shall perform the following tests.

a) 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.

High-Level Test Description
The evaluator changed the password length to be 15 characters, then changed the password for a user using the identified TSFI and including a combination of case letters, numbers, and special characters. After which, the evaluator showed that the password can be used to login to the Web GUI. The evaluator also performed the test by changing the password length to be 8 characters, then changing the password of a user to be only 7 characters and showed that it is rejected. The password change was then attempted by changing it to be 8 characters, which was accepted.
Findings: PASS

b) 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.

Note:	See previous test case.

2.3.3 FIA_UIA_EXT.1 User Identification and Authentication

2.3.3.1 TSS

132 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”.

Findings:	[ST] Section 6.4.3 - The local CLI requires the user to authenticate to the TOE component’s local authentication mechanism with their username/password combination. The SSH CLI requires the user to authenticate with either a public key or username/password combination. The Web GUI requires the user to authenticate with a username/password. The TOE then either grants administrative access if the correct credentials or public keys are provided or indicates that the login was unsuccessful.

133 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.

Findings:	[ST] Section 6.4.3 - The only pre-authentication service allowed on the CLI is the display of the standard Linux pre-authentication banner; which can be configured by the Security Administrator through the CLI. The local CLI requires the user to authenticate to the TOE component's local authentication mechanism with their username/password combination. The only pre-authentication service allowed at the Web GUI is the pre-authentication banner; which can be configured by the Security Administrator through the CLI.

134 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.

Findings:	[ST] Section 6.4.3 - The TOE requires all users to be successfully identified and authenticated. Each TOE component maintains its own local authentication mechanism.

135 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.

Findings:	[ST] Section 6.4.3 - Each TOE component maintains its own local authentication mechanism. The only pre-authentication service allowed on the CLI is the display of the standard Linux pre-authentication banner; which can be configured by the Security Administrator through the CLI. The only pre-authentication service allowed at the Web GUI is the pre-authentication banner; which can be configured by the Security Administrator through the CLI.

2.3.3.2 Guidance Documentation

136 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 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.

Findings:	Section 3.7 of the [CC_GUIDE] describes preparatory steps needed before successful logon can occur. For example, the Web UI needs to have a valid TLS web server certificate, and the CLI requires additional configuration needed to enforce security properties as per the other SFRs. Section 3.3 of the [CC_GUIDE] provides instructions on connecting to each of the in-scope management interfaces.

2.3.3.3 Tests

137 The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

High-Level Test Description
The evaluator logged into the identified management interface using a known-good credential and logged out. The evaluator then attempted to log into the identified management interface using a known-bad credential and verified that the TOE does not allow the user to be logged in. and that the appropriate audit messages appeared.
Findings: PASS

b) Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

High-Level Test Description
The evaluator accessed the TOE GUI without logging in and determined the set of services available. The evaluator also accessed the CLI via SSH and determined the set of services available.
Findings: PASS

c) Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.

High-Level Test Description
The device does not have any services configured prior to I&A. The evaluator accessed the TOE serial console without logging in and determined the set of services available.
Findings: PASS

d) Test 4: For distributed TOEs where not all TOE components support the authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the evaluator shall test that the components authenticate Security Administrators as described in the TSS.

Note:	See previous tests. All components were tested. The components authenticates the security administrator as specified in the TSS.

2.3.4 FIA_UAU_EXT.2 Password-based Authentication Mechanism

138 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 FIA_UAU.7 Protected Authentication Feedback

2.3.5.1 TSS

139 None.

2.3.5.2 Guidance Documentation

140 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.

Findings:	The evaluator examined guidance documentation to determine that no necessary preparatory steps are necessary to ensure authentication data is not revealed while entering for each local login.

2.3.5.3 Tests

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

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

High-Level Test Description
The evaluator logged into the local management interface, and ensured that the password field does not echo characters – even a masking character -- as claimed by the ST.
Findings: PASS

2.4 Security management (FMT)

2.4.1 General requirements for distributed TOEs

2.4.1.1 TSS

142 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.

Findings:	[ST] Section 6.5.6 – Table 19 outlines the management capabilities of the TOE and whether each TOE component supports the functionality.

2.4.1.2 Guidance Documentation

143 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.

Findings:	The Guidance documentation describes management of each TOE component covering all relevant aspects covered by FMT SFRs. Descriptions of the functions claimed by FMT_SMF.1 can be found in the following sections of the [CC_GUIDE]. Management claims apply to all components independently unless explicitly stated otherwise. - Ability to administer the TOE locally and remotely; See Section 3.3 of [CC_GUIDE] - Ability to configure the access banner; See Section 3.3 of [CC_GUIDE] - Ability to configure the session inactivity time before session termination or locking; See Section 3.3 of [CC_GUIDE] - Ability to update the TOE, and to verify the updates using [hash comparison] capability prior to installing those updates; See Section 2.4 of [CC_GUIDE] - Ability to configure the authentication failure parameters for FIA_AFL.1; See Section 3.7 of [CC_GUIDE] - Ability to start and stop services: See section 3.9 of [CC_GUIDE]. - Ability to manage the cryptographic keys; See Section 3.7, and Section 3.8 of [CC_GUIDE] - Ability to configure the cryptographic functionality: this selection in FMT_SMF.1 is selected because the distributed TOE components have a registration channel (as per Application Note 24 in the [PP]. The configuration of the registration channel is described in section 3.8 of [CC_CGUIDE]. - Ability to configure the interaction between TOE components: See section 3.8 of [CC_GUIDE]. - Ability to re-enable an Administrator account: See section 3.7 of [CC_GUIDE]. - Ability to set the time which is used for time-stamps: See section 3.5 of [CC_GUIDE]. - Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors: Trust anchors are maintained by the NG1 component and are described in sections 3.7 and 3.8 of [CC_GUIDE]. - Ability to import X.509v3 certificates to the TOE's trust store: Trust anchors are maintained by the NG1 component and are described in sections 3.7 and 3.8 of [CC_GUIDE]. - Ability to manage the trusted public keys database: Section 3.7 of the [CC_GUIDE] describes the use of the authorized_keys file with respect to managing user authentication keys for SSH.

2.4.1.3 Tests

144 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).

Findings:	The evaluator verified and tested all supported functions related to FMT SFRs on each TOE components as outlined in Table 19 of the ST during initial installation and configuration, as well as, during the testing efforts.

2.4.2 FMT_MOF.1/ManualUpdate

2.4.2.1 TSS

145 For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

Findings:	See section 2.4.1.1

2.4.2.2 Guidance Documentation

146 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).

Findings:	[CC_GUIDE] Section 2.4 "Updating the TOE" provides instructions on updating the NG1 component and the InfiniStream component of the TOE.

147 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).

Findings:	[CC_GUIDE] Section 2.4 'Updating the TOE' describes how all steps for updating all TOE components. It states that 'The nGeniusOne and InfiniStream TOE components may be updated in no particular order. All services should be stopped before upgrades are performed, allowing for no dependency on which component is updated first'.

2.4.2.3 Tests

148 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.

149 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.

High-Level Test Description
The evaluator logged into the Web GUI using an account with privileges which should not permit upgrades, and attempted to upgrade the device. The action failed.
Findings: PASS

2.4.3 FMT_MTD.1/CoreData Management of TSF Data

2.4.3.1 TSS

150 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.

Findings:	[ST] Section 6.5.4 - Users are required to login before being provided with access to any administrative functions. In addition [ST] Section 6.4.3 - The only pre-authentication service allowed on the CLI is the display of the standard Linux pre-authentication banner. The only pre-authentication service allowed at the Web GUI is the pre-authentication banner; which can be configured by the Security Administrator through the CLI.

151 If 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.

Findings:	[ST] Section 6.5.4 - Access to TSF data and functions, including managing the TOE's trust store, is restricted to Security Administrators.

2.4.3.2 Guidance Documentation

152 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.

Findings:	Please see section 2.4.1.2 above for a description of the evaluated TSF data-manipulating functions described within the [CC_GUIDE]. In addition, configuration information is provided in section 3.1 of [CC_GUIDE] to ensure that only authorized administrators have access to certain functions via the "sudo" command. The administrative Web UI needs no further configuration to restrict commands.

153 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.

Findings:	Section 3.7 of the [CC_GUIDE] outlines how to use the nscertustil.sh script to manage certificates on the NG1 server. Section 3.8 of the [CC_GUIDE] describes sufficient

	information for administering and maintaining the trust store, as well as how to designate a CA trust anchor for the purposes of adding InfiniStream components

.

2.4.3.3 Tests

154 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.

2.4.4 FMT_SMF.1 Specification of Management Functions

155 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.

2.4.4.1 TSS (containing also requirements on Guidance Documentation and Tests)

156 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).

Findings:	[ST] Section 6.5.6 - The TOE may be managed via the CLI (console & SSH) or GUI (HTTPS). This section also lists the management capabilities of the TOE. Table 19 of the [ST] identifies the interfaces on which each management function can be performed. In addition, please see section 2.4.1.2 above for a description of the evaluated TSF data-manipulating functions described within the [CC_GUIDE]. Table 19 in the TSS is consistent with the description in the [CC_GUIDE].

157 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.

Findings:	[ST] Section 6.5.6 and [CC_GUIDE] section 3.3 both describe the local administrative interface.

158 For distributed TOEs with the option 'ability to configure the interaction between TOE components' the evaluator shall examine that the ways to configure the interaction between TOE components is detailed in the TSS and Guidance Documentation. The evaluator shall check that the TOE behaviour observed during testing of the configured SFRs is as described in the TSS and Guidance Documentation.

Findings:	[ST] Section 6.5.6 references FCO_CPC_EXT.1 (in section 6.2.1 of the TSS) which explains that the administrator can configure the interaction between TOE components.

	In the [CC_GUIDE], section 3.8 describes how the NG1 can add and remove InfiniStream components. Testing of the components yielded behaviour consistent with the descriptions contained in the [ST] and the [CC_GUIDE].

2.4.4.2 Guidance Documentation

159 See section 2.4.4.1.

2.4.4.3 Tests

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

2.4.5 FMT_SMR.2 Restrictions on security roles

2.4.5.1 TSS

161 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.

Findings:	[ST] Section 6.5.7 - The TOE operates three pre-defined users: a) backups. Privileged access to the TOE via CLI. This profile equates to the Security Administrator role defined in this Security Target. b) Administrator. Privileged access to the TOE via Web GUI. This profile equates to the Security Administrator role defined in this Security Target. c) unlocker. Restricted access to the TOE via CLI. Limited to unlocking the backups user when the account has been locked due to authentication failures. Management of TSF data via the CLI or web GUI is restricted to Security Administrators. To be considered the Security Administrator in the web GUI, the user must belong to the Security Administrator role.

2.4.5.2 Guidance Documentation

162 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.

Findings:	Section 3.3 of the [CC_GUIDE] contains instructions for administering the TOE both locally and remotely. No instructions are needed to configure a modern web browser to accommodate the web UI cryptographic properties. An SSH client will need to be configured to be capable of negotiating the server configuration options defined in section 3.1 of [CC_GUIDE].

2.4.5.3 Tests

163 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.

Note	There are no explicit test activities and therefore none are recorded here. All interfaces are tested throughout this test plan.

2.5 Protection of the TSF (FPT)

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

2.5.1.1 TSS

164 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.

Findings:	[ST] Section 6.6.3 - Symmetric or asymmetric keys cannot be viewed from TOE components. Security Administrators of the Web GUI or CLI are unable to view the keys stored in files or in memory.

2.5.2 FPT_APW_EXT.1 Protection of Administrator Passwords

2.5.2.1 TSS

165 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.

Findings:	[ST] Section 6.6.2 - Administrator passwords are not stored by any TOE component in plaintext. All administrative passwords are hashed using SHA-256 and the hash is what is stored by each TOE component. There is no function provided by the TOE to display a password value in plaintext. Each TOE component has its own local password store.

2.5.3 FPT_TST_EXT.1 TSF testing

2.5.3.1 TSS

166 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.

Findings:	[ST] Section 6.6.4 – Outlines the various tests that are ran. Also states “These tests and their response to failures is sufficient to ensure that the TSF behaves as described in the ST because it would detect any unauthorized modifications to the TOE, failures or tampering of the hardware (which could be an attempt to compromise its storage or take the TOE out of the range of operating conditions specified for its entropy source), and any cryptographic failures that could result in the establishment of insecure trusted channels.”

167 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.

Findings:	[ST] Section 6.6.4 - All self-tests are performed by both TOE components at start-up.

2.5.3.2 Guidance Documentation

168 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.

Findings:	[CC_GUIDE] Section 2.3 ‘Power-On Self-Tests” describes the possible errors that each TOE component can emit as well as any remediation steps to try. The tests are consistent with those described in the [ST] Section 6.6.4.

169 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.

