www.GossamerSec.com

Assurance Activity Report for CoreLight Sensors with BROLIN V28

Version 0.3
12/20/24

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

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

Document: AAR- VID11489

Revision History

Revision	Date	Authors	Summary
Version 0.1	11/12/24	John Messiha	Initial draft
Version 0.2	12/5/2024	John Messiha	Address ECR comments
Version 0.3	12/20/2024	John Messiha	Address ECR comments

The TOE Evaluation was Sponsored by:

Corelight, Inc.
548 Market St, PMB 77799
San Francisco, CA 94104-5401

Evaluation Personnel:

- John Messiha

Common Criteria Versions:

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

Common Evaluation Methodology Versions:

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

Table of Contents

1. Introduction

This document presents evaluations results of the Corelight Sensors with Brolin v28 NDcPP22e evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 CAVP CERTIFICATES

The TOE provides the cryptography to support all security functions. All algorithms claimed have Cryptographic Algorithm Validation Program (CAVP) for the models specified in the table below.

	AP200	AP520	AP1001	AP1100	AP1200	AP3000	AP3100	AP3200	AP5000	AP5002	AP5200
CPU	Intel Xeon Scalable Silver 4110	Intel Xeon Scalable Gold 5317	Intel Xeon Scalable Silver 4116	Intel Xeon Scalable Silver 4314	AMD EPYC 9254	Intel Xeon Scalable Gold 6238	Intel Xeon Scalable Gold 5318Y	AMD EPYC 9534	AMD EPYC 7742	AMD EPYC 7713	AMD EPYC 9754
Micro-architecture	Skylake-SP	Icelake-SP	Skylake-SP	Icelake-SP	Genoa/Zen 4	Cascade Lake-SP	Icelake-SP	Genoa/Zen 4	Rome/Zen2	Milan/Zen3	Bergamo/Zen 4c
NIC vendor	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel
Image	Common	Common	Common	Common	Common	Common	Common	Common	Common	Common	Common
Tested?											
CAVP	Yes	Equivalent	Equivalent	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Equivalent
NDcPP	Yes	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Yes	Equivalent

The table below identifies the algorithm certificates supported by the Corelight Cryptographic Module, Corelight Cryptographic Module KAS, and Corelight RSA Engine all version 2.1.2.

Requirements	Functions	CAVP Cert
	Cryptographic key generation	
NDcPP22e:FCS_CKM.1	RSA schemes using key sizes of 2048 and 3072	A5867
NDcPP22e:FCS_CKM.1	ECC schemes using 'NIST curves' P-256, P-384, P-521 (keyge/keyver)	A5859
NDcPP22e:FCS_CKM.1	FFC schemes using 'safe prime' group DH14 (2048), DH16 (4096)	Tested with known, good implementation
	Cryptographic key establishment	
NDcPP22e:FCS_CKM.2	RSA schemes using key sizes of 2048 and 3072	Tested with known, good implementation
NDcPP22e:FCS_CKM.2	Diffie-Hellman schemes using safe primes	Tested with known, good implementation

Requirements	Functions	CAVP Cert
NDcPP22e:FCS_CKM.2	Elliptic curve-based key establishment schemes: P-256, P-384. P-521	A5862
	Encryption/Decryption	
NDcPP22e:FCS_COP.1/DataEncryption	AES CBC, CTR, GCM (128, 256 bits)	A5859
	Cryptographic hashing	
NDcPP22e:FCS_COP.1/Hash	SHA-1/256/384/512 (digest sizes 160, 256, 384 and 512 bits)	A5859
	Keyed-hash message authentication	
NDcPP22e:FCS_COP.1/KeyedHash	HMAC-SHA-1/256/384/512 (key and output MAC size 160, 256, 384, 512)	A5859
	Cryptographic signature services	
NDcPP22e:FCS_COP.1/SigGen	RSA schemes using key sizes of 2048 and 3072 (siggen/sigver)	A5859
NDcPP22e:FCS_COP.1/SigGen	Elliptic Curve Digital Signature Algorithm (ECDSA) with an elliptical curve P-256, P-384. P-521 (siggen/sigver)	A5859
	Random bit generation	
NDcPP22e:FCS_RBG_EXT.1	CTR_DRBG (AES) with SW based noise source (256 bits)	A5859

Table 1 Cryptographic Algorithm Certificates

Expanding on the table above, the following platforms have been CAVP tested, with the highlighted models also having been fully CC tested:

- AP200 - Intel Xeon Scalable Silver 4110 Skylake-SP
- AP1100 - Intel Xeon Scalable Silver 4314 Sunny Cove/Icelake-SP
- AP1200 - AMD EPYC 9254 Genoa/Zen 4
- AP3000 - Intel Xeon Scalable Gold 6238 Skylake/Cascade Lake-SP
- AP3100 - Intel Xeon Scalable Gold 5318Y Sunny Cove/Icelake-SP
- AP3200 - AMD EPYC 9354 Genoa/Zen4
- AP5000 - AMD EPYC 7742 Rome/Zen2
- AP5002 - AMD EPYC 7713 Milan/Zen3

The following models are equivalent and thus covered by the CAVP testing:

- AP520 - Intel Xeon Scalable Gold 5317 Sunny Cove/Icelake-SP

- Equivalent to the AP3100 as its CPU is in the same series with the same microarchitecture
- AP1001 - Intel Xeon Scalable Silver 4116 Skylake-SP
  - Equivalent to the AP200 as its CPU is in the same series with the same microarchitecture
- AP5200 - AMD EPYC 9754 Bergamo/Zen 4c
  - Equivalent to the AP3200 as the AMD Zen4 and Zen4c microarchitectures have the same instruction set. The difference in the microarchitectures is Zen4c has half the cache of zen4.

1.2 EVALUATED PLATFORM EQUIVALENCE

The TOE is Corelight Sensors with BroLin v28. The TOE includes several models as shown below:

The evaluation includes eleven different hardware platforms, with some tested and the remainder claimed as equivalent. The remainder of this section provides the equivalency argument between the tested and equivalent hardware.

The below includes the full list of models and summarizes which devices have been tested for CC and CAVP.

	AP200	AP520	AP1001	AP1100	AP1200	AP3000	AP3100	AP3200	AP5000	AP5002	AP5200
CPU	Intel Xeon Scalable Silver 4110	Intel Xeon Scalable Gold 5317	Intel Xeon Scalable Silver 4116	Intel Xeon Scalable Silver 4314	AMD EPYC 9254	Intel Xeon Scalable Gold 6238	Intel Xeon Scalable Gold 5318Y	AMD EPYC 9534	AMD EPYC 7742	AMD EPYC 7713	AMD EPYC 9754
Micro-architecture	Skylake-SP	Icelake-SP	Skylake-SP	Icelake-SP	Genoa/Zen 4	Cascade Lake-SP	Icelake-SP	Genoa/Zen 4	Rome/Zen2	Milan/Zen3	Bergamo/Zen 4c
NIC vendor	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel
Image	Common	Common	Common	Common	Common	Common	Common	Common	Common	Common	Common
Tested?											
CAVP	Yes	Equivalent	Equivalent	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Equivalent
NDcPP	Yes	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Equivalent	Yes	Equivalent

Section 1.1 further expands upon the CAVP equivalency. The rest of this section is intended to expand upon the CC testing equivalency.