Findings:	Section 2.3 ‘Power-on Self-Tests” of [CC_GUIDE] describes how to determine from the error messages which TOE component has failed the self-test. It’s states that any failure of the POST test writes a diagnostic code to the console (an LCD K/V/M or a terminal connected to the serial port) and sounds a beep code on the system speaker.

2.5.3.3 Tests

170 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.

171 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.

172 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.

173 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.

High-Level Test Description
For InfiniStream: The evaluator rebooted the device. After the device rebooted the evaluator reviewed the self-test log messages and verified that all necessary self-tests were executed. For nGeniusONE: The evaluator rebooted the device. After the device rebooted the evaluator reviewed the self-test log messages and verified that all necessary self-tests were executed.
Findings: PASS

2.5.4 FPT_TUD_EXT.1 Trusted Update

2.5.4.1 TSS

174 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.

Findings:	[ST] Section 6.6.5 - The current TOE version for each component can be queried through the use of the Web GUI. Delayed activation is not supported by the TOE.

175 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.

Findings:	[ST] Section 6.6.5 - The Security Administrator must download the update along with the SHA256 hash associated with the file from the NETSCOUT support website to their local machine. The ST does not claim support for digital signature verification of software updates.

176 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.

Findings:	The TOE does not support these selections.

177 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.

Findings:	[ST] Section 6.6.5 - Each TOE component is manually updated by the Security Administrator. The Security Administrator must download the updates along with the SHA256 hash associated with the file from the NETSCOUT support website to their local machine. The TOE component itself never communicates with the vendor support website. The NG1 is updated using the CLI and the InfiniStream is updated via the NG1. The TOE is designed to function properly as individual components are updated.

178 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.

Findings:	[ST] Section 6.6.5 - The Security Administrator must download the update along with the SHA256 hash associated with the file from the NETSCOUT support website to their local machine. Additionally, the security administrator must verify the hash with hash of the update file, and manually initiate the update.

2.5.4.2 Guidance Documentation

179 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.

Findings:	[ADMIN] section “*Verify the Software Version”* explains how to query the TOE version. There are two means to query the TOE version: *Navigate to /opt/NetScout/rtm/bin, execute the ./localconsole command, and verify the version information shown at the top of the display.* *From the operating system command line, enter the following:* *cat /opt/NetScout/rtm/pa/bin/decoderelease.properties* *The screen outputs text similar to the following:* *decodeengine.version = Version 6.3.0 Buildxxx*

Make sure the build number matches the expected version.

The TOE does not claim delayed activation.

180 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.

Findings: Section 2.4 of the [CC_GUIDE] states the following:

To verify upgrade images perform the following procedures:

a) Access the system command-line as the backups user.

b) Navigate to the directory to which you copied the downloaded files.

c) Ensure the checksum files and binary are in this same directory.

d) Use the following command to generate a new checksum for the binary, and compare it to the downloaded checksum. /usr/bin/sha256sum -c <sha checksum filename>

Example of valid file output:

a) [backups@host /opt/install]# /usr/bin/sha256sum -c pm-6330-XXX-lin.sha256

pm-6330-XXX-lin.bin: OK

Example of invalid file output :

a) [backups@host /opt/install]# /usr/bin/sha256sum -c pm-6330-XXX-lin.sha256

pm-6330-XXX-lin.bin: FAILED

sha256sum: WARNING: 1 of 1 computed checksum did NOT match

181 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.

Findings: [CC_GUIDE] states in Section 2.4 that the published hash can be obtained from the vendor.

182 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.

Findings:	Section 2.2 of the [CC_GUIDE] describes how to verify software version, and Section 2.4 identifies how to check for and apply updates.

183 If this information was not provided in the TSS: For distributed TOEs, the evaluator shall examine the Guidance Documentation to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component.

Findings:	Section 6.6.5 of the [ST] states "*The NG1 is updated using the CLI and the InfiniStream is updated via the NG1. The TOE is designed to function properly as individual components are updated.*" Similarly, the guidance documentation in [CC_GUIDE] section 2.4 describes how all TOE components are updated using a vendor-supplied hash that is checked manually by an Administrator.

184 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.

Note:	The TOE does not support certificate-based mechanism to verify software updates.

2.5.4.3 Tests

185 The evaluator shall perform the following tests:

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

High-Level Test Description
Before performing the installation the evaluator determined current version of the TOE installed. The evaluator then executed the installation of legitimate version of the TOE. After the install, the evaluator confirmed that the TOE version is consistent with the newly installed version.
Findings: PASS

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

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

2) An image that has not been signed

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

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

Test Not Applicable:	The TOE does not verify digital signatures to authorize the installation of images.

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

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

verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

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

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

Test not Applicable	The TOE itself does not verify a hash value over an image against a published hash value.

186 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.

187 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).

Note	The TOE only supports manual updates.

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

Note:	All distributed components were tested for claims consistent with Test 1, 2, and 3.

2.5.5 FPT_STM_EXT.1 Reliable Time Stamps

2.5.5.1 TSS

189 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.

Findings:	[ST] Section 6.6.6 - The TOE incorporates an internal clock that is used to maintain date and time. The Security Administrator sets the date and time during initial TOE configuration and may change the time during operation. The TOE makes use of time for the following: a) Audit record timestamps b) Session timeouts (lockout enforcement) c) Certificate validation

190 **[Modified by TD0632]** 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.

Findings:	N/A as "obtain time from the underlying virtualization system" is not selected in the [ST].

2.5.5.2 Guidance Documentation

191 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.

Findings:	[CC_GUIDE] section 3.5 describes how the time is updated on all TOE components.

192 **[Modified by TD0632]** 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.

Findings:	N/A as "obtain time from the underlying virtualization system" is not selected in the [ST].

2.5.5.3 Tests

193 The evaluator shall perform the following tests:

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

High-Level Test Description
Using the local host computer hosting the GUI, the evaluator changed the date/time in the past by 1 day, 1 hour and 42 minutes, and verified the time was set properly. Using the local host computer hosting the GUI, the evaluator changed the date/time to the future by 7 days, 1 hour and 42 minutes, and verified the time was set properly.
Findings: PASS

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

Test Not Applicable	The TOE does not claim NTP.

c) **[Modified by 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.

Test Not Applicable	The TOE does not claim obtaining time from the underlying VS.

194 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.

Note:	All TOE components independently audit their actions with messages that contain unambiguous time information as to when the message was generated.

2.6 TOE Access (FTA)

2.6.1 FTA_SSL_EXT.1 TSF-initiated Session Locking

2.6.1.1 TSS

195 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.

Findings:	[ST] Section 6.7.1 - The Security Administrator may configure the TOE to terminate an inactive local interactive session (CLI) following a specified period of time.

2.6.1.2 Guidance Documentation

196 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.

Findings:	Section 3.3 of the [CC_GUIDE] indicates that only session termination is supported. The instructions to configure the session termination mechanism – called the "inactivity period" [Console and SSH CLI] or "session timeout" [Web GUI HTTPS] – are provided in section 3.3 of [CC_GUIDE].

2.6.1.3 Tests

197 The evaluator shall perform the following test:

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

High-Level Test Description
For each of 2 or 3 and 5 minutes: The evaluator changed the idle timeout value; Logged into the device; Wait 30 seconds before the timeout expires, verify the session is still alive by sending a keep alive as described above in the TSFI commands. This should reset the timeout clock. The purpose is to ensure the timeout is not premature. Wait another minute. Verify the session is still alive by sending a keep alive. This should reset the timeout clock. The purpose is to ensure the timeout has been reset by the initial keep alive action above. Wait for the full duration of the timeout without sending any keep alives. The session should terminate.
Findings: PASS

2.6.2 FTA_SSL.3 TSF-initiated Termination

2.6.2.1 TSS

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

Findings:	[ST] Section 6.7.2 - The Security Administrator may configure the TOE to terminate an inactive remote interactive session (CLI and Web GUI) following a specified period of time. The Security Administrator may configure the TOE to terminate an active session after a specified period of time. The session timeout and inactive timeout

	values are indicated in minutes. Each TOE component enforces its own termination of an idle session and must be individually configured.

2.6.2.2 Guidance Documentation

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

Findings:	Section 3.3 of [CC_GUIDE] describes how to configure inactivity timeout and session termination for the remote administrative sessions.

2.6.2.3 Tests

200 For each method of remote administration, the evaluator shall perform the following test:

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

High-Level Test Description
For each of 3 and 5 minutes: The evaluator changed the idle timeout value. Log into the device; With 30 seconds past, verify the session is still alive by sending a keep alive as described above in the TSFI commands. This should reset the timeout clock. The purpose is to ensure the timeout is not premature. Wait for the full duration of the timeout without sending any keep alives. The session should terminate.
Findings: PASS

2.6.3 FTA_SSL.4 User-initiated Termination

2.6.3.1 TSS

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

Findings:	[ST] Section 6.7.3 - Administrative users may terminate their own sessions at any time using the 'exit' command. Administrative users may terminate their Web GUI session at any time using the Logoff option in the Menu.

2.6.3.2 Guidance Documentation

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

Findings:	[CC_GUIDE] Section 3.3 outlines how to terminate local and remote sessions.

2.6.3.3 Tests

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

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

High-Level Test Description
The evaluator logged into the serial console, then logged out using the TSFI previous discussed. The session termination was observed.
Findings: PASS

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

High-Level Test Description
The evaluator logged into the Web GUI interface, then logged out. The evaluator logged into InfiniStreamNG CLI, then logged out. The evaluator logged into NG1 CLI, then logged out. The session termination was observed each time.
Findings: PASS

2.6.4 FTA_TAB.1 Default TOE Access Banners

2.6.4.1 TSS

204 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).

Findings:	[ST] Section 6.7.4 - The TOE displays an administrator configurable message to users prior to login at the CLI and web GUI.

2.6.4.2 Guidance Documentation

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

Findings:	Section 3.3 of [CC_GUIDE] describes how to configure the banner messages. Note that the banner for the serial console and SSH console are configured simultaneously as per section 3.3 of [CC_GUIDE].

2.6.4.3 Tests

206 The evaluator shall also perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

High-Level Test Description
GUI (NG1) The evaluator accessed the NG1 Web interface to check messages. The banner messages was then changed via the banner properties file. A new remote session was then started, which showed that the banner was modified and is presented prior to I&A. Local and SSH, NG1: The evaluator logged into the CLI and modified the banner properties file. The configured banner was observed at both the Serial and SSH CLI interfaces. Local and SSH, InfiniStreamNG The evaluator logged into the CLI and modified the banner properties file. The configured banner was observed at both the Serial and SSH CLI interfaces.
Findings: PASS

2.7 Trusted path/channels (FTP)

2.7.1 FTP_ITC.1 Inter-TSF trusted channel

2.7.1.1 TSS

207 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.

Findings:	[ST] Section 6.8.1 - The TOE provides the ability to secure sensitive data in transit to and from assured endpoints in the TOE's Operational Environment. All TOE components are acting as a SSH client and are conformant to FCS_SSHC_EXT.1.

2.7.1.2 Guidance Documentation

208 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.

Findings:	Section 3.6 of [CC_GUIDE] provides instructions to set up the trusted channel to the audit server. Instructions for troubleshooting the audit tunnel are also provided.

2.7.1.3 Tests

209 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.

210 The evaluator shall perform the following tests:

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

Findings:	The TOE maintains trusted channels to the remote audit log, which is set up as per the evaluated configuration. It is constantly tested throughout the evaluation.

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

High-Level Test Description
The evaluator followed the guidance found in Section 3.6 ‘Audit Logging’ of [CC_GUIDE] and verified that the communication channel can be initiated from the TOE.
Findings: PASS

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

Findings:	Refer to previous test.

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

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE’s application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

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

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

High-Level Test Description
Engage packet capture over the logging interface. Log into the CLI using a predefined user and good password. Interrupt the logging channel until the MAC layer is disrupted -- eg. when the link light goes out and then plug the cable back in until the link light comes back on. Exit the SSH session Examine packet capture and ensure that communications are appropriately protected and no TSF data is sent in plaintext. Then for the long duration interrupt test. repeat the test but with a long network interruption where the disconnect ensures that the SSH channel fully disconnects and cannot be automatically fixed. This takes about 17 minutes.
Findings: PASS

Further assurance activities are associated with the specific protocols.

211 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.

Findings:	The tests were executed on all TOE components.

212 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 FTP_TRP.1/Admin Trusted Path

2.7.2.1 TSS

213 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.

Findings:	[ST] Section 6.8.2 – The TOE provides the following trusted paths for remote administration:

a) CLI over SSH per FCS_SSHS_EXT.1

b) Web GUI over HTTPS per FCS_HTTPS_EXT.1.1 (HTTPS uses TLS per FCS_TLSS_EXT.1)

2.7.2.2 Guidance Documentation

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

Findings:	Section 3.3 of [CC_GUIDE] describes each remote administration interface.

2.7.2.3 Tests

215 The evaluator shall perform the following tests:

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

Findings:	The only trusted paths are the web interface and CLI, which are both set up as per the evaluated configuration. They are constantly tested throughout the evaluation.

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

High-Level Test Description
Engage wireshark over the appropriate interface. Log into the trusted path, and execute a command or edit a file. Examine wireshark and verify that the trusted path sends encrypted traffic after any initial plaintext protocol negotiation occurs. Repeat on each TOE component.
Findings: PASS

216 Further assurance activities are associated with the specific protocols.

217 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.

Findings:	The evaluator tested all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

3 Evaluation Activities for Optional Requirements

3.1 Identification and Authentication (FIA)

3.1.1 FIA_X509_EXT.1/ITT X.509 Certificate Validation

3.1.1.1 TSS

218 The evaluator shall examine the TSS to ensure it 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). If selected, the TSS shall describe how certificate revocation checking is performed. It is not sufficient to verify the status of a X.509 certificate only when it's loaded onto the device.

Findings:	[ST] Section 6.4.5 - As part of the certificate validation checking, the TSF will validate certificate revocation status using an OCSP server in the Operational Environment

3.1.1.2 Guidance Documentation

219 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 describe how certificate revocation checking is performed.

Findings:	[CC_GUIDE] Section 3.8 - The TOE performs certificate validity checking for the TLS connection between TOE components. As part of the certificate validation checking, the TSF will validate certificate revocation status using an OCSP server in the Operational Environment. If the revocation status cannot be verified, the certificate will be rejected. The TOE ensures the extendedKeyUsage field includes the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) for server certificates used in TLS, or the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) for OCSP certificates used for OCSP.

3.1.1.3 Tests

220 The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device. The evaluator shall perform the following tests for FIA_X509_EXT.1.1/ITT. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols.:

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

TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store).

High-Level Test Description
The evaluator presented the TOE with a valid chain of certificates (terminating in a trusted CA certificate) and demonstrated that the function succeeds.
Findings: PASS

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.

High-Level Test Description
The evaluator presented the TOE with an invalid chain of certificates (by removing the intermediate CA certificate) and demonstrate that the function fails.
Findings: PASS

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

High-Level Test Description
The evaluator created/cloned an X.509 certificate with a ‘notBefore’ date in the past and a ‘notAfter’ date in the past but after the ‘notBefore’ date. The certificate was then imported for use on the InfiniStream device, after which, the evaluator logged into the NG1 management UI and attempted to update the note information on the InfiniStream server which established a TLS session and performs certificate verification. This resulted in the function failing.
Findings: PASS

c) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates--conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the TOE certificate and revocation of the TOE 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. No testing is required if no revocation method is selected. 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.

High-Level Test Description
The Evaluator ensured that the OCSP responder is running with valid index file. The evaluator then verified that a valid certificate results in a successful connection when attempting to add an InfiniStream component. The evaluator then revoked the InfiniStream server certificate and restarted the OCSP responder with the revoked index file.

The evaluator verified that the connection now fails due to the certificate being revoked.
Findings: PASS

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

High-Level Test Description
The Evaluator created an OCSP signing certificate using a known good CA certificate that has the OCSPSigning extendedKeyUsage flag enabled. The evaluator cloned the known good certificate and **removed** the OCSPSigning extendedKeyUsage. The OCSP signature only depends on the (cloned) private key of the CA used to sign it and the TOE does not engage in any certificate pinning. The evaluator then replaced the old OCSP certificate with the new one. From the NG1 Web GUI the evaluator attempted to connect to the InfiniStream component and verify that the connection fails due to the OCSP responder certificate not having the OCSPSigning bit.
Findings: PASS

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

High-Level Test Description
The evaluator used a tool to act as a TLS server with appropriate parameters, acting as an InfiniStream device: With the test server listening on the correct port, the evaluator logged into the NG1 Web GUI and attempted to add the test server as an InfiniStream device. The TLS handshake failed, with reason for failure error message.
Findings: PASS

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

High-Level Test Description
The evaluator used a tool to act as a TLS server with appropriate parameters, acting as an InfiniStream device: With the test server listening on the correct port, the evaluator logged into the NG1 Web GUI and attempted to add the test server as an InfiniStream device. The TLS handshake failed, with reason for failure error message.
Findings: PASS

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

High-Level Test Description
The evaluator used a tool to act as a TLS server with appropriate parameters, acting as an InfiniStream device: With the test server listening on the correct port, the evaluator logged into the NG1 Web GUI and attempted to add the test server as an InfiniStream device. The TLS handshake failed, with reason for failure error message.
Findings: PASS

221 **[Modified by TD0527]** The following tests are run when a minimum certificate path length of three certificates is implemented.

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

223 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.

224 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.

225 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.

Test Not Applicable:	The TOE claims a chain of two certs for the intra-TOE communications and this test requires a minimum claimed chain of three. Therefore, this test case is N/A.

3.1.2 FIA_X509_EXT.1.2/ITT

226 The evaluator shall perform the following tests for FIA_X509_EXT.1.2/ITT. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/ITT. 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.

227 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).

228 For each of the following tests the evaluator shall create a chain of at least two certificates: a self-signed root CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

a) Test 1: The evaluator shall ensure that one CA 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).

High-Level Test Description
The evaluator loaded a known-good CA into the TOE trust store, and verified that the certificate is imported successfully. The known good CA certificate was then cloned with the basicConstraints extension removed. The evaluator verified that the cloned certificate fails to be imported.
Findings: PASS

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

High-Level Test Description
The evaluator cloned the known good CA certificate setting the basicConstraints extension to have the CA flag set to FALSE. The evaluator then verified that the cloned certificate will fail to be imported by the TOE.
Findings: PASS

3.2 Protection of the TSF (FPT)

3.2.1 FPT_ITT.1 Basic internal TSF data transfer protection

229 If the TOE is not a distributed TOE, then no evaluator action is necessary. For a distributed TOE the evaluator carries out the activities below.

3.2.1.1 TSS

230 The evaluator shall examine the TSS to determine that, for all communications between components of a distributed TOE, each communications mechanism is identified in terms of the allowed protocols for that IT entity. The evaluator shall also confirm that all protocols listed in the TSS for these inter-component communications are specified and included in the requirements in the ST.

Findings:	[ST] Section 6.6.1 - The TOE protects communication between the TOE components using HTTPS/TLS protocol and cipher suites.

3.2.1.2 Guidance Documentation

231 The evaluator shall confirm that the guidance documentation contains instructions for establishing the relevant allowed communication channels and protocols between each pair of authorized TOE components, and that it contains recovery instructions should a connection be unintentionally broken.

Findings:	Section 3.8 of the [CC_GUIDE] describes how to establish trusted communication channels and protocols between each TOE component. If the connection between the NG1 and InfiniStream components are unintentionally broken, section 3.8 of [CC_GUIDE] claims no plaintext is sent. The reconnect re-initiates the TCP handshake and TLS handshake. TLS will be reused when the connection is re-established.

3.2.1.3 Tests

232 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall ensure that communications using each protocol between each pair of authorized TOE components is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

High-Level Test Description
The evaluator engaged a packet capture over the appropriate interface, and ensured that access is configured for NG1 on the InfiniStream device. With the InfiniStream certificate root CA installed in the NG1 trust store, the evaluator logged into the Web GUI as the security administrator and attempted to add the InfiniStream device to NG1 after configuring authenticating parameters. The packet capture was examined and traffic encryption was verified.
Findings: PASS

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

High-Level Test Description
With the InfiniStream certificate root CA installed in the NG1 trust store,; the evaluator logged into the Web GUI as the security administrator and edited the note comment for the InfiniStream device and attempted to update the information. The packet capture was then inspected and verified that no data is sent in plaintext.
Findings: PASS

c) Test 3: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route between distributed components.

The evaluator shall ensure that, for each different pair of non-equivalent component types, the connection is physically interrupted for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration that is shorter than the application layer timeout but is of sufficient length to interrupt the network link layer.

The evaluator shall ensure that when physical connectivity is restored, either communications are appropriately protected, or the secure channel is terminated and the registration process (as described in the FTP_TRP.1/Join) re-initiated, with the TOE generating adequate warnings to alert the Security Administrator.

In the case that 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 components.

The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

High-Level Test Description
The evaluator engage a packet capture over an appropriate interface. Using the NG1 Web GUI, the evaluator edited the note comment for the InfiniStream device and attempted to update the information. The network cable connecting the TOE into the switch was then physically disconnect for a short duration. Then the network cable was reconnected to the switch. The packet capture was then inspected, and it was verified that no data was sent in plaintext. The test was repeated with a longer timeout duration. Examination of the packet capture verified that the log interface continues to send encrypted application Data packets without any user intervention.
Findings: PASS

233 Further assurance activities are associated with the specific protocols.

3.3 Communication (FCO)

3.3.1 FCO_CPC_EXT.1 Component Registration Channel Definition

3.3.1.1 TSS

234 (Note: paragraph 274 lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

235 The evaluator shall examine the TSS to confirm that it:

a) Describes the method by which a Security Administrator enables and disables communications between pairs of TOE components.

b) Describes the relevant details according to the type of channel in the main selection made in FCO_CPC_EXT.1.2:

- First type: the TSS identifies the relevant SFR iteration that specifies the channel used
- Second type: the TSS (with support from the operational guidance if selected in FTP_TRP.1.3/Join) describes details of the channel and the mechanisms that it uses (and describes how the process ensures that the key is unique to the pair of components) – see also the Evaluation Activities for FTP_TRP.1/Join.

236 The evaluator shall confirm that if any aspects of the registration channel are identified as not meeting FTP_ITC.1 or FPT_ITT.1, then the ST has also selected the FTP_TRP.1/Join option in the main selection in FCO_CPC_EXT.1.2.

Findings:	[ST] Section 6.2.1 - Registration of the TOE's components is performed manually by the Security Administrator and uses a channel that meets the secure channel requirements in FPT_ITT.1. To disable communication between the components, the Security Administrator can delete the InfiniStream appliance entry using the Web GUI.

3.3.1.2 Guidance Documentation

237 (Note: [n.b. PP paragraph 274] lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

238 The evaluator shall examine the guidance documentation to confirm that it contains instructions for enabling and disabling communications with any individual component of a distributed TOE. The evaluator shall confirm that the method of disabling is such that all other components can be prevented from communicating with the component that is being removed from the TOE (preventing the remaining components from either attempting to initiate communications to the disabled component, or from responding to communications from the disabled component).

Findings:	[CC_GUIDE] Section 3.8 contains instructions on enabling and disabling communications with individual components of a distributed TOE. A component that is disabled is removed from the NG1. In addition, InfiniStream components do not communication with one another.

239 The evaluator shall examine the guidance documentation to confirm that it includes recovery instructions should a connection be unintentionally broken during the registration process.

Findings:	[CC_GUIDE] section 3.8 - When the registration channel connection is broken, no plaintext is sent. The administrator must re-initiate the connection. TLS will be used when the connection is established.

240 If the TOE uses a registration channel for registering components to the TOE (i.e. where the ST author uses the FTP_ITC.1/FPT_ITT.1 or FTP_TRP.1/Join channel types in the main selection for FCO_CPC_EXT.1.2) then the evaluator shall examine the Preparative Procedures to confirm that they:

a) describe the security characteristics of the registration channel (e.g. the protocol, keys and authentication data on which it is based) and shall highlight any aspects which do not meet the requirements for a steady- state inter-component channel (as in FTP_ITC.1 or FPT_ITT.1)

b) identify any dependencies between the configuration of the registration channel and the security of the subsequent inter-component communications (e.g. where AES-256 inter-component communications depend on transmitting 256 bit keys between components and therefore rely on the registration channel being configured to use an equivalent key length)

c) identify any aspects of the channel can be modified by the operational environment in order to improve the channel security and shall describe how this modification can be achieved (e.g. generating a new key pair, or replacing a default public key certificate).

Findings:	Instructions for the NG1 to support HTTPS over TLS can be found in Sections 3.1 and 3.8 of the [CC_GUIDE], as well as, Section 6.6.4 *Configuring SSL/TLS* of [ADMIN]. The evaluator found in the [CC_GUIDE] and confirmed during testing that the security properties of the InfiniStream components and the NG1 component must be preconfigured by the Administrator to meet all of the necessary steady-state requirements prior to a successful connection. Private keys needed to establish the identity of the InfiniStream components are described in section 3.8 of [CC_GUIDE] and are established locally on-device. Once the InfiniStream components are joined with the NG1 component, the channel is operational-ready and no modifications are needed to improve the channel security.