Each Corelight Sensor pND model leverages the same image, which runs BroLin v28 as the underlying OS. All security functionality is built into the software image. While the software image is the same, the image will install different code paths depending on the type of CPU. They are different code paths for Intel vs AMD and therefore both a Intel (AP200) and an AMD (AP5002) model were tested. The code path does not differ between microarchitectures of the same CPU type, for example Intel Skylake vs Intel Icelake. The CAVP testing described in Section 1.1 above, further demonstrates that the different microarchitectures do not impact the successful implementation of any of the crypto functionality.

Beyond the CPU, there are other hardware differences, however they differ primarily in physical form factor, number and types of connections and slots, and relative performance. These differing characteristics primarily affect only non-TSF relevant functionality (such as throughput, processing speed, number and type of network connections supported, number of concurrent connections supported, and amount of storage). Again, all TOE security functions are implemented in software. Additionally, all 11 models leverage intel NICs, with the necessary drives included in the TOE image.

To summarize, the evaluation team ran the entire test suite on each of the following devices to cover all possible code paths:

- AP200 (Intel)
- AP5002 (AMD)

1.3 REFERENCES

The following evidence was used to complete the Assurance Activities:

• Corelight Sensors with BroLin v28 Security Target, Version 0.6, December 20, 2024 (ST).

• Corelight Sensor AP 200, AP 520, AP 1001, AP 1100, AP 1200, AP 3000, AP 3100, AP 3200, AP 5000, AP 5002 & AP 5200 Common Criteria Guidance Document, Version 0.2, December 20, 2024 (Admin Guide).

2. Protection Profile SFR Assurance Activities

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

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 Security Audit (NDcPP22e:FAU_GEN.1 | NDcPP22e:FAU_GEN.2) of the ST states the TOE generates a comprehensive set of audit logs that identify specific TOE operations whenever an auditable event occurs. This covers all events identified in the audit table for the SFR. Each audit record contains the date and time of the event, type of event, subject identity, and the outcome (success or failure) of the event.

All configuration changes are recorded with subject identity as the user request is made through the command line interface (CLI) with either local or remote connection as well as through the TOE's Web UI. Administrative tasks of generating and deleting cryptographic keys contain the necessary audit information as the key name is used to identify the key.

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

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

The section entitled "Sample Audit Events" in the Admin Guide contains samples of audit records associated with each auditable event identified by the Security Target.

This information includes details about the audit records which the TOE generates including details encompassing the required content. During testing, the evaluator mapped the entries in the tables in this section to the TOE generated events.

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

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

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

cases but includes also events about successful actions like successful build up/tear down of a secure communication channel between TOE components.

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

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

2.1.2 User identity association (NDcPP22e:FAU_GEN.2)

2.1.2.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See NDcPP22e:FAU_GEN.1.

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

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

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

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

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

Not applicable as the TOE is not distributed.

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

2.1.3.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.

The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally; what happens when the local audit data store is full; and how these records are protected against unauthorized access.

The evaluator shall examine the TSS to ensure it describes whether the TOE is a standalone TOE that stores audit data locally or a distributed TOE that stores audit data locally on each TOE component or a distributed TOE that contains TOE components that cannot store audit data locally on themselves but need to transfer audit data to other TOE components that can store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs it contains a list of TOE components that store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs that contain components which do not store audit data locally but transmit

their generated audit data to other components it contains a mapping between the transmitting and storing TOE components.

The evaluator shall examine the TSS to ensure that it details the behavior of the TOE when the storage space for audit data is full. When the option 'overwrite previous audit record' is selected this description should include an outline of the rule for overwriting audit data. If 'other actions' are chosen such as sending the new audit data to an external IT entity, then the related behaviour of the TOE shall also be detailed in the TSS.

The evaluator shall examine the TSS to ensure that it details whether the transmission of audit information to an external IT entity can be done in real-time or periodically. In case the TOE does not perform transmission in real-time the evaluator needs to verify that the TSS provides details about what event stimulates the transmission to be made as well as the possible acceptable frequency for the transfer of audit data.

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

For distributed TOEs the evaluator shall examine the TSS to ensure it describes which TOE components are storing audit information locally and which components are buffering audit information and forwarding the information to another TOE component for local storage. For every component the TSS shall describe the behaviour when local storage space or buffer space is exhausted.

Section 6.1 Security audit (NDcPP22e:FAU_STG_EXT.1) of the ST states The TOE can be configured to export audit events securely to an audit server using FTP within the SSH v2 protocol. The audit server in this case is the SFTP server.

The TOE is a standalone TOE that stores audit data locally. The TOE stores up to 100,000 audit records locally. When the local data is full, the oldest audit events are overwritten to allow new audit events to be created. The TOE is designed to store 100K records in the database. API queries however are limited to 7 days. Security Administrators can access the audit events and can clear the audit events. This way, audit events are protected against unauthorized access.

The TOE transmits audit data to an external audit server periodically (every two minutes) in batches. If there is an SSH connection failure, the TOE will continue to store local audit events on the TOE and will transmit any locally stored contents when connectivity to the audit server is restored.

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

The evaluator shall also examine the guidance documentation to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit

event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and 'cleared' periodically by sending the data to the audit server.

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

The sections entitled "Audit server requirements for audit log export" and "Audit Log Export" in the Admin Guide details the configuration of protected syslog via a trusted channel. The TOE supports audit forwarding over SSH. The same section also details local log storage. The Corelight appliance is designed to have capacity for 100,000 audit records. This capacity should be sufficient to store multiple days' worth of auditing data, even with a large number of events such as those generated during the appliance reconfigurations. Once this limit is reached, oldest records are removed in order to permit new records to be added. Administrators can access the audit events locally, however while there is room to store months of data on a typical system, API queries cannot back further than 7 days. In order to remain compliant with the certification all audit data must be exported to a secure external audit server in order to protect from loss of this information.

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

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

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

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

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

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

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

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

Test 1: The successful establishing of the SSH syslog connection is demonstrated in FTP_ITC.1. In each case, the TOE initiated the connection without administrator intervention. The use of SSH ensured no audits were viewed in cleartext. The audits collected as part of FAU_GEN.1 throughout testing were gathered from the remote syslog server running OpenSSH_8.9 thus demonstrating that audits were successfully received by the remote syslog server.

Test 2: The evaluator also continued to generate audit data until the local storage space was exceeded. The evaluator verified that when the local audit storage was filled to the maximum, the existing audit data was rotated and archived.

Test 3: Not applicable. The TOE does not claim support for FAU_STG_EXT.2/LocSpace.

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

2.2 Cryptographic support (FCS)

2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.2.1.1 NDcPP22e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Section 6.2 Cryptographic support (NDcPP22e:FCS_CKM.1) of the ST states The TOE supports RSA key sizes of 2048 bits and 3072 bits, for key generation conforming to Cryptographic key generation conforming to FIPS PUB 186-4 Digital Signature Standard (DSS), Appendix B.3. The RSA keys are used in support of SSH and TLS communications.

The TOE supports Elliptical NIST curve sizes of P-256, P-384 and P-521 conforming to Cryptographic key generation conforming to FIPS PUB 186-4 Digital Signature Standard (DSS)", Appendix B.4. The Elliptic keys are used in support of ECDH key exchange for SSH and TLS. The TOE supports FFC Schemes using 'safe-prime' groups that meet the following: "NIST Special Publication 800-56A Revision 3, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" and RFC 3526. The TOE supports DH14 and DH16(server only) key generation in support of DH key exchanges as part of SSH.