241 As background for the examination of the registration channel description, it is noted that the requirements above are intended to ensure that administrators can make an accurate judgement of any risks that arise from the default registration process. Examples would be the use of self-signed certificates (i.e. certificates that are not chained to an external or local Certification Authority), manufacturer-issued certificates (where control over aspects such as revocation, or which devices are issued with recognised certificates, is outside the control of the operational environment), use of generic/non-unique keys (e.g. where the same key is present on more than one instance of a device), or well-known keys (i.e. where the confidentiality of the keys is not intended to be strongly protected – note that this need not mean there is a positive action or intention to publicise the keys).

242 In the case of a distributed TOE for which the ST author uses the FTP_TRP.1/Join channel type in the main selection for FCO_CPC_EXT.1.2 and the TOE relies on the operational environment to provide security for some aspects of the registration channel security then there are additional requirements on the Preparative Procedures as described in section 3.4.1.2.

Note:	The ST does not select FTP_TRP.1/Join.

3.3.1.3 Tests

243 (Note: paragraph 274 lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

244 The evaluator shall carry out the following tests:

a) Test 1.1: the evaluator shall confirm that an IT entity that is not currently a member of the distributed TOE cannot communicate with any component of the TOE until the non-member entity is enabled by a Security Administrator for each of the non-equivalent TOE components[3] that it is required to communicate with (non-equivalent TOE components are as defined in the minimum configuration for the distributed TOE)

High-Level Test Description
Ensure TLS certificates are signed and loaded on the InfiniStream component. Ensure that the InfiniStream certificate root CA is install in NG1 trust store. First logging to the InfiniStream component and clear channel registration information, then log into the NG1 Web GUI and attempt to add the InfiniStream device. The device should fail to be added. Now login to the InfiniStream component and add the channel registration configuration back for NG1 and attempt to add the device again. The device should now be added successfully
Findings: PASS

b) Test 1.2: the evaluator shall confirm that after enablement, an IT entity can communicate only with the components that it has been enabled for. This includes testing that the enabled communication is successful for the enabled component pair, and that communication remains unsuccessful with any other component for which communication has not been explicitly enabled

Some TOEs may set up the registration channel before the enablement step is carried out, but in such a case the channel must not allow communications until after the enablement step has been completed.

245 The evaluator shall repeat Tests 1.1 and 1.2 for each different type of enablement process that can be used in the TOE.

High-Level Test Description
Ensure TLS certificates are signed and loaded. Login to the InfiniStream component and enable access for nGeniusONE ,

[3] An ‘equivalent TOE component’ is a type of distributed TOE component that exhibits the same security characteristics, behaviour and role in the TSF as some other TOE component. In principle a distributed TOE could operate with only one instance of each equivalent TOE component, although the minimum configuration of the distributed TOE may include more than one instance (see discussion of the minimum configuration of a distributed TOE, in section A.9). In practice a deployment of the TOE may include more than one instance of some equivalent TOE components for practical reasons, such as performance or the need to have separate instances for separate subnets or VLANs.

High-Level Test Description
Log into nGeniusONE and add/relearn the InfiniStream component.
Verify that the components are able to maintain communication.
Edit the InfiniStream component on nGeniusONE device configuration menu and update the Note information, then click OK.
Verify that the communication channel is maintained.
Log into the InfiniStream component and remove nGeniusONE access
Log into nGeniusONE and attempt to Relearn the InfiniStream component.
Verify that communication between the components stops.
Findings: PASS

c) Test 2: The evaluator shall separately disable each TOE component in turn and ensure that the other TOE components cannot then communicate with the disabled component, whether by attempting to initiate communications with the disabled component or by responding to communication attempts from the disabled component.

High-Level Test Description
Disable NG1 access on the InfiniStream device,
From NG1 attempted to communicated with the InfiniStream server
The communication attempt shall fail
Reenable NG1 access to the InfiniStream device then attempt to communicated with each component.
Log into NG1 Web GUI and delete the InfiniStream component
Verify that the device is deleted and communication stops
Very that the audit record shows appropriate messages
Findings: PASS

d) Test 3: The evaluator shall carry out the following tests according to those that apply to the values of the main (outer) selection made in the ST for FCO_CPC_EXT.1.2.

1) If the ST uses the first type of communication channel in the selection in FCO_CPC_EXT.1.2 then the evaluator tests the channel via the Evaluation Activities for FTP_ITC.1 or FPT_ITT.1 according to the second selection – the evaluator shall ensure that the test coverage for these SFRs includes their use in the registration process.

2) If the ST uses the second type of communication channel in the selection in FCO_CPC_EXT.1.2 then the evaluator tests the channel via the Evaluation Activities for FTP_TRP.1/Join.

3) If the ST uses the ‘no channel’ selection, then no test is\ required.

Findings: The channel was tested via the Evaluation Activities for FTP_ITT.1.

e) Test 4: The evaluator shall perform one of the following tests, according to the TOE characteristics identified in its TSS and operational guidance:

1) If the registration channel is not subsequently used for inter-component communication, and in all cases where the second selection in FCO_CPC_EXT.1.2 is made (i.e. using FTP_TRP.1/Join) then the evaluator shall confirm that the registration channel can no longer be used after the registration process has completed, by attempting to use the channel to communicate with each of the endpoints after registration has completed

2) If the registration channel is subsequently used for inter-component communication then the evaluator shall confirm that any aspects identified in the operational guidance as necessary to meet the requirements for a steady-state inter-component channel (as in FTP_ITC.1 or FPT_ITT.1) can indeed be carried out (e.g. there might be a requirement to replace the default key pair and/or public key certificate).

High-Level Test Description
Ensure that the InfiniStream certificate root CA is install in NG1 trust store. Follow the instructions in Section 3.8 Trusted Channels for configuring the communication channel and verify that the instructions can be carried out successfully.
Findings: PASS

f) Test 5: For each aspect of the security of the registration channel that operational guidance states can be modified by the operational environment in order to improve the channel security (cf. AGD_PRE.1 refinement item 2 in (cf. the requirements on Preparative Procedures in 3.5.1.2), the evaluator shall confirm, by following the procedure described in the operational guidance, that this modification can be successfully carried out.

High-Level Test Description
Follow the instructions in Section 3.8 Trusted Channels for security configuration related to TLS and securing the communication channel, and verify that the instructions can be carried out successfully.
Findings: PASS

4 Evaluation Activities for Selection-Based Requirements

4.1 Security Audit (FAU)

4.1.1 FAU_GEN_EXT.1 Security Audit Data Generation for Distributed TOE Components

246 For distributed TOEs, the requirements on TSS, Guidance Documentation and Tests regarding FAU_GEN_EXT.1 are already covered by the corresponding requirements for FAU_GEN.1.

4.1.2 FAU_STG_EXT.4 Protected Local audit event storage for distributed TOEs & FAU_STG_EXT.5 Protected Remote audit event storage for Distributed TOEs

4.1.2.1 TSS

247 The evaluator examines the TSS to confirm that it describes which TOE components store their security audit events locally and which send their security audit events to other TOE components for local storage. For the latter, the target TOE component(s) which store security audit events for other TOE components shall be identified. For every sending TOE component the corresponding receiving TOE component(s) need to be identified. For every transfer of audit information between TOE components it shall be described how the data is secured during transfer according to FTP_ITC.1 or FPT_ITT.1.

Findings:	[ST] Section 6.1.4 - Each TOE component stores audit data locally.

248 For each TOE component which does not store audit events locally by itself, the evaluator confirms that the TSS describes how the audit information is buffered before sending to another TOE component for local storage.

Findings:	All TOE components store audit events locally.

4.1.2.2 Guidance Documentation

249 The evaluator shall examine the guidance documentation to ensure that it describes how the link between different TOE components is established if audit data is exchanged between TOE components for local storage. The guidance documentation shall describe all possible configuration options for local storage of audit data and provide all instructions how to perform the related configuration of the TOE components.

Findings:	Audit data is not exchanged between TOE components for local storage. As part of section 3.6 of [CC_GUIDE], each component is configured to ensure they can locally log all necessary entries.

250 The evaluator shall also ensure that the guidance documentation describes for every TOE component which does not store audit information locally how audit information is buffered before transmission to other TOE components.

Findings:	N/A -- All TOE components store audit information locally as per [ST] section 6.1.3.

4.1.2.3 Tests

251 For at least one of each type of distributed TOE components (sensors, central nodes, etc.), the following tests shall be performed using distributed TOEs.

252 Test 1: For each type of TOE component, the evaluator shall perform a representative subset of auditable actions and ensure that these actions cause the generation of appropriately formed audit records. Generation of such records can be observed directly on the distributed TOE component (if there is appropriate interface), or indirectly after transmission to a central location.

High-Level Test Description
Log into each TOE component and perform a sample of management functions to generate audit data. Verify that audit data is being generated and that they meet the audit requirements.
Findings: PASS

253 Test 2: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to the external audit server (as specified in FTP_ITC.1), the evaluator shall configure a trusted channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

Findings:	Verification that the data is encrypted is satisfied by FTP_ITC.1 for the logging channel.

254 Test 3: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to another TOE component (as specified in FTP_ITT.1 or FTP_ITC.1, respectively), the evaluator shall configure a secure channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

Findings:	This feature is not claimed.

255 While performing these tests, the evaluator shall verify that the TOE behaviour observed during testing is consistent with the descriptions provided in the TSS and the Guidance Documentation. Depending on the TOE configuration, there might be a large number of different possible configurations. In such cases, it is acceptable to perform subset testing, accompanied by an equivalency argument describing the evaluator's sampling methodology.

Findings:	The TOE components were found to conform with the statements identified in the TSS: each TOE component independently transmits their own audit logs to an audit server.

4.2 Cryptographic Support (FCS)

4.2.1 FCS_HTTPS_EXT.1 HTTPS Protocol

4.2.1.1 TSS

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

Findings:	[ST] Section 6.3.8 explains in sufficient detail how the HTTPS implementation complies with RFC 2818. The TOE web GUI is accessed via an HTTPS connection using the TLS implementation described by FCS_TLSS_EXT.1.

4.2.1.2 Guidance Documentation

257 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.

Findings:	[CC_GUIDE] Section 3.7 contains instructions on uploading certificates for use in the TOE as an HTTPS server. The TOE cannot be used as an HTTPS client.

4.2.1.3 Tests

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

259 Tests are performed in conjunction with the TLS evaluation activities.

260 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.

4.2.2 FCS_SSHC_EXT.1 SSH Client

4.2.2.1 TSS

FCS_SSHC_EXT.1.2 [Modified by TD0636]

261 The evaluator shall check to ensure that the TSS contains a list of the public key algorithms that are acceptable for use for user authentication and that this list is consistent with asymmetric key generation algorithms selected in FCS_CKM.1, hashing algorithms selected in FCS_COP.1/Hash, and signature generation algorithms selected in FCS_COP.1/SigGen. The evaluator shall confirm the TSS is unambiguous in declaring the TOE's ability to authenticate itself to a remote endpoint with a user-based public key.

Findings:	[ST] Section 6.3.10 - public key for authentication: ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521. These match FCS_CKM.1, FCS_COP.1/Hash, and FCS_COP.1/SigGen.

262 If password-based authentication method has been selected in the FCS_SSHC_EXT.1.2, then the evaluator shall confirm it is also described in the TSS.

Findings:	Password-based authentication has not been selected in the ST.

FCS_SSHC_EXT.1.3

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

Findings:	[ST] Section 6.3.10 - The SSH implementation will detect all large packets greater than 262,144 bytes and drop in accordance with RFC 4253.

FCS_SSHC_EXT.1.4

264 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.

Findings:	[ST] Section 6.3.10 - encryption algorithms: AES128-CTR and AES256-CTR. This matches the selections for this component.

FCS_SSHC_EXT.1.5 [Modified by TD0636]

265 The evaluator shall confirm the TSS describes how a host-key public key (i.e., SSH server’s public key) is associated with the server identity.

266 The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the host-key public key algorithms supported by the TOE are specified as well. The evaluator shall check the TSS to ensure that the host-key public key algorithms specified are identical to those listed for this component.

Findings:	[ST] Section 6.3.10 - public key for authentication: ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521. This matches the selections for this component.

267 If x509v3-based public key authentication algorithms are claimed, the evaluator shall confirm that the TSS includes the description of how the TOE establishes the server’s identity and how this identity is confirmed with the one that is presented in the provided certificate. For example, the TOE could verify that a server’s configured IP address matches the one presented in the server’s x.509v3 certificate.

Findings:	x509v3 based public key authentication algorithms are not claimed.

FCS_SSHC_EXT.1.6

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

Findings:	[ST] Section 6.3.10 - data integrity: hmac-sha2-256, hmac-sha2-512. This matches the component.

FCS_SSHC_EXT.1.7

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

Findings:	[ST] Section 6.3.10 - The [ST] claims diffie-Hellman-group14-sha1, diffie-hellman-group14- sha256, diffie-hellman-group16-sha512, ecdh-sha2-nistp384, ecdh-sha2-nistp521 key agreement schemes which match the selections for the SFR component in [ST] section 5.

FCS_SSHC_EXT.1.8

270 The evaluator shall check that the TSS specifies the following:

1) Both thresholds are checked by the TOE.

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

Findings:	[ST] Section 6.3.10 - The SSH connections will be rekeyed after no more than one hour and no more than one gigabyte of transmitted data.

4.2.2.2 Guidance Documentation

[Modified by TD0636] FCS_SSHC_EXT.1.2

271 The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections initiated by the TOE.

Findings:	[CC_GUIDE] Sections 3.6 provides instructions on configuring the audit log channel to make use of public/private key authentication.

FCS_SSHC_EXT.1.4

272 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).

Findings:	There are no instructions needed to configure the TOE's SSH client to make use of the claimed algorithms. They are configured out-of-the box when using SSH tunnelling.

FCS_SSHC_EXT.1.5

273 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).

Findings:	There are no instructions needed to configure the TOE's SSH client to restrict and permit the claimed host key algorithms for the remote syslog server. They are configured out-of-the box when using SSH tunnelling.

FCS_SSHC_EXT.1.6

274 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).

Findings:	There are no instructions needed to configure the TOE's SSH client to make use of the claimed algorithms. They are configured out-of-the box when using SSH tunnelling.

FCS_SSHC_EXT.1.7

275 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.

Findings:	There are no instructions needed to configure the TOE's SSH client to make use of the claimed algorithms. They are configured out-of-the box when using SSH tunnelling.

FCS_SSHC_EXT.1.8

276 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.

Findings:	The limits are not configurable.

4.2.2.3 Tests

FCS_SSHC_EXT.1.2 [Modified by TD0636]

277 Test objective: The purpose of these tests is to check the authentication of the client to the server using each claimed authentication method.

278 Test 1: For each claimed public-key authentication method, the evaluator shall configure the TOE 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 server to demonstrate the use of all claimed public key algorithms. It is sufficient to observe the successful completion of the SSH Authentication Protocol to satisfy the intent of this test.

High-Level Test Description
Using a test SSH server, force the TOE to connect to the test server with each claimed public key algorithm and verify that the connection attempt is successful.
Findings: PASS

279 Test 2: [Conditional] If password-based authentication method has been selected in the FCS_SSHC_EXT.1.2, then following the guidance documentation the evaluator shall configure the TOE to perform password-based authentication with a remote SSH server to demonstrate that the TOE can successfully authenticate using a password as an authentication method.

Test Not Applicable	The TOE does not claim password-based authentication.

FCS_SSHC_EXT.1.3

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

High-Level Test Description
Using a custom server tool, the evaluator forced the TOE to receive a packet larger than the expected TOE buffer size and showed that the TOE rejects the packet in some way.
Findings: PASS

FCS_SSHC_EXT.1.4

281 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a SSH connection with a remote server (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.

High-Level Test Description
The evaluator permitted the TOE client to connect to a test SSH server and captured the TOE client's advertised supported cipher algorithms. The evaluator verified that the advertised set matches the claimed set.
Findings: PASS

FCS_SSHC_EXT.1.5

282 Test 1: The evaluator shall establish an SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test. Test objective: The purpose of this positive test is to check the authentication of the server by the client (when establishing the transport layer connection), and not for checking generation of the authentication message from the client (in the User Authentication Protocol). The evaluator shall therefore establish sufficient separate SSH connections (with an appropriately configured server) to cause the TOE to demonstrate use of all public key algorithms claimed in FCS_SSHC_EXT.1.5 in the ST.

High-Level Test Description
The evaluator used the TOE client and connected to a test SSH server which only provides a host key using the specified public key algorithms in turn. The evaluator showed that the client successfully authenticated the host.
Findings: PASS

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

284 Test objective: The purpose of this negative test is to verify that the server rejects authentication attempts of clients that present a public key that does not match public key(s) associated by the TOE with the identity of the client (i.e. the public keys are unknown to the server). To demonstrate correct functionality it is sufficient to determine that an SSH connection was not established after using a valid username and an unknown key of supported type.

High-Level Test Description
The evaluator ensured that the TOE has a supported public/private key pair. Then, off-TOE, the evaluator used a different public key (generated with the same public key algorithm). The evaluator then permitted the TOE client to connect to the non-TOE server. The connection attempt failed.
Findings: PASS

FCS_SSHC_EXT.1.6

285 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.

286 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.

High-Level Test Description
Using an SSH Server, the evaluator forcibly permitted only the claimed integrity algorithms and showed that the connections by the TOE SSH client are accepted to form a successful connection.
Findings: PASS

287 Test 2: (conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST) The evaluator shall configure an SSH server to only allow a MAC algorithm that is not included in the ST selection. The evaluator shall attempt to connect from the TOE to the SSH server and observe that the attempt fails.

288 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 .

High-Level Test Description
Using an SSH Server, the evaluator forcibly permitted an integrity algorithm which is not claimed by the TOE and showed that a TOE SSH client connection results in a failed connection.
Findings: PASS

FCS_SSHC_EXT.1.7

289 Test 1: The evaluator shall configure an SSH server to permit all allowed key exchange methods. The evaluator shall attempt to connect from the TOE to the SSH

server using each allowed key exchange method, and observe that each attempt succeeds.

High-Level Test Description
Using an SSH server, the evaluator forcibly permitted only one claimed key exchange mechanism at a time and showed that the TOE client will successfully connect using that algorithm.
Findings: PASS

FCS_SSHC_EXT.1.8

290 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.

291 For testing of the time-based threshold the evaluator shall use the TOE to connect to an SSH server 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).

292 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 server the TOE is connected to.

High-Level Test Description
Using a custom SSH server, the evaluator used the TOE client to connect to the server and trickle data over the channel to avoid disconnection due to idle timeout. The TOE rekeyed before 1 hour had elapsed. Additionally, the TOE was responsible for sending the rekey initiation.
Findings: PASS

293 For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH server 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_SSHC_EXT.1.8).

294 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).

295 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 server the TOE is connected to.

High-Level Test Description
Using a custom SSH server, the evaluator permitted the TOE client to connect to the server. The server sent a large amount of data over the channel back to the client. The evaluator witnessed that the TOE rekeys before 1 GB in the aggregate has been transmitted. The TOE was responsible for sending the rekey initiation.

High-Level Test Description
Findings: PASS

296 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).

Findings:	Rekey thresholds are not configurable.

297 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

Findings:	The TOE does not have hardware limitations.

FCS_SSHC_EXT.1.9

298 Test 1: The evaluator shall delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator shall initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the Security Administrator to accept or deny the key before continuing the connection.

High-Level Test Description
The evaluator first cleared the known host key database. Then using the TOE SSH client, connected to an SSH server and showed that the TOE warned the administrator that the host is unknown and it rejected the connection attempt until after the host key has been manually added.
Findings: PASS

299 Test 2: The evaluator shall add an entry associating a host name with a public key into the TOE's local database. The evaluator shall replace, on the corresponding SSH server, the server's host key with a different host key. If 'password-based' is selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords). If 'password-based' is not selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE

to the SSH server using public key-based authentication and shall ensure that the TOE rejects the connection.

High-Level Test Description
The evaluator added a host key to the known hosts database. A different host key was then generated for the non-TOE SSH server. Using the TOE SSH client, the evaluator attempted to connect to the SSH server that advertises the wrong host key and showed that the TOE rejected the connection.
Findings: PASS

4.2.3 FCS_SSHS_EXT.1 SSH Server

4.2.3.1 TSS

FCS_SSHS_EXT.1.2 [Modified by TD0631]

300 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).

Findings:	[ST] Section 6.3.11 - public key for authentication: ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521. This is consistent with FCS_COP.1/SigGen.

301 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.

Findings:	[ST] Section 6.3.11 - Users are verified when attempting to authenticate via public key through the use of the authorized_keys file.

302 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.

Findings:	[ST] Section 6.3.11 - authentication support: public key-based and password-based

FCS_SSHS_EXT.1.3

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

Findings:	[ST] Section 6.3.11 - The SSH implementation will detect all large packets greater than 262,144 bytes and drop in accordance with RFC 4253.

FCS_SSHS_EXT.1.4

304 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.

Findings:	[ST] Section 6.3.11 - encryption algorithm: AES128-CBC, AES256-CBC, AES128-CTR and AES256-CTR **NOTE:** nGeniusOne supports CBC ciphers and InfiniStream supports CTR ciphers. This matches the selection for this component.

FCS_SSHS_EXT.1.5 [Modified by TD0631]

305 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.

Findings:	[ST] Section 6.3.11 - public key for authentication: ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521. These match the selections for the component.

FCS_SSHS_EXT.1.6

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

Findings:	[ST] Section 6.3.11 - data integrity: hmac-sha2-256, hmac-sha2-512. These match the selections for this component.

FCS_SSHS_EXT.1.7

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

Findings:	[ST] Section 6.3.11 - key agreement scheme: Diffie-Hellman-group14-sha1. This matches the key agreement scheme listed in section 5 of the [ST].

FCS_SSHS_EXT.1.8

308 The evaluator shall check that the TSS specifies the following:

3) Both thresholds are checked by the TOE.

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

Findings:	[ST] Section 6.3.11 - The SSH connections will be rekeyed after no more than one hour and no more than one gigabyte of transmitted data.

4.2.3.2 Guidance Documentation

FCS_SSHS_EXT.1.4

309 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).

Findings:	There are no instructions needed to configure the TOE's SSH server to make use of the claimed algorithms. They are configured out-of-the box.

FCS_SSHS_EXT.1.5

310 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).

Findings:	[CC_GUIDE] section 3.1 includes instructions to restrict the set of advertised host key algorithms sent to a remote SSH client using the "HostKeyAlgorithms" configuration parameter. The restricted set is consistent with the claimed host public key algorithms in the [ST]. In addition, the same section stipulates "PubKeyAcceptedKeyTypes" as a way to restrict the set of remote user-based public key algorithms to permit. The restricted set is consistent with the claimed user public key algorithms in the [ST].

FCS_SSHS_EXT.1.6

311 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).

Findings:	There are no instructions needed to configure the TOE's SSH server to make use of the claimed algorithms. They are configured out-of-the box.

FCS_SSHS_EXT.1.7

312 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.

Findings:	There are no instructions needed to configure the TOE's SSH server to make use of the claimed algorithms. They are configured out-of-the box.

FCS_SSHS_EXT.1.8

313 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.

Findings:	The thresholds are not configurable. They are fixed in place as part of the initial configuration of the TOE described in section 3.1 of [CC_GUIDE] using the "RekeyLimit" setting.

4.2.3.3 Tests

FCS_SSHS_EXT.1.2 [Modified by TD0631]

314 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.

High-Level Test Description
For each supported client public-key authentication algorithm, the evaluator configured the TOE to bind an SSH public key of the given algorithm to a defined user. Then the evaluator used an SSH client to attempt to use the private key to authenticate with the TOE as that user. The evaluator showed the authentication was successful.
Findings: PASS

315 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.

High-Level Test Description
The evaluator configured the TOE to bind an SSH public key of the given algorithm to a defined user. Then the evaluator used an SSH client to attempt to use a different private key of the same algorithm to authenticate with the TOE as that user. The evaluator showed the authentication failed.
Findings: PASS

316 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.

Note	This test was conducted as part of FIA_UIA_EXT.1/FIA_UAU_EXT.2.

317 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.

Note	This test was conducted as part of FIA_UIA_EXT.1/FIA_UAU_EXT.2.

FCS_SSHS_EXT.1.3

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

High-Level Test Description
Using a custom tool, the evaluator transmitted a packet larger than the expected TOE buffer size and showed that the TOE rejects the packet in some way.
Findings: PASS

FCS_SSHS_EXT.1.4

319 The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a 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.

High-Level Test Description
Using an SSH client, the evaluator connected to the TOE SSH server and captured the TOE server's advertised supported cipher algorithms. The evaluator verified that the advertised set matches the claimed set. The evaluator then used an SSH client to connect to the TOE using only one of those ciphers and show that the connection is successful.
Findings: PASS

FCS_SSHS_EXT.1.5 [Modified by TD0631]

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

321 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.

High-Level Test Description
The evaluator used an SSH client to confirm that the client can authenticate the TOE host public key using the claimed host public key algorithm.
Findings: PASS

322 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.

323 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.

High-Level Test Description
The evaluator loaded a supported public key into the TOE. Then Off-TOE, the evaluator used a different private key (generated with a different, unsupported public key algorithm) to try to connect. The connection attempt failed.

High-Level Test Description
Findings: PASS

FCS_SSHS_EXT.1.6

324 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.

325 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.

High-Level Test Description
Using an SSH client, the evaluator forcibly negotiated only the claimed integrity algorithms and showed that they are accepted to form a successful connection.
Findings: PASS

326 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.

327 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.