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

The section entitled "SFTP Authentication" in the Admin Guide states that the Corelight appliance does not allow the administrator to provide a private SSH key for authentication with the remote SFTP server. This means an administrator cannot make choices about the cryptographic algorithms used to generate the key, as well as parameters, such as the size of the modulus with an RSA key, for instance. The appliance will automatically generate an RSA key, and the administrator will be required to retrieve the key from the sensor, which they will then have to authorize on the SFTP server. The SFTP batch exporter only supports public key exchange for authentication, thus it is necessary for the SFTP server to support public key authentication.

The section entitled "CSR generation" in the Admin Guide details how the administrator can generate a CSR and import the signed certificate to the appliance to be used as the web server certificate. This section also states the allowed key sizes when generating a CSR.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Component Testing Assurance Activities: Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public

verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d.

Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

a) Random Primes:

- Provable primes

- Probable primes

b) Primes with Conditions:

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

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

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly

produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

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

- Primes q and p shall both be provable primes

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

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

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

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

- len(q) + 64 bit output of RBG, followed by a mod q-1 operation and a +1 operation, where 1<= x<=q-1.

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

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

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

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

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

(TD0580 applied)

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

The evaluator verified the correctness of the TSF's implementation of key generation using FFC safe-prime groups and RSA scheme through the testing of FCS_SSHS_EXT.1 and FTP_TRP.1/Admin. Establishing an SSH session uses these schemes and demonstrates correctness.

2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.2.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. It is sufficient to provide the scheme, SFR, and service in the TSS.

The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be:

Scheme | SFR | Service

RSA | FCS_TLSS_EXT.1 | Administration

ECDH | FCS_SSHC_EXT.1 | Audit Server

ECDH | FCS_IPSEC_EXT.1 | Authentication Server

The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

(TD0580 applied)

Section 6.2 Cryptographic support (NDcPP22e:FCS_CKM.2) of the ST states The TOE supports Cryptographic Key Establishment using the following schemes:

- RSA key based establishment conforming to RSAES-PKCS1-v1_5 as specified in Section 7.2 of RFC 3447, “Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. The TOE implements RSA key establishment scheme with key sizes of 2048 and 3072 bits
- Elliptical curve-based establishment conforming to NIST Special Publication 800-56A Revision 2, Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography.
- FFC Schemes using “safe-prime” groups that meet the following: ‘NIST Special Publication 800-56A Revision 3, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography” and groups listed in RFC 3526.

The TOE uses RSA and ECC schemes during TLS and uses FFC safe primes and ECC schemes during SSH.

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	WebUI for remote administration
ECC	FCS_SSHC_EXT.1	Audit log secure export
	FCS_SSHS_EXT.1	CLI remove administration
	FCS_TLSS_EXT.1	WebUI for remote administration
FFC/safe primes	FCS_SSHC_EXT.1	Audit log secure export
	FCS_SSHS_EXT.1	CLI remove administration

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

The sections entitled "Key-based Authentication with SSH" and “SFTP Authentication” in the Admin Guide lists the SSH key exchange methods that are allowed in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Component Testing Assurance Activities: Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

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

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses RSAES-PKCS1-v1_5.

FFC Schemes using 'safe-prime' groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

(TD0580 applied)

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

The evaluator verified the correctness of the TSF's implementation of key establishment using FFC safe-prime groups and RSA scheme through the testing of FCS_SSHS_EXT.1 and FTP_TRP.1/Admin. Establishing an SSH session uses these schemes and demonstrates correctness.

2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.3.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction

method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for2). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

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

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

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

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

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_CKM.4) states the TOE satisfies all requirements as specified in FCS_CKM.4 of NDcPPv2.2e for destruction of keys and CSPs and includes the following cryptographic keys

Key	Purpose	Storage (RAM/Flash)	Encrypted/Plaintext	Destruction and when
TLS session keys	Keys exchanged for protecting the confidentiality of the remote administration session	RAM	Plaintext	Overwritten w/ zeros after use

TLS auth keys	X509 certificate used for authentication as TLS server	Flash	Plaintext	Stored persistently in the TOE's data partition, and overwritten w/ zeros upon reset to factory defaults
SSH session keys	Keys exchanged for protecting the confidentiality of the SSH sessions	RAM	Plaintext	Overwritten w/ zeros after use
SSH host keys	The host key for authentication of the TOE as an SSH server. As an SSH client, the host key of the SSH server the TOE is connecting too.	Flash	Plaintext	Stored persistently in the TOE's data partition, and overwritten w/ zeros prior to generating new ones or upon reset to factory defaults
SSH public keys	The public key for authentication as an SSH client. As an SSH server, the public key of the client authenticating to the TOE.	Flash	Plaintext	Stored persistently in the TOE's data partition, and overwritten w/ zeros prior to generating new ones or upon reset to factory defaults

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

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

The TOE does not have conditions that involve delayed key destruction.

Component Testing Assurance Activities: None Defined

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

2.2.4.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_COP.1/DataEncryption) of the ST states The TOE supports AES encryption and decryption conforming to CBC as specified in ISO 10116, CTR as specified in ISO 10116 and GCM as specified in ISO 19772.

The AES key size supported are 128 bits and 256 bits and the AES modes supported are: CBC, CTR and GCM.

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

The section entitled "SFTP Authentication" in the Admin Guide states that an administrator cannot make choices about the cryptographic algorithms used for the SFTP export function. The algorithms supported by the SFTP exporter are hard coded and they align with the selections in the Security Target.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

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

KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key.

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

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

PT = IV

else:

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

PT = CT[i-1]

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

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

AES-GCM Test

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

128 bit and 256 bit keys

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

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

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

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-CTR Known Answer Tests

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Due to the fact that Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

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

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

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

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

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with

a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384 ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128].

AES-CTR Multi-Block Message Test

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

AES-CTR Monte-Carlo Test

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

Input: PT, Key

for i = 1 to 1000:

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

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

There is no need to test the decryption engine.

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

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

2.2.5.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_COP.1/Hash) of the ST states The TOE supports Cryptographic hashing services conforming to ISO/IEC 10118-3:2004. The hashing algorithms are used in SSH and TLS connections for secure communications.

The following hashing algorithms are supported: SHA-1, SHA-256, SHA-384 and SHA-512.

The message digest sizes supported are: 160, 256, 384 and 512 bits.

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

The section entitled "SFTP Authentication" in the Admin Guide lists the SSH ciphers, key exchange methods, authentication algorithms, and MAC algorithms allowed by the TOE in the evaluated configuration when acting as the SSH Client to export logs to an SFTP server.

The section entitled “Key-based Authentication with SSH” in the Admin Guide lists the host key algorithms, encryption ciphers/algorithms, key exchange methods, and MAC algorithms supported by the TOE in the evaluated configuration when acting as the SSH Server to offer the SSH interface.

The section entitled " Web UI Connections" in the Admin Guide states that the TOE uses TLS protocol version 1.2 and it lists the supported ciphersuites allowed in the evaluated configuration when acting as the TLS Server to offer the Web UI interface. The ciphersuites indicate the supported hash size in its name, SHA-1 represented by “SHA”, SHA-256 represented by “SHA256”, and SHA-384 represented by “SHA384”.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

No additional configuration is needed.

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

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

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

2.2.6.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_COP.1/KeyedHash) of the ST states The TOE supports Keyed-hash message authentication conforming to ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”. HMAC algorithms are used in support of SSH and TLS sessions.