High-Level Test Description
Using an SSH client, the evaluator forcibly negotiated an integrity algorithm which is not claimed by the TOE and showed that it resulted in a failed connection.
Findings: PASS

FCS_SSHS_EXT.1.7

328 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.

High-Level Test Description
Using an SSH client, the evaluator forcibly negotiated the diffie-hellman-group1-sha1 key exchange algorithm which is not supported by the TOE and showed that it resulted in a failed connection.
Findings: PASS

329 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.

High-Level Test Description
Using an SSH client, the evaluator forcibly negotiated each of the claimed key exchange algorithms in turn and showed that it resulted in a successful connection.
Findings: PASS

FCS_SSHS_EXT.1.8

330 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.

331 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).

332 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.

High-Level Test Description
Using a custom SSH client, the evaluator connected to the TOE and trickled data over the channel to avoid disconnection due to idle timeout. The evaluator observed that the TOE rekeyed before 1 hour elapsed. The TOE was found to be responsible for sending the rekey initiation.
Findings: PASS

333 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).

334 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).

335 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.

High-Level Test Description
Using a custom SSH client, the evaluator connected to the TOE to send large amounts of data over the channel. The evaluator verified that the TOE rekeyed before 1 GB in the aggregate had been transmitted, and that the TOE was responsible for sending the rekey initiation.
Findings: PASS

336 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).

Findings:	The thresholds are not configurable.

337 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

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.

Findings:	The TOE does not have hardware limitations.

4.2.4 FCS_TLSC_EXT.1 Extended: TLS Client Protocol without mutual authentication

4.2.4.1 TSS

FCS_TLSC_EXT.1.1

338 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.

Findings:	[ST] Section 6.3.12 contains a list of allowed ciphersuites and they match the selection in the component in section 5.3.3.

FCS_TLSC_EXT.1.2

339 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. The evaluator shall ensure that this description identifies if certificate pinning is supported or used by the TOE and how it is implemented.

Findings:	[ST] Section 6.3.12 - The TOE supports the use of DNS as reference identifiers and can be found in both the CN and SAN of the certificate. The TOE's TLS implementation will only support a wildcard in the left-most label (e.g. *.example.com). All other usages of a wildcard will cause a failure in the connection. The TOE does not support service name reference identifiers, IP addresses, or pinned certificates.

340 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.

Findings:	The ST author did not select attributes from RFC 5280.

341 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.

Findings:	[ST] Section 6.3.12 does not claim IP addresses as a reference identifier.

FCS_TLSC_EXT.1.4

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

Findings:	[ST] Section 6.3.12 -The TOE will present Supported Elliptical Curve extensions in the Client Hello. The only allowable NIST curve is: secp256r1.

4.2.4.2 Guidance Documentation

FCS_TLSC_EXT.1.1

343 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.

Findings:	[CC_GUIDE] Section 3.1 provides instructions on initially configuring the ciphers for the TOE TLS client. However, once configured, these ciphers are not configurable during operation. The set of ciphers described in section 3.1 of the [CC_GUIDE] are consistent with those described in the [ST].

FCS_TLSC_EXT.1.2

344 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.

Findings:	Section 3.8 of [CC_GUIDE] contains configuration information on peer certificate identifiers and CA certificate policy. Specifically, the [CC_GUIDE] indicates that "*[e]ach TOE component certificate supports the use of the SAN extension*." IP addresses are explicitly not used for reference identifiers as per directions provided in section 3.8 of [CC_GUIDE].

345 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.

Findings:	FCO_CPC_EXT.1.2 does not select “no channel”. Rather, FCS_TLSC_EXT.1 is exclusively used for the distributed TOE and claimed under FPT_ITT.1 using RFC 6125 section 6.

FCS_TLSC_EXT.1.4

346 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.

Findings:	[CC_GUIDE] Section 3.1 provides instructions on initially configuring the key sizes for the TOE TLS client. However, once configured, the key size is no longer configurable.

4.2.4.3 Tests

FCS_TLSC_EXT.1.1

347 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).

High-Level Test Description
Using a Lightship developed TLS server, the evaluator forced the TOE client to negotiate all specifically claimed ciphersuites.
Findings: PASS

348 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.

High-Level Test Description
The evaluator constructed two X.509 certificates: one with an extendedKeyUsage with ‘serverAuth’ and another without. The evaluator forced the TOE client to attempt a handshake with a test server and showed that the X.509 certificate without the serverAuth in the EKU field failed.
Findings: PASS

349 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.

High-Level Test Description
Using a Lightship developed TLS server, the evaluator forced the TOE client to attempt a handshake with a test server using any of the claimed ciphersuites. The Lightship TLS server sends back an otherwise validly constructed server certificate which does not match the requested ciphersuite and the evaluator verified that the connection failed.
Findings: PASS

350 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.

High-Level Test Description
Using a Lightship developed TLS server, the evaluator forced the TOE client to attempt a handshake with a test server using TLS_NULL_WITH_NULL_NULL (cipher ID 0x0000). The evaluator verified that the TOE denied the connection.
Findings: PASS

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.

High-Level Test Description
Using a Lightship developed TLS server, the evaluator forced the TOE client to attempt a handshake with a test server sending a non-negotiated ciphersuite. The evaluator verified that the TOE rejected the connection.
Findings: PASS

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.

High-Level Test Description
The evaluator forced the TOE client to connect to a Lightship TLS server which used an unsupported EC curve. The evaluator verified that TOE disconnected after receiving the server's Key Exchange handshake message.
Findings: PASS

351 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.

High-Level Test Description
Using a Lightship developed TLS server, the evaluator forced the TOE client to attempt a handshake with a test server advertising an incorrect TLS version. The evaluator verified that the TOE rejected the connection.
Findings: PASS

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.

High-Level Test Description
Using a Lightship developed TLS server, the evaluator forced the TOE client to attempt a handshake with a test server sending a mangled key exchange signature. The evaluator verified that the handshake did not finish successfully, and no application data flowed.
Findings: PASS

352 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.

High-Level Test Description
Using a Lightship developed TLS server, the TOE client was forced to attempt a handshake with a test server sending a mangled finished message and the connection failed. No application data flowed.
Findings: PASS

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.

High-Level Test Description
Using a Lightship developed TLS server, the TOE client was forced to attempt a handshake with a test server sending a garbled message before the ChangeCipherSpec was sent and the connection failed. No application data flowed.
Findings: PASS

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.

High-Level Test Description
The TOE client was forced to attempt a handshake with a Lightship developed test server. The server passed an invalid nonce to the client which resulted in the client terminating the handshake after the Server Key Exchange was received. No application data flowed.
Findings: PASS

FCS_TLSC_EXT.1.2

353 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.

354 Note that for some tests additional conditions apply.

355 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.

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

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

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

High-Level Test Description
The evaluator forced the TOE client to attempt a handshake using a custom test server to send X.509 certificates that have the characteristics required by the test. The connection failed to establish.
Findings: PASS

b. Test 2 [conditional]: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, URI). When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the SAN.

High-Level Test Description
The evaluator forced the TOE client to attempt a handshake using a custom test server to send X.509 certificates that have the characteristics required by the test. The connection failed to establish.
Findings: PASS

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

High-Level Test Description
The TOE mandates the presence of the SAN extension for all claimed identifier types.
Findings: N/A

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

High-Level Test Description
The evaluator forced the TOE client to attempt a handshake using a custom test server to send X.509 certificates that have the characteristics required by the test. The connection is successfully established.
Findings: PASS

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

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

High-Level Test Description
The evaluator forced the TOE client to attempt a handshake using a custom test server to send X.509 certificates that have the characteristics required by the test. The connection failed to establish.
Findings: PASS

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.)

High-Level Test Description
The evaluator forced the TOE client to attempt a handshake using a custom test server to send X.509 certificates that have the characteristics required by the test. The connection was a success with a wildcard, and with a reference identifier in the left-most label. All other configurations failed.
Findings: PASS

f. **[Modified by TD0634]**

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

Test 6 [conditional]: If IP address identifiers supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.0.1, CN=2001:0DB8:0000:0000:0008:0800:200C:* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6). 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.

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

High-Level Test Description
The TOE does not claim the use of IP addresses as a reference identifier.
Findings: N/A

g. 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-at-serialNumber=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.)

Findings:	FPT_ITT with RFC 5280 is not claimed.

FCS_TLSC_EXT.1.3

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

358 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.

Note	This test case is performed as part of FIA_X509_EXT.1.

359 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.

Note	This test case is performed as part of FIA_X509_EXT.1.

360 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.

Note	No administrative override.

FCS_TLSC_EXT.1.4

361 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.

High-Level Test Description
The evaluator forced the TOE client to connect to a custom test server with a supported ciphersuite and supported elliptic curve algorithm. The connection is successfully established.
Findings: PASS

4.2.5 FCS_TLSS_EXT.1 Extended: TLS Server Protocol without mutual authentication

4.2.5.1 TSS

FCS_TLSS_EXT.1.1

362 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.

Findings:	[ST] Section 6.3.13 - The server only allows TLS protocol version 1.2 (rejecting any other protocol version) and is restricted to the following ciphersuites by default: a) TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289 b) TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289 c) TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289 d) TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289 This information is consistent with the selections for this component.

FCS_TLSS_EXT.1.2

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

Findings:	[ST] Section 6.3.13 - The server only allows TLS protocol version 1.2 (rejecting any other protocol version).

FCS_TLSS_EXT.1.3 [Modified by TD0635]

364 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.

Findings:	[ST] Section 6.3.13 - The TLS server is capable of negotiating ciphersuites that include ECDHE key agreement schemes. The TOE supports key establishment using ecdhe curve secp256r1.

FCS_TLSS_EXT.1.4 [Modified by TD0569]

365 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).

366 If the TOE claims a (D)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.

Findings:	[ST] Section 6.3.13 - The NG1 TLS server supports session resumption using session tickets according to RFC 5077. Session tickets are encrypted using AES-CBC symmetric algorithms, using key size of 128 consistent with FCS_COP.1/DataEncryption. The InfiniStream TLS server supports no session resumption or session tickets.

367 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.

Findings:	[ST] Section 6.3.13 - The TLS server support session resumption using session tickets according to RFC 5077. Session tickets are encrypted using AES-CBC symmetric algorithms, using key size of 128 consistent with FCS_COP.1/DataEncryption.

368 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.

Findings:	[ST] Section 6.3.13 - The TLS server support session resumption using session tickets according to RFC 5077.

369 If the TOE claims a (D)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.

Findings:	[ST] Section 6.3.13 - The TLS server support session resumption using session tickets according to RFC 5077. Session tickets are encrypted using AES-CBC symmetric algorithms, using key size of 128 consistent with FCS_COP.1/DataEncryption.

4.2.5.2 Guidance Documentation

FCS_TLSS_EXT.1.1

370 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).

Findings:	[CC_GUIDE] Section 3.1 provides instructions on initially configuring the ciphers for the TOE TLS server.

FCS_TLSS_EXT.1.2

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

Findings:	There are no instructions needed to configure the TOE's TLS server to restrict or enable protocol versions. They are configured out-of-the box.

FCS_TLSS_EXT.1.3

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

Findings:	[CC_GUIDE] Section 3.1 provides instructions on initially configuring the elliptic curves used within the TLS server. However, once configured, they are not managed operationally.

[Modified by TD0569] FCS_TLSS_EXT.1.4

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

Findings:	There are no instructions needed to configure the TOE's TLS server to configure support for session resumption. They are configured out-of-the box.

4.2.5.3 Tests

FCS_TLSS_EXT.1.1

374 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).

High-Level Test Description
Using a Lightship developed TLS client, the evaluator connected to the TOE using the claimed ciphersuites.
Findings: PASS

375 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.

High-Level Test Description
Using a Lightship developed TLS client, the evaluator connected to the TOE using an unsupported ciphersuite. Then connected to the TOE using TLS_NULL_WITH_NULL_NULL and witnessed that the connection attempt failed.

High-Level Test Description
Findings: PASS

376 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.

High-Level Test Description
Using a Lightship developed TLS client, the evaluator connected to the TOE and modified the first payload byte in the Client Finished message, and witnessed that the connection attempt failed.
Findings: PASS

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'.

High-Level Test Description
The evaluator performed a successful handshake using one of the accepted ciphersuites and verified that the Server Finished message is encrypted.
Findings: PASS

FCS_TLSS_EXT.1.2

377 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.

High-Level Test Description
Using a Lightship developed TLS client, the evaluator connected to the TOE and attempted to negotiate SSL 2.0, SSL 3.0, TLS 1.0 and TLS 1.1 . The evaluator observed that only claimed protocols are accepted.
Findings: PASS

FCS_TLSS_EXT.1.3

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

d. 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.

High-Level Test Description
Using a Lightship developed TLS client, the evaluator connected to the TOE using a valid ECDHE ciphersuite and curve combination and verified that the public key size that comes back in the Server Key Exchange message matches the expected bit size for the chosen curve.
Findings: PASS

e. 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.

High-Level Test Description
Using a Lightship developed TLS client, the evaluator connected to the TOE using a valid ECDHE ciphersuite and an unsupported curve and verified that the TOE fails to send back a Server Hello message and terminates the connection.
Findings: PASS

379 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).

Findings:	N/A, DHE ciphersuites not supported.

380 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.

Findings:	N/A, RSA ciphersuites not supported.

FCS_TLSS_EXT.1.4 [Modified by TD0569]

381 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).

382 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.

Findings:	NG1 components support session resumption based on session tickets according to RFC5077. InfiniStream components do not support session resumption.

High-Level Test Description
Show that the InfiniStream TOE component does not claim the use of either session IDs or session tickets for the purposes of session resumption.
Findings: PASS

383 **[Modified by 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.

Findings:	NG1 components support session resumption based on session tickets according to RFC5077. InfiniStream components do not support session resumption.

384 **[Modified by 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) **[Modified by TD0556]** 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.

Findings:	NG1 components support session resumption based on session tickets according to RFC5077. InfiniStream components do not support session resumption.

High-Level Test Description
The evaluator showed that the TOE will handle session resumption via Session Tickets as per the provided test steps. Using a custom tool the evaluator connected to the TOE and verified session resumption base on session ticket. The test succeeded. Using a custom tool the evaluator connected to the TOE and verified that the session resumption fails.
Findings: PASS

4.3 Identification and Authentication (FIA)

4.3.1 FIA_X509_EXT.1/Rev X.509 Certificate Validation

4.3.1.1 TSS

385 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).

Findings:	[ST] Section 6.4.5 - The NG1 supports a minimum path length of three certificates for the Web GUI. The certificate path is terminated in a trusted CA certificate, the validity of dates are checked, the basicConstraints extension is present, and the CA flag is

	set to TRUE for all CA certificates. Finally, the TOE ensures the extendedKeyUsage field includes the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) for server certificates used in TLS, or the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) for OCSP certificates used for OCSP. There are no exceptions to the rules for extendedKeyUsage fields.

386 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.

Findings:	[ST] Section 6.4.5 - As part of the certificate validation checking, the TSF will validate certificate revocation status using an OCSP server in the Operational Environment.

4.3.1.2 Guidance Documentation

387 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.

Findings:	[CC_GUIDE] Section 3.8 - The TOE performs certificate validity checking for the TLS connection between TOE components. As part of the certificate validation checking, the TSF will validate certificate revocation status using an OCSP server in the Operational Environment. If the revocation status cannot be verified, the certificate will be rejected. The TOE ensures the extendedKeyUsage field includes the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) for server certificates used in TLS, or the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) for OCSP certificates used for OCSP.

4.3.1.3 Tests

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

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

b) 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.

High-Level Test Description
The evaluator created a root CA and intermediate CA (signed by the root) certificates. Then loaded the root and intermediate CA certificate into the TOE's trust store successfully. The evaluator then removed the root and intermediate CA certificates from the TOE trust store, then attempted to load only the intermediate into the TOE trust store and observed that this fails.
Findings: PASS

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

High-Level Test Description
The evaluator created/cloned an X.509 certificate with a 'notAfter' date in the past, then attempt to import the expired certificate and observed that the function failed.
Findings: PASS

d) 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.

High-Level Test Description
The evaluator loaded the CA certificate into the TOE trust store, then with the OCSP responder listening with valid index file. The evaluator verified that the certificate is loaded successfully. Then revoked the server certificate and restarted the OCSP server. The evaluator then attempted to load the certificate and verified that the certificate import fails due to the certificate being revoked. Then unrevoked the certificate from the OCSP and restart the OCSP server. The evaluator also revoked the intermediate CA and restart the root CA OCSP server. The evaluator then attempted to load the revoked certificate observed that the import failed due to the certificate being revoked. Then unrevoked the intermediate CA and restart the OCSP server. After which the certificate results in a successful connection.
Findings: PASS

e) 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.

High-Level Test Description
The evaluator created an OCSP signing certificate using a known good CA certificate that has the OCSPSigning extendedKeyUsage flag enabled. The known good certificate was then cloned to remove the OCSPSigning extendedKeyUsage. The OCSP signature only depends on the (cloned) private key of the CA used to sign it and the TOE does not engage in any certificate pinning. The old OCSP certificate was replaced with the newly cloned certificate, and a new request as initiated by attempting to import a certificate into the trust store.. It was observed that the connection now fails due to the OCSP response being signed by a CA without the proper flag.
Findings: PASS

f) 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.)

High-Level Test Description
The evaluator modified one of the first 8 byte of the Web server certificate and attempted to install it on the TOE. The certificate validation failed.
Findings: PASS

g) Test 6: The evaluator shall modify any byte in the certificate signatureValue field (see RFC5280 Sec. 4.1.1.3), which is normally the last field in the certificate, and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

High-Level Test Description
The evaluator modified the signature field of the x509 certificate and attempted to install it into the trust store. The certificate validation failed.
Findings: PASS

h) 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.)

High-Level Test Description
The evaluator modified a byte in the public key of the certificate and demonstrate that the certificate fails to validate. The certificate validation failed.
Findings: PASS

[Modified by TD0527]

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

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

391 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.

Findings:	N/A, the TOE does not process CA certificates presented in certificate message.

392 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.

Findings:	N/A, the TOE does not process CA certificates presented in certificate message.

393 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.

High-Level Test Description
The evaluator constructed a chain of three ECDSA certificates: a leaf, an intermediate CA and a trust anchor, then showed that the leaf and chain are valid. The evaluator reran the test by creating a clone of the Intermediate CA, such that the public key is explicitly defined rather than being a named curve, and showed that the leaf and chain are not validated correctly.
Findings: PASS

394 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 in functions the 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.

395 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).

396 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).

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

High-Level Test Description
Load a known-good CA into the TOE trust store. Verify that the import is successful. Clone the known good CA certificate and remove the basicConstraints extension. Attempt to install the clone certificate. Verify that the import fails.
Findings: PASS

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

High-Level Test Description
Clone the known good CA certificate and set the basicConstraints extension to have the CA flag set to FALSE. Attempt to install the clone certificate. Verify that the import fails.
Findings: PASS

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).

Note:	Certificates in this context are only used for the NG1 Web UI.

4.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication

4.3.2.1 TSS

399 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.

Findings:	[ST] Section 6.4.5 - The NG1 component chooses its server certificate by using the "nscertutil" to import the certificate, validate it against the CAs in the trust store and then placing it into the correct path. The InfiniStream component chooses its server certificate by first importing the Certificate Response to the trust store, and then configuring the "lighthttpd.pem" file with the certificates and key as a bundle.

400 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.

Findings:	[ST] Section 6.4.5 - In the event that the revocation status cannot be verified, the certificate will be rejected.

4.3.2.2 Guidance Documentation

401 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.

Findings:	Section 3.8 of [CC_GUIDE] outlines how the TOE stores and uses certificates. Section 3.8 of [CC_GUIDE] describes OCSP configuration requirements, and states that when the validity of a certificate cannot be established due to a failed connection to the OCSP responder, the certificate not be accepted, no Administrator override is possible.

4.3.2.3 Tests

402 The evaluator shall perform the following test for each trusted channel:

403 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.

High-Level Test Description
The evaluator stopped the OCSP responders server. The evaluator showed that when the OCSP responder is unavailable, that any attempt to validate a certificate will fail.
Findings: PASS

4.3.3 FIA_X509_EXT.3 Extended: X509 Certificate Requests

4.3.3.1 TSS

404 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.

Findings:	“Device-specific information” is not selected for FIA_X509_EXT.1.3.

4.3.3.2 Guidance Documentation

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.

Findings:	Section 3.8 of [CC_GUIDE] provides information on generating the Certificate Request from each TOE component. In addition, the same section provides information on when to supply the CSR to the CA for signing and how to import the resulting signed certificate. The [ST] claims Common Name, Organization, Organizational Unit, and Country. The [CC_GUIDE] section 3.8 provides instructions for establishing these fields. Additional instructions for the NG1 to support HTTPS over TLS for the Web UI can be found in Section 3.7 of [CC_GUIDE].

4.3.3.3 Tests

405 The evaluator shall perform the following tests:

a) Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a Certification Request. The evaluator shall capture the generated message 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.

High-Level Test Description
Using the TOE CSR generator, the evaluator created a new CSR and downloaded it to an external CA entity for signing. Using OpenSSL, the evaluator verified that the information in the CSR is as expected.
Findings: PASS

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

High-Level Test Description
The CSR from the previous test was signed and copied to the TOE. The evaluator attempted to import the certificate into the trust store without the signing CA in the trust store and observed that the import fails. The signing CA certificate was then installed on the TOE after which the certificate was imported successfully.
Findings: PASS

4.4 Security management (FMT)

4.4.1 FMT_MTD.1/CryptoKeys Management of TSF Data

4.4.1.1 TSS

406 For distributed TOEs see chapter 2.4.1.1.

Findings:	See section 2.4.1.1.

407 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.

Findings:	The TOE is a distributed TOE.

4.4.1.2 Guidance Documentation

408 For distributed TOEs see chapter 2.4.1.2.

Findings:	See chapter 2.4.1.2.

409 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.

Findings:	The TOE is a distributed TOE.

4.4.1.3 Tests

410 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.

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

High-Level Test Description
The evaluator attempted to log into the CLI as user without security administrator privileges and attempted to modify, add, or delete crypto keys. The evaluator verified that any attempt by the user was unsuccessful.
Findings: PASS

4.4.2 FMT_MOF.1/Functions Management of security functions behaviour

4.4.2.1 TSS

412 For distributed TOEs see chapter 2.4.1.1.

Findings:	See section 2.4.1.1.

413 For non-distributed TOEs, the evaluator shall ensure the TSS for each administrative function identified the TSS details how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE).

Findings:	The TOE is a distributed TOE.

4.4.2.2 Guidance Documentation

414 For distributed TOEs see chapter 2.4.1.2.

Findings:	See chapter 2.4.1.2.

415 For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE) are performed to include required configuration settings.

Findings:	The TOE is a distributed TOE.

4.4.2.3 Tests

416 Test 1 (if ‘transmission of audit data to external IT entity’ is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity without prior

authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters 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 modify the security related parameters 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.

High-Level Test Description
The evaluator logged into the NG1 CLI as a user without permissions for modifying the audit tunnel and attempted to access the audit tunnel configuration files. The InfiniStream components only have security administrators. Users do not have access to the management interface prior to authenticating, therefore, only the security administrator is allowed to perform management functions.
Findings: PASS

417 Test 2 (if ‘transmission of audit data to external IT entity’ is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as Security Administrator. The effects of the modifications should be confirmed.

418 The evaluator does not have to test all possible values of the security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity but at least one allowed value per parameter.

High-Level Test Description
Using the privileged security administrator, the evaluator modified the IP and port of the syslog provider, and verified that the TOE attempts to communicate using the new parameters.
Findings: PASS

419 Test 1 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters 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 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 term ‘handling of audit data’ refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

Findings:	NA, handling of audit data' is not selected from the second selection.

420 Test 2 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data with prior authentication as Security Administrator. The effects of the modifications should be

confirmed. The term ‘handling of audit data’ refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

421 The evaluator does not necessarily have to test all possible values of the security related parameters for configuration of the handling of audit data but at least one allowed value per parameter.

Findings:	‘Handling of audit data' is not selected from the second selection.

422 Test 1 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full without prior authentication as Security Administrator (by authentication as_a user with no administrator privileges or without user authentication at all). This attempt 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 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.

Note:	The TOE does not claim this functionality and this test will not be conducted.

423 Test 2 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full with prior authentication as Security Administrator. This attempt should be successful. The effect of the change shall be verified.

424 The evaluator does not necessarily have to test all possible values for the behaviour when Local Audit Storage Space is full but at least one change between allowed values for the behaviour.

Note:	The TOE does not claim this functionality and this test will not be conducted.

425 Test 3 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions without administrator authentication shall 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 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.

Note:	The TOE does not claim this functionality and this test will not be conducted.

426 Test 4 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection with prior authentication as Security Administrator. This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions with Security Administrator authentication shall be successful.

Note:	The TOE does not claim this functionality and this test will not be conducted.

4.4.3 FMT_MOF.1/Services Management of security functions behaviour

4.4.3.1 TSS

427 For distributed TOEs see chapter 2.4.1.1.

Findings:	See section 2.4.1.1.

428 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.

Findings:	The TOE is a distributed TOE.

4.4.3.2 Guidance Documentation

429 For distributed TOEs see chapter 2.4.1.2.

Findings:	See chapter 2.4.1.2.

430 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.

Findings:	The TOE is a distributed TOE.

4.4.3.3 Tests

431 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.