HMAC algorithm	Hash function	Block size	Key length	MAC length
HMAC-SHA-1	SHA-1	512-bits	160 bits	160 bits
HMAC-SHA-256	SHA-256	512 bits	256 bits	256 bits
HMAC-SHA-384	SHA-384	1024 bits	384 bits	384 bits
HMAC-SHA-512	SHA-512	1024 bits	512 bits	512 bits

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the values used by the HMAC function: key length, hash function used, block size, and output MAC length used defined in the Security Target supported by the TOE for keyed hash function.

The section entitled "SFTP Authentication" in the Admin Guide lists the SSH ciphers, key exchange methods, authentication algorithms, and MAC algorithms allowed by the TOE in the evaluated configuration when acting as the SSH Client to export logs to an SFTP server.

The section entitled “Key-based Authentication with SSH” in the Admin Guide lists the host key algorithms, encryption ciphers/algorithms, key exchange methods, and MAC algorithms supported by the TOE in the evaluated configuration when acting as the SSH Server to offer the SSH interface.

The section entitled " Web UI Connections" in the Admin Guide states that the TOE uses TLS protocol version 1.2 and it lists the supported ciphersuites allowed in the evaluated configuration when acting as the TLS Server to offer the Web UI interface. The ciphersuites indicate the supported keyed hash size in its name, HMAC-SHA-1 represented by “SHA”, HMAC-SHA-256 represented by “SHA256”, and HMAC-SHA-384 represented by “SHA384”. Note this only applies to ciphersuites with “CBC” in the name as GCM ciphersuites do not use HMAC algorithms.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the

system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

No additional configuration is needed.

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

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

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

2.2.7.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_COP.1/SigGen) of the ST states The TOE provides Cryptographic signature generation and verification in accordance with the following cryptographic algorithms:

- RSA digital signature conforming to FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 5.5, using PKCS #1 v2.1 Signature Schemes RSASSA-PSS and/or RSASSA-PKCS1v1_5; ISO/IEC 9796-2, Digital signature scheme 2 or Digital Signature scheme 3.
    - The RSA key sizes supported are: 2048 and 3072 bits.
- The TOE uses Elliptical curve digital signature algorithm conforming to FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 6 and Appendix D, Implementing “NIST curves” [P-256, P-384, P-521]; ISO/IEC 14888-3, Section 6.4
    - The Elliptical curve key size supported is 256, 384, and 521 bits.

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

The section entitled "CSR generation" in the Admin Guide describes generating a 2048 RSA private key file. The sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate

in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

No additional configuration is needed.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

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

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

Signature Verification Test

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

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

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

2.2.8 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.2.8.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.8.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.8.3 NDcPP22e:FCS_HTTPS_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_HTTPS_EXT.1) of the ST states the TOE adheres to RFC 2818 by acting as a server and hosting the Admin Web UI on port 443. The TOE provides a server identity certificate to connecting clients (either a self-signed certificate, or one that the admin can load or issue against a TOE generated CSR).

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

The section entitled " Web UI Connections" in the Admin Guide states that the TOE offers a Web GUI that is protected with HTTPS where the TOE acts as the HTTPS server and no additional configuration is needed from the user. This section also explains the procedure to access the TOE's Web GUI using a web browser.

Component Testing Assurance Activities: This test is now performed as part of FIA_X509_EXT.1/Rev testing.

Tests are performed in conjunction with the TLS evaluation activities.

If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

The TOE's HTTPS channel does not support mutual authentication and therefore FIA_X509_EXT.1 is not applicable to HTTPS in this evaluation. The testing of the HTTPS channel was demonstrated in FCS_TLSS_EXT.1.

2.2.9 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.9.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.9.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_RBG_EXT.1) of the ST states The TOE uses CTR_DRBG conforming to ISO/IEC 18031:2011. An entropy document was submitted and approved.

The CTR_DRBG is seeded by an entropy source that accumulates entropy from software-based noise source with a minimum of 256 bits of entropy.

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

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

The TOE does not offer any configuration for the RNG functionality.

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

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

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

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

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

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

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

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

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

2.2.10 SSH Client Protocol - per TD0636 (NDcPP22e:FCS_SSHC_EXT.1)

2.2.10.1 NDcPP22e:FCS_SSHC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.2 NDcPP22e:FCS_SSHC_EXT.1.2

TSS Assurance Activities: 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.

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.

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHC_EXT.1.2) of the ST states the TOE supports authenticating itself to a remote endpoint with a user-based public key.

The following public key algorithms are supported: rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384 and ecdsa-sha2-nistp521.

These claims are consistent with FCS_CKM.1, FCS_COP.1/Hash, and FCS_COP.1/SigGen.

Password-based authentication is not supported.

Guidance Assurance Activities: 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.

The section entitled "SFTP Authentication" in the Admin Guide lists the SSH ciphers, key exchange methods, authentication algorithms, and MAC algorithms allowed by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the

system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

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

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.

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 1: The evaluator attempted to connect the TOE to an SSH server using a pubkey. The TOE claims support for ECDSA public keys. This supported public key algorithm resulted in a successful connection. All the claimed public key algorithms are demonstrated in FCS_SSHC_EXT.1.5 Test 1.

Test 2: Not applicable as password-based authentication is not supported.

2.2.10.3 NDcPP22e:FCS_SSHC_EXT.1.3

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHC_EXT.1.3) of the ST states the TOE accepts packet size up to 262144 bytes and meets the requirements of RFC 4253. Any packet greater than the max size will be dropped and the connection will be terminated.

Guidance Assurance Activities: None Defined

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

The evaluator created and sent a packet to the TOE that was larger than the maximum packet size of 262144 bytes. The evaluator observed that the TOE rejected the packet and the connection was closed.

2.2.10.4 NDcPP22e:FCS_SSHC_EXT.1.4

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

well. The evaluator shall check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHC_EXT.1.4) of the ST states The TOE supports the following encryption algorithms: aes128-cbc, aes256-cbc, aes128-ctr, aes256-ctr for SSH transport. There are no optional characteristics specified for FCS_SSHC_EXT.1.4. The encryption algorithms specified are identical to those claimed for the SFR.

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

The section entitled "SFTP Authentication" in the Admin Guide lists the SSH ciphers, key exchange methods, authentication algorithms, and MAC algorithms allowed by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

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

The evaluator attempted to establish an SSH connection with each of the following SSH algorithms to encrypt the session: aes128-cbc, aes256-cbc, aes128-ctr, and aes256-ctr. The evaluator captured packets associated with each of the connection attempts and observed that these algorithms resulted in a successful connection to the SSH test server.

2.2.10.5 NDcPP22e:FCS_SSHC_EXT.1.5

TSS Assurance Activities: 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.

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

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.

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHC_EXT.1.5) of the ST states the following are the public key algorithms supported: rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384 and ecdsa-sha2-nistp521. There are no optional characteristics specified for FCS_SSHC_EXT.1.5. The public key algorithms specified are identical to those claimed for the SFR.

When configuring the SFTP channel for the audit server, the administrator defines the expected server host key.

x509v3-based public key authentication algorithms are not claimed

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

The section entitled "SFTP Authentication" in the Admin Guide lists the SSH ciphers, key exchange methods, authentication algorithms, and MAC algorithms allowed by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Testing Assurance Activities: 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 is therefore intended to 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.

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.

Test 1: The evaluator attempted to connect the TOE to the SSH server alternately using each of the authentication algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator confirmed that the TOE supports these algorithms: rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, and ecdsa-sha2-nistp521 as claimed.

Test 2: The evaluator followed up with a disallowed authentication algorithm and confirmed that it was rejected.

2.2.10.6 NDcPP22e:FCS_SSHC_EXT.1.6

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHC_EXT.1.6) of the ST states the TOE supports the following data integrity MAC algorithms: hmac-sha1, hmac-sha2-256, and hmac-sha2-512. The data integrity algorithms specified are identical to those claimed for the SFR.

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

The section entitled "SFTP Authentication" in the Admin Guide lists the SSH ciphers, key exchange methods, authentication algorithms, and MAC algorithms allowed by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Testing Assurance Activities: Test 1 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall establish an SSH connection using each of the algorithms, except 'implicit', specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 2 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall configure an SSH 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.

Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 1: The evaluator attempted to connect the TOE to an SSH server alternately using each of the MAC algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator confirmed that the TOE supports these algorithms hmac-sha2-256, and hmac-sha2-512 as claimed.

Test 2: The evaluator followed up with a disallowed MAC algorithm to ensure it was rejected. The TOE rejected a connection attempt using this disallowed MAC algorithm.

2.2.10.7 NDcPP22e:FCS_SSHC_EXT.1.7

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHC_EXT.1.7) of the ST states the TOE supports the following key exchange algorithms: diffie-hellman-group14-sha1 and ecdh-sha2-nistp256. The key exchange algorithms specified are identical to those claimed for the SFR.

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

The section entitled "SFTP Authentication" in the Admin Guide lists the SSH ciphers, key exchange methods, authentication algorithms, and MAC algorithms allowed by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Testing Assurance Activities: 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.

Test 1: The evaluator attempted to connect the TOE to an a SSH server using each of the key exchange algorithms that can be claimed to determine which ciphers are supported with successful connections. The algorithms used

were diffie-hellman-group14-sha1, and ecdh-sha2-nistp256. The evaluator confirmed that the connections were successful in each case.

2.2.10.8 NDcPP22e:FCS_SSHC_EXT.1.8

TSS Assurance Activities: 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.

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHC_EXT.1.8) of the ST states The TOE is capable of rekeying. The TOE verifies the following thresholds:

- No longer than one hour
- No more than one gigabyte of transmitted data

The TOE continuously checks both conditions. When either of the conditions is met, the TOE will initiate a rekey.

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

The SSH session rekey thresholds are not configurable. They are hardcoded to rekey after one hour or following the exchange of 1 GB of data, whichever comes first.

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

For testing of the time-based threshold the evaluator shall use 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).

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.

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).

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

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

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

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

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

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

For the 1-hour time rekey test, this test is practically not testable given the TOE's SSH Client implementation. The TOE uses its SSH Client implementation (as the SFTP logs export function) to export logs to a remote SSH server. And, by design, the SFTP exporter closes the SSH connection as soon as the log export is complete; it does not keep the SSH connection open.

The security administrator can specify the intervals of how often the log export function runs (eg. every 3 minutes, every 5 minutes) but that does not change the fact that the SSH connection is closed as soon as the logs export is complete.

For the data rekey test, the vendor provided a special build where the TOE has been configured with a lower data rekey threshold (1 KB instead of the hard-coded value of 1 GB). The evaluator then attempted to connect the TOE to an SSH server generating enough data and confirmed via logs and packet captures that the TOE initiated a rekey as expected before the data rekey threshold was reached.

2.2.10.9 NDcPP22e:FCS_SSHC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: 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.

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.

Test 1: The evaluator attempted to connect the TOE to an SSH server using a host key not recognized by the TOE. The evaluator found that this connection was rejected.

Test 2: The evaluator then configured the correct server host key on the TOE, re-attempted the connection and verified that the connection was accepted by the TOE. The evaluator then kept the correct server host key configured on the TOE and changed the server host key being presented by the SSH server and verified that the TOE also rejected the connection.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.11.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 NDcPP22e:FCS_SSHS_EXT.1.2

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

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

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHS_EXT.1.2) of the ST states the TOE supports public key authentication and password-based authentication.

The following public key algorithms: ssh-rsa, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521. This is consistent with the claims in FCS_COP.1/SigGen.

The TOE verifies the presented public key matches the one configured for the user by the administrator.

Section 6.3 Identification and authentication of the ST states that for remote SSH administration, the TOE supports RSA and ECDSA public key authentication. If the user uses public key-based authentication and it is successful, then the user is granted access to the TOE. If the user enters invalid user credentials, they will not be granted access and will be presented the login page. Username and password are also supported. If the username and password match the expected value, the administrator will be granted access, or if the values do not match, the administrator will be denied. In the case of a failed login attempt, the account lockout counter will be incremented.

Guidance Assurance Activities: None Defined

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

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

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

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

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

(TD0631 applied)

Test 1: The evaluator configured a remote SSH client to present a public key corresponding to each supported client public-key authentication algorithm supported by the TOE. The evaluator then established sufficient separate SSH connections using each key and verified that the SSH connection was successful in each time.

Test 2: The evaluator demonstrated an unsuccessful login using an unrecognized public key.

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

Test 4: This was performed as part of Test 3.

2.2.11.3 NDcPP22e:FCS_SSHS_EXT.1.3

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHS_EXT.1.3) of the ST states the TOE accepts packet size up to 262144 bytes and meets the requirements of RFC 4253. Any packet greater than the max size will be dropped and the connection will be terminated.

Guidance Assurance Activities: None Defined

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

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

2.2.11.4 NDcPP22e:FCS_SSHS_EXT.1.4

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHS_EXT.1.4) of the ST states the TOE supports the following encryption algorithms: aes128-ctr, aes256-ctr for SSH transport. There are no optional characteristics specified for FCS_SSHS_EXT.1.4. The encryption algorithms specified are identical to those claimed for the SFR.

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

The section entitled “Key-based Authentication with SSH” in the Admin Guide lists the host key algorithms, encryption ciphers/algorithms, key exchange methods, and MAC algorithms supported by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Testing Assurance Activities: The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall start session establishment for an SSH connection from a remote client (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

The evaluator attempted to connect to the TOE using a SSH client alternately using each of the ciphers that can be claimed to verify they are supported with successful connections. In each case the evaluator viewed the Server: Key Exchange Init packet and saw that no additional ciphers beyond those that were claimed were seen as supported.

2.2.11.5 NDcPP22e:FCS_SSHS_EXT.1.5

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHS_EXT.1.5) of the ST states the following are the public key algorithms supported: rsa-sha2-256, rsa-sha2-512, and ecdsa-sha2-nistp256. There are no optional characteristics specified for FCS_SSHS_EXT.1.5. The public key algorithms specified are identical to those claimed for the SFR.

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

The section entitled “Key-based Authentication with SSH” in the Admin Guide lists the host key algorithms, encryption ciphers/algorithms, key exchange methods, and MAC algorithms supported by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

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

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

Has effectively been moved to FCS_SSHS_EXT.1.2.

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

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

(TD0631 applied)

Test 1: The evaluator established an SSH connection with the TOE using each claimed host key algorithm. The connection was successful.

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

2.2.11.6 NDcPP22e:FCS_SSHS_EXT.1.6

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHS_EXT.1.6) of the ST states the TOE supports the following data integrity MAC algorithms: hmac-sha2-256, and hmac-sha2-512. The data integrity algorithms specified are identical to those claimed for the SFR.