High-Level Test Description
The evaluator logged into the TOE as a user without the security administrator privileges and attempted to enable/disable a service provided by the TOE. The evaluator was unable to execute any functions.
Findings: PASS

432 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.

High-Level Test Description
The evaluator logged into the TOE as the security administrator and attempted to enable/disable a service provided by the TOE. The functions executed successfully.
Findings: PASS

5 Evaluation Activities for Security Assurance Requirements

5.1 ASE: Security Target

433 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).

Findings:	The evaluator performed the work units as presented in the CEM, in addition to the EAs in the sections above.

434 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.

ASE_TSS.1 element	Evaluator Action
ASE_TSS.1.1C	The evaluator shall examine the TSS to determine that it is clear which TOE components contribute to each SFR or how the components combine to meet each SFR. The evaluator shall verify the sufficiency to fulfil the related SFRs. This includes checking that the TOE as a whole fully covers all SFRs and that all functionality that is required to be audited is in fact audited regardless of the component that carries it out.

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

Findings:	[ST] Section 5.3 Table 12 contains this information.

436 A.9.1 Evaluator Actions for Assessing the ST

437 A.9.1.1 TSS

438 The evaluator shall examine the TSS to identify any extra instances of TOE components allowed in the ST and shall examine the description of how the additional components maintain the SFRs to confirm that it is consistent with the role that the component plays in the evaluated configuration. For example: the secure channels used by the extra component for intra-TOE communications (FPT_ITT) and external communications (FTP_ITC) must be consistent, the audit information generated by the extra component must be maintained, and the management of the extra component must be consistent with that used for the original instance of the component in the minimum configuration.

Findings:	The ST states in Section 6.2.1 that the minimum configuration is the deployment of an NG1 and one InfiniStream. Multiple InfiniStreams can be deployed and maintain their own separate communication channels with the NG1 and external IT entities that comply with FPT_ITT.1 and FTP_ITC.1. The InfiniStreams do not communicate with each other.

5.2 ADV: Development

439 The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any required supplementary information required by this cPP that is not to be made public.

440 The functional specification describes the TOE Security Functions Interfaces (TSFIs). It is not necessary to have a formal or complete specification of these interfaces.

441 No additional "functional specification" documentation is necessary to satisfy the Evaluation Activities specified in [SD].

442 The Evaluation Activities in [SD] are associated with the applicable SFRs; since these are directly associated with the SFRs, the tracing in element ADV_FSP.1.2D is implicitly already done and no additional documentation is necessary.

443 5.2.1.1 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.

Findings:	From section 7.2.1 of the NDcPP: "For this cPP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation." The [ST] and the AGD comprise the functional specification. If the test in [SD] cannot be completed because the [ST] or the AGD is incomplete, then the functional specification is not complete and observations are required. During the evaluator's use of the product and its interfaces (the Web GUI, SSH CLI, local serial port), there were no areas that were deficient.

444 5.2.1.2 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.

Findings:	See comments in the previous work unit.

445 5.2.1.3 Evaluation Activity: The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

Findings:	See comments in the previous work unit.

5.3 AGD: Guidance

446 The design information about the TOE is contained in the guidance documentation available to the end user as well as the TSS portion of the ST, and any required supplementary information required by this cPP that is not to be made public.

447 5.3.1.1 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.

Findings:	Registered users can download the guidance documents from NETSCOUT's web portal: https://www.netscout.com/support-services.

448 5.3.1.2 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.

Findings:	There is only one operational environment claimed in the [ST]. All TOE platforms claimed in [ST] are covered by the operational guidance.

449 5.3.1.3 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.

Findings:	The [CC_GUIDE] section 1.4.3 provides this statement.

450 5.3.1.4 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.

Findings:	The [CC_GUIDE] section 1.4.3 indicates the in-scope functionality. Furthermore, section 3.3 of [CC_GUIDE] provide an overview of the in-scope management interfaces. There is a statement in section 3.1 of [CC_GUIDE] that iDRAC has not been evaluated and should be disconnected from the network.

451 5.3.1.5 Evaluation Activity: 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) **[NIAP TD0536]** 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:

5) 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).

6) 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.

Findings:	The [CC_GUIDE] Section 2 provides instructions for obtaining and verifying TOE updates. See work unit 5.3.1.3 for configuration of the cryptographic engine. See work unit 5.3.1.4 for details as to what was covered by the EAs.

452 5.3.2.1 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).

Findings:	The [CC_GUIDE] section 1.4.3 provides a brief description of the components operating in the operational environment, including the OCSP responder, and the syslog server. The explicit use of OCSP responders are described in section 3.8 of [CC_GUIDE]. The explicit configuration and use of the syslog over SSH channel is described in section 3.6 of [CC_GUIDE]. Security objectives for the Operational Environment are covered by Assumptions described in section 1.4.4 of [CC_GUIDE]. This table provides guidance to administrators to ensure that their environment can meet the requirements.

453 5.3.2.2 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.

Findings:	There is only one operational environment claimed in the [ST] and [CC_GUIDE]. All TOE platforms claimed in [ST] are covered by the operational guidance.

454 5.3.2.3 Evaluation Activity: The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

Findings:	See previous work unit.

455 5.3.2.4 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.

Findings:	The guidance documentation in [CC_GUIDE] and [ADMIN] provides extensive information on managing the security of the TOE as an individual product. Additional best practice guidance provided within those documents help instil a culture of secure manageability within a larger operational environment.

456 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.

Findings:	Section 3.3 of the [CC_GUIDE] provides instructions for the protection of the administration interfaces, while Section 3.7 describes the internal authentication mechanisms. Additionally, Section 3.4 of the [CC_GUIDE] includes instructions for changing default passwords.

A.9.2 Evaluator Actions for Assessing the Guidance Documentation

A.9.2.1 Guidance Documentation

The evaluator shall examine the description of the extra instances of TOE components in the guidance documentation to confirm that they are consistent with those identified as allowed in the ST. This includes confirmation that the result of applying the guidance documentation to configure the extra component will leave the TOE in a state such that the claims for SFR support in each component are as described in the ST and therefore that all SFRs continue to be met when the extra components are present.

Findings:	Section 3.8 of the [CC_GUIDE] states that the minimum configuration is the deployment of an NG1 and one InfiniStream. Multiple InfiniStreams can be deployed and maintain their own separate communication channels with the NG1 and external IT entities that comply with FPT_ITT.1 and FTP_ITC.1. The InfiniStreams do not communicate with each other.

The evaluator shall examine the secure communications described for the extra components to confirm that they are the same as described for the components in the minimum configuration (additional connections between allowed extra components and the components in the minimum configuration are allowed of course).

Findings:	Section 3.8 of the [CC_GUIDE] states that the minimum configuration is the deployment of an NG1 and one InfiniStream. Multiple InfiniStreams can be deployed and maintain their own separate communication channels with the NG1 and external IT entities that comply with FPT_ITT.1 and FTP_ITC.1. The InfiniStreams do not communicate with each other.

5.4 ALC: Life-cycle Support

5.4.1 Labelling of the TOE (ALC_CMC.1)

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

ALC_CMC.1-1: The evaluator shall check that the TOE provided for evaluation is labelled with its reference.

Findings:	The physical boundary of the TOE includes the models shown in Table 5 of the ST. The software label can be accessed from the Web GUI by navigating to Settings > About menu; and by reviewing the contents of a version file in the CLI. The hardware model is identified on the back of each device.

ALC_CMC.1-2: The evaluator shall check that the TOE references used are consistent.

Findings:	The TOE references are consistently used across all components and documentation.

5.4.2 TOE CM coverage (ALC_CMS.1)

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

ALC_CMS.1-1: The evaluator shall check that the configuration list includes the following set of items:

a) the TOE itself;

b) the evaluation evidence required by the SARs in the ST.

Findings:	The [ST], in section 2.4.1, describes each of the vendor-defined evaluation evidence needed to satisfy the SARs.

ALC_CMS.1-2: The evaluator shall examine the configuration list to determine that it uniquely identifies each configuration item.

Findings:	Each configuration item is uniquely identified by its naming convention. Specifically, the version and/or date of publication along with the title is sufficient to be able to uniquely identify the item. The [ST], in sections 1.2 and 2.4, describes each of the vendor-defined evaluation evidence needed to satisfy the SARs.

5.5 ATE: Tests

5.5.1 Independent Testing – Conformance (ATE_IND.1)

457 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.

458 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.

459 The evaluator should consult Appendix 709 when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

460 Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section A.9.3.1.

461 A.9.3 Evaluator Actions for Testing the TOE

462 A.9.3.1 Tests

463 The evaluator tests the TOE in the minimum configuration as defined in the ST (and the guidance documentation).

464 If the description of the use of extra components in the ST and guidance documentation identifies any difference in the SFRs allocated to a component, or the scope of the SFRs involved (e.g. if different selections apply to different instances of the component) then the evaluator tests these additional SFR cases that were not included in the minimum configuration.

465 In addition, the evaluator tests the following aspects for each extra component that is identified as allowed in the distributed TOE:

466 Communications: the evaluator follows the guidance documentation to confirm, by testing, that any additional connections introduced with the extra component and not present in the minimum configuration are consistent with the requirements stated in the ST (e.g. with regard to protocols and ciphersuites used). An example of such an additional connection would be if a single instance of the component is present in the minimum configuration and adding a duplicate component then introduces an extra communication between the two instances. Another example might be if the use of the additional components necessitated the use of a connection to an external authentication server instead of using locally stored credentials.

Findings:	All SFRs identified in the ST are applicable for each extra component and were tested on each component.

467 Audit: the evaluator confirms that the audit records from different instances of a component can be distinguished so that it is clear which instance generated the record.

Findings:	The evaluator verified that the audit records from different instances of a component can be distinguished so that it is clear which instance generated the record. Each audit record includes the hostname of the device from which it was generated.

468 Management: if the extra component manages other components in the distributed TOE then the evaluator shall follow the guidance documentation to confirm that management via the extra component uses the same roles and role holders for administrators as for the component in the minimum configuration.

Findings:	The evaluator verified that that the extra component uses the same roles for administrators as is specified for the component in the minimum configuration.

6 Vulnerability Assessment

469 **[Modified by TD0547]** 5.6.1.1 Evaluation Activity: 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.

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

Findings:	The evaluator collected this information from the developer which was used to feed into the Type 1 Flaw Hypotheses search (below).

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.

Findings:	The [ST] section 5.3 provides a mapping of the requirements between the distributed TOE components. Each component TOE independently meets the NDcPP requirements with protocol selections being the only difference. A mapping of auditable events are specified in the [ST] section 6.1.1. Each TOE component records the associated audit events. The TOE claims FCO_CPC_EXT.1. The guidance requirements identified in 3.5.1.2 of the [SD] have been addressed in Section 3.3.1.2 of this document.

471 5.6.1.2 Evaluation Activity: The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

Findings:	The following sources of public vulnerabilities were considered in formulating the specific list of flaws to be investigated by the evaluators, as well as to reference in directing the evaluators to perform key-word searches during the evaluation of the TOE. Hypothesis sources for public vulnerabilities were:

a) NIST National Vulnerabilities Database (can be used to access CVE and US-CERT databases identified below): https://web.nvd.nist.gov/view/vuln/search

b) Common Vulnerabilities and Exposures: https://cve.mitre.org/cve/ https://www.cvedetails.com/vulnerability-search.php

c) US-CERT: https://www.kb.cert.org/vuls/html/search

d) CCS – Alerts and advisories: https://cyber.gc.ca/en/alerts-advisories

e) Google

Type 1 Hypothesis searches were conducted last on April 3, 2024. Including search terms for components, and third party programs and libraries:

- NETSCOUT nGeniusONE 6.3.3
- NETSCOUT InfiniStreamNG 6.3.3
- NETSCOUT 1401J
- NETSCOUT 2401J
- NETSCOUT 2695J
- NETSCOUT 4795J
- NETSCOUT 4895J
- NETSCOUT 6695J
- NETSCOUT 9795J
- NETSCOUT 9802J
- NETSCOUT 9807J
- NETSCOUT 9895J
- NETSCOUT 690J
- Intel® Xeon® Gold 6142
- Intel® Xeon® Silver 4110
- Intel® Xeon® Gold 6126
- Intel® Xeon® Gold 6152
- Intel Atom® Processor C3955
- Dell OpenManage Server Administrator
- OpenSSH
- Apache HTTP Server 2.4
- OpenSSL 1.0.2

- CentOS 7

- Java 11 JSSE

- Bouncy Castle

- bc-fja

- Dom4j

- iDRAC

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

Type-2 hypotheses identified for the NDcPP are found in the vulnerability assessment.

The evaluation team developed Type 3 flaw hypotheses in the vulnerability assessment.

The evaluation team conducted fuzzing in accordance with Type 4 flaw hypothesis testing.

No residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.