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

The section entitled “Key-based Authentication with SSH” in the Admin Guide lists the host key algorithms, encryption ciphers/algorithms, key exchange methods, and MAC algorithms supported by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Testing Assurance Activities: Test 1 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall establish an SSH connection using each of the algorithms, except 'implicit', specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

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

Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 1: The evaluator established an SSH connection with the TOE using each of the claimed integrity algorithms. The evaluator observed a successful connection using each claimed integrity algorithm.

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

2.2.11.7 NDcPP22e:FCS_SSHS_EXT.1.7

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHS_EXT.1.7) of the ST states The TOE supports the following key exchange algorithms: diffie-hellman-group14-sha256, diffie-hellman-group16-sha512, ecdh-sha2-nistp256, ecdh-sha2-nistp384, and ecdh-sha2-nistp521. The key exchange algorithms specified are identical to those claimed for the SFR.

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

The section entitled “Key-based Authentication with SSH” in the Admin Guide lists the host key algorithms, encryption ciphers/algorithms, key exchange methods, and MAC algorithms supported by the TOE in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Testing Assurance Activities: Test 1: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

Test 1: The evaluator attempted to establish an SSH connection with the TOE using diffiehellman-group1-sha1 key exchange. The connection attempt failed.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using each allowed key exchange method: diffie-hellman-group14-sha1, ecdh-sha2-nistp256, diffie-hellman-group16-sha512, ecdh-sha2-nistp384, and ecdh-sha2-nistp521. The connection succeeded in each case.

2.2.11.8 NDcPP22e:FCS_SSHS_EXT.1.8

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

a) Both thresholds are checked by the TOE.

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

Section 6.2 Cryptographic support (NDcPP22e:FCS_SSHS_EXT.1.8) of the ST states The TOE is capable of rekeying. The TOE verifies the following thresholds:

- No longer than one hour
- No more than one gigabyte of transmitted data

The TOE continuously checks both conditions. When either of the conditions is met, the TOE will initiate a rekey.

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

The section entitled “Key-based Authentication with SSH” in the Admin Guide states that the SSH connections to the sensor will rekey after processing increments of 1GB of data or at intervals of 1 hour.

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

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

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

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8).

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

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

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

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

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

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

The SSH data and time rekey thresholds are not configurable so the default of 1 GB and 1 hour were tested. The evaluator attempted to connect to the TOE using a SSH client generating 1 GB of data and verified that a rekey happened when the threshold was reached. The evaluator attempted to connect to the TOE using a SSH client waiting an hour and verified that a rekey happened right before the 1-hour threshold.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.12 TLS Server Protocol Without Mutual Authentication - per TD0635 (NDcPP22e:FCS_TLSS_EXT.1)

2.2.12.1 NDcPP22e:FCS_TLSS_EXT.1.1

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

Section 6.2 Cryptographic support (NDcPP22e: FCS_TLSS_EXT.1.1) of the ST states the TOE supports the following ciphersuites:

- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
- TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,

- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289

This is consistent with the SFR claims.

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

The section entitled "Web UI Connections" in the Admin Guide states that the TOE uses TLS protocol version 1.2 and it lists the supported ciphers, key exchange methods and algorithms allowed in the evaluated configuration. The TOE acts as the TLS server to offer its Web GUI and no additional configuration is needed from the user.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

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

Test 2: The evaluator shall 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.

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

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

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

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

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

Test 1: The evaluator attempted to connect to the TOE using each of the claimed ciphersuites. A packet capture was obtained for each connection attempt. The evaluator confirmed that for each of the claimed ciphersuites, the connection was successful.

Test 2: The evaluator attempted to connect to the TOE using the TLS_NULL_WITH_NULL_NULL ciphersuite and observed that the connection was rejected. The evaluator then attempted to connect to the TOE using all ciphers except those defined in the PP and observed that the TOE rejected the connection attempt.

Test 3: (a,b): The evaluator made connection attempts from a client to the TOE. The client implementation of the TLS protocol was modified as stated in parts a and b of the assurance activity. In Scenario 3 b), the evaluator observed the packet capture and ensured that the first byte of the encrypted Finished message does not equal 0x14.

2.2.12.2 NDcPP22e:FCS_TLSS_EXT.1.2

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

Section 6.2 Cryptographic support (NDcPP22e: FCS_TLSS_EXT.1.2) of the ST states the TOE's server includes a non-modifiable configuration that prohibits all TLS versions other than v1.2. If the TOE receives a TLS attempt using a version other than v1.2 the connection attempt will be rejected.

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

The section entitled "Web UI" in the Admin Guide states that the TOE uses TLS protocol version 1.2 and it lists the supported ciphers, key exchange methods and algorithms allowed in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

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

The evaluator attempted to establish a TLS session on the TOE with each of the unsupported versions of SSL and TLS. The evaluator used a network sniffer to capture the session negotiation and observed that the expected protocol and version were offered and rejected during negotiation.

2.2.12.3 NDcPP22e:FCS_TLSS_EXT.1.3

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

Section 6.2 Cryptographic support (NDcPP22e: FCS_TLSS_EXT.1.3) of the ST states the TOE includes ECDHE ciphersuites and supports curves secp256r1, secp384r1, secp521r1.

Diffie-Hellman groups are not supported by the TOE.

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

The section entitled "Web UI" in the Admin Guide states that the TOE uses TLS protocol version 1.2 and it lists the supported ciphers, key exchange methods and algorithms allowed in the evaluated configuration.

Additionally, the sections entitled "FIPS mode requirement" and "Enabling Common Criteria mode" in the Admin Guide state that FIPS compliance is a prerequisite for the Common Criteria certification. The sensor must be configured to operate in FIPS mode, which among other things imposes limitations on availability of algorithms from the cryptographic module. It is necessary to enable Common Criteria mode in order to make adjustments to

allowable algorithms as well as other security parameters which put the system into a state that is compliant with the certification. When the Common Criteria mode is enabled, a limited set of algorithms will be enforced by the system. This is entirely governed by the appliance and there are no external controls. Only those algorithms claimed as part of the certification will be available.

Testing Assurance Activities: Test 1: [conditional] If ECDHE ciphersuites are supported:

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

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

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).

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.

Test 1: The evaluator attempted to establish a TLS session with the TOE and configure the remote peer (i.e., the client) to offer each supported ECDH key exchange. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is accepted by the TOE when using supported ECDH key exchange groups. The evaluator then configured the remote peer (i.e., the client) to offer up an unsupported ECDHE curve size (P-192) and verified that the TOE rejected the connection.

Test 2: Not applicable. The TOE does not support DHE ciphersuites.

Test 3: The evaluator attempted to establish a TLS session with the TOE and configure the remote peer (i.e., the client) to offer each supported RSA key exchange. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is accepted by the TOE when using supported RSA key exchange methods.

2.2.12.4 NDcPP22e:FCS_TLSS_EXT.1.4

TSS Assurance Activities: 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).

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.

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.

Section 6.2 Cryptographic support (NDcPP22e: FCS_TLSS_EXT.1.4) of the ST states the TOE supports session resumption based on session tickets that adhere to the format provided in section 4 of RFC 5077. The session tickets are encrypted using symmetric algorithms AES used in CBC and GCM modes and key sizes of 128 and 256 bits. This is consistent with FCS_COP.1/DataEncryption.

Guidance Assurance Activities: None Defined

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

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

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

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

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps: Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

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

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

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

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.

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

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with a ServerHello with an empty SessionTicket extension, NewSessionTicket, ChangeCipherSpec and Finished messages (as seen in figure 2 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.

Test 1: Not applicable. The TOE supports session resumption based on session tickets.

Test 2: Not applicable. The TOE does not support session resumption based on session IDs.

Test 3: The evaluator first attempted to resume a session using a valid session ticket. The server correctly reused the client's proposed session ticket and the session was successfully resumed. In this case, the sessionID in the second ServerHello packet matches the client Hello's proposed sessionID.

The evaluator then attempted a second connection in which the session ticket was modified to be different from the server provided session ticket. The evaluator verified that the Client Hello's proposed Session Ticket was rejected as an invalid ticket, and the server sent a new NewSessionTicket packet.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 Identification and authentication (FIA)

2.3.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.3.1.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 NDcPP22e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it contains a description, for each supported method for remote administrative actions, of how successive unsuccessful authentication attempts are detected and tracked. The TSS shall also describe the method by which the remote administrator is prevented from successfully logging on to the TOE, and the actions necessary to restore this ability.

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

Section 6.3 Identification and authentication (NDcPP22e:FIA_AFL.1) of the ST states the TOE allows the administrator to configure the number of successive failed authentication attempts.

When a user fails to authenticate a number of times equal to the configured limit, the TOE locks the claimed user identity until the configured time is reached.

Administrators can configure unsuccessful authentication attempts range between 3 – 15 within 60 minutes. When the account is locked, the TOE does not permit any further actions until the account is accessible.

The authentication failures cannot lead to a situation where no administrator access is available since the local CLI would not be subject to lockout.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the remote administrator to once again successfully log on is described for each 'action' specified (if that option is chosen). If different actions or mechanisms are implemented depending on the secure protocol employed (e.g., TLS vs. SSH), all must be described.

The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

The section entitled "Enabling temporary account lockout for remote connections" in the Admin Guide provides instructions to enable and disable the account lockout policy which sets the number of successive unsuccessful authentication attempts before an account becomes locked. The instructions state that the administrator can define the policy to keep the account locked until the locked account can be automatically unlocked after a specified period. This section provides instructions to define the unlock period.

This section also states that the TOE can be configured to block remote login requests from a user account for a configurable period of time after a configurable number of failed remote login attempts. Admin accounts are never locked out from using the local console.

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

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

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

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

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

Test 1: The evaluator configured the maximum number of failed login attempts to 3. The evaluator then logged in to the TOE using incorrect credentials three times. The evaluator then attempted to use correct credentials and found that the account had been locked out in each case.

Test 2: This test has been performed in conjunction with test 1 above where the evaluator waited for just less than the time period configured for the lockout and verified that an authentication attempt using valid credentials did not result in successful access.

2.3.2 Password Management - per TD0792 (NDcPP22e:FIA_PMG_EXT.1)

2.3.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

(TD0792 applied)

Section 6.3 Identification and authentication (NDcPP22e:FIA_PMG_EXT.1) of the ST states the TOE provides the following password management capabilities for administrator passwords:

- Passwords shall be able to be composed of any combination of upper and lower case letters, numbers, and the following special characters: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)” ,“ ~ “,“ “.

- Minimum password lengths shall be configurable to 8 characters to maximum of 64 characters. The default minimum password length is 8 characters.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it:

a) identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and

b) provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

The section entitled "Password Requirements" in the Admin Guide contains steps that identifies the characters that may be used in passwords. This section includes guidance to security administrators on the composition of strong passwords. It also requires security administrators to set the minimum password length to 8 characters.

Also, this section explains that the administrator may configure a minimum password length between 8 and 64 characters.

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

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

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

Test 1: The evaluator successfully set valid user passwords on the TOE with compositions including the following:

-Minimum configurable password length

-Demonstrating special character set

-Demonstrating number set

-Demonstrating uppercase set

-Demonstrating lowercase set

-Maximum password length

Test 2: The evaluator attempted to set invalid user passwords on the TOE with compositions including the following. All invalid password attempts were rejected.

-Short password

-Long password

-No numbers

-No special character

-No uppercase

-No lowercase

2.3.3 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.3.3.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

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

The section entitled "Obscured Password" in the Admin Guide states that passwords are obscured by default on all TOE interfaces and that no further configuration is required.

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

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

Test 1: This test was performed as part of the tests for FIA_UIA_EXT.1 where the evaluator observed that passwords are obscured on the console logins.

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

2.3.4.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

This is covered under FIA_UIA_EXT.1.

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

2.3.5.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no unauthorized access to any TOE component can occur.

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

Section 6.3 Identification and authentication (NDcPP22e:FIA_UIA_EXT.1) of the ST states The TOE does not permit any actions prior to Administrators logging into the TOE. They can view the banner at the login prompt.

Administrative access to the TOE is facilitated through one of several interfaces:

- Connecting to the console port by plugging a keyboard and monitor directly into the ports on the back of the TOE
- Remotely connecting to each appliance via SSHv2
- Remotely connecting to each appliance via the TOE's TLS Web UI

For local administration, the TOE prompts the user for a username and password. When the user provides the correct username and password, this is compared to the known user database and if they match then the user is granted access. Otherwise, the user will not be granted access to the TOE. The TOE does not provide a reason for failure in the cases of a login failure.

For remote SSH administration, the TOE supports RSA and ECDSA public key authentication. If the user uses public key-based authentication and it is successful, then the user is granted access to the TOE. If the user enters invalid user credentials, they will not be granted access and will be presented the login page. Username and password is also supported.

For remote TLS Web UI administration, the TOE supports a username and password. When the user provides the correct username and password, this is compared to the known user database and if they match then the user is

granted access. Otherwise, the user will not be granted access to the TOE. The TOE does not provide a reason for failure in the cases of a login failure.

The TOE is not distributed.

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

The TOE provides the ability for security administrators to securely configure and manage the TOE via a locally connected terminal, a remote SSH connection or an HTTPS/TLS Web-based GUI connection.

The section entitled "Initial onboarding of a brand-new appliance" in the Admin Guide describes how to initially setup the TOE from a brand-new state and it contains the procedure for creating a new administrator account which will then be used to access and configure the TOE either locally through the local console or remotely through the Web UI or SSH interfaces.

The section entitled "Web UI" in the Admin Guide states that the TOE uses TLS protocol version 1.2 and it lists the supported ciphers, key exchange methods and algorithms allowed in the evaluated configuration.

The section entitled “Key-based Authentication with SSH” in the Admin Guide details how to access the TOE's SSH interface and it lists the allowed ciphers, key exchange methods and algorithms allowed in the evaluated configuration.

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

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

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

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

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

Test 1: The evaluator performed an unsuccessful and successful logon of each type (SSH pubkey, SSH password, Web UI and Local Console) using bad and good credentials respectively, and observed that when providing correct credentials, the connection was successful, but providing incorrect credentials resulted in a denial of access.

Test 2: There are no services available prior to login. The evaluator was able to observe the TOE display a banner to the user before login as identified in the tests performed in FTA_TAB.1.

Test 3: No functions were available to the administrator accessing the console with the exception of displaying the banner.

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

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

2.3.6.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.6.2 NDcPP22e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.3 Identification and authentication (NDcPP22e:FIA_X509_EXT.3) of the ST states in addition to the Common Name, Organization, Organizational Unit, and Country, the TOE additionally allows an administrator to specify the State, Locality, Email, and a list of Subject Alternative Names that the TOE will include in the Certificate Signing Request (CSR) it generates.

Device-specific information is not selected.

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

The section entitled "CSR generation" in the Admin Guide contains steps to generate a CSR for use by the TOE as a TLS Server. The steps identify the command that is used to generate a CSR which can be exported from the TOE. This command takes a number of arguments on the command line which prompts the user for values used in the certificate. The values which the TOE prompts for are Common Name, Country, Email, Locality, Organization Unit, Organization, and State.

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

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

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

Test 1: The evaluator followed operational guidance to log into the TOE and then generate a CSR. The evaluator captured the generated request and ensured that it contains the information claimed in the ST.

Test 2: The evaluator attempted to import a certificate without a valid certification path and the import failed. The evaluator then attempted to import a certificate with a valid certification path and the import succeeded.

2.4 Security management (FMT)

2.4.1 Management of Security Functions Behaviour (NDcPP22e:FMT_MOF.1/AutoUpdate)

2.4.1.1 NDcPP22e:FMT_MOF.1.1/AutoUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

For non-distributed TOEs, the evaluator shall ensure the TSS describes how checking for automatic updates or automated updates (whichever is supported by the TOE) are performed to include required configuration settings to enable and disable automated checking or automated updates.

Section 6.4 Security management (NDcPP22e:FMT_MOF.1/AutoUpdate) of the ST states the TOE permits the administrator (either through the TOE's Web UI or through its CLI) to enable the TOE to contact Corelight's servers to automatically check for new firmware releases. Additionally, the administrator can further specify whether the TOE should automatically download and install new firmware release, as they become available. The security administrator can enable automatic updates on the TOE by toggling on the feature "Install software updates" under the configuration -> maintain screen. Once that feature is enabled, the TOE will automatically apply any software updates as they become available. If disabled, the TOE lists the available updates (without applying them) so that the security administrator can then choose which update to apply manually.

The TOE is not distributed.

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

For non-distributed TOEs, the evaluator shall also ensure the TSS describes how checking for automatic updates or automated updates (whichever is supported by the TOE) are performed to include required configuration settings to enable and disable automated checking or automated updates (whichever is supported by the TOE).

The TSS states that the TOE permits the administrator (either through the TOE's Web UI or through its CLI) to enable the TOE to contact Corelight's servers to automatically check for new firmware releases. Additionally, the administrator can further specify whether the TOE should automatically download and install new firmware releases, as they become available. The security administrator can enable automatic updates on the TOE by toggling on the feature "Install software updates" under the configuration -> maintain screen. Once that feature is enabled, the TOE will automatically apply any software updates as they become available. If disabled, the TOE lists the available updates (without applying them) so that the security administrator can then choose which update to apply manually.

Component Testing Assurance Activities: The evaluator shall try to enable and disable automatic checking for updates or automatic updates (whichever is supported by the TOE) without prior authentication as Security Administrator (by authenticating as a user with no administrator privileges or without user authentication). The attempt to enable/disable automatic checking for updates 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 automatic checking for updates can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

The evaluator shall try to enable and disable automatic checking for updates or automatic updates (whichever is supported by the TOE) with prior authentication as Security Administrator. The attempt to enable/disable automatic checking for updates should be successful.

The evaluator confirmed that enabling automatic updates on the TOE can't be executed prior to authentication.

The evaluator then provided the correct credentials for login. Once the evaluator logged in successfully to the TOE, only then, he was able to enable/disable automatic updates on the TOE.

2.4.2 Management of Security Functions Behaviour (NDcPP22e:FMT_MOF.1/Functions)

2.4.2.1 NDcPP22e:FMT_MOF.1.1/Functions

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

For non-distributed TOEs, the evaluator shall ensure the TSS 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).

Section 6.4 Security management (NDcPP22e:FMT_MOF.1/Functions) of the ST states the TOE allows the administrator to configure and modify transmission of the TOE's audit functionality via SFTP.

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

For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation 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.

The sections entitled "Audit server requirements for audit log export", "Audit Log Export", and "Enabling Audit Log Export" in the Admin Guide describe how the Security Administrator modifies the behavior of transmitting audit data to an external IT entity, handling of audit data, and audit functionality when Local Audit Storage Space is full.

Component Testing Assurance Activities: 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.

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.

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.

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.

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.

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.

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.

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.

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.

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.

Test 1: This test has been performed in conjunction with test 3.

Test 2: This test has been performed in conjunction with test 4.

Test 3: The evaluator confirmed that no functions (for example, modifying audit data behavior) can be executed prior to authentication.

Test 4: The evaluator provided the correct credentials for login. Once the evaluator logged in successfully to the TOE, only then, he was able to perform functions (for example, modifying audit data behavior) and execute them.

The evaluator confirmed that with prior authentication, the administrator is able to perform these actions:

1) Delete the audit records and verify that only the records authorized for deletion are deleted.

2) Modify the behaviour of the transmission of audit data to an external IT entity.

The effects of the modifications (aka the export is configured properly) are verified in NDcPP22e:FAU_STG_EXT.1-t2 and NDcPP22e:FTP_ITC.1-t1 respectively.

2.4.3 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate)

2.4.3.1 NDcPP22e:FMT_MOF.1.1/ManualUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

There are no specific requirements for non-distributed TOEs.

The TOE is not distributed.

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

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

The section entitled "Appliance Software Updates" in the Admin Guide details installing and updating the TOE version including all steps necessary to perform manual updates.

Component Testing Assurance Activities: The evaluator shall try to perform the update using a legitimate update image without prior authentication as Security Administrator (either by authentication as a user with no administrator privileges or without user authentication at all - depending on the configuration of the TOE). The attempt to update the TOE should fail.

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

The TOE restricts the ability to perform manual updates to Security Administrators with prior authentication. This has been tested in NDcPP22e:FIA_UIA_EXT.1 where the evaluator showed that the TOE does not offer any services through the administrative interface prior to authenticating the connecting user other than the display of the TOE warning banner. The successful update with prior authentication is demonstrated in Test Case NDcPP22e:FPT_TUD_EXT.1.

2.4.4 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData)

2.4.4.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 Security management (NDcPP22e:FMT_MTD.1/CoreData) of the ST states Administrative users are required to login before being provided with access to any administrative functions. They can view the banner at the login prompt (both CLI and Web UI). The TOE restricts the ability to manage the TOE to Security Administrators. This includes handling of X509 certificates and the trust store.

The TOE maintains the following role: Security administrator (Admin). The role defined has a set of permissions that will grant them access to the TOE data.

Component Guidance Assurance Activities: The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

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

The section entitled "Accessing the appliance to make configuration changes" in Admin Guide states that no features aside from login are available until after successful authentication on all supported interfaces (local console, SSH, and Web UI).

The section entitled “CSR generation” in the Admin Guide provides information for the Admin to configure and maintain the trust store in a secure way, securely load CA certificates into the trust store and how to designate a CA certificate as a trust anchor.

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

No separate testing for FMT_MTD.1/CoreData is required as all of the management functions have already been exercised under other SFRs.

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

2.4.5.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.4 Security management (NDcPP22e:FMT_MTD.1/CryptoKeys) of the ST states The TOE allows the administrator to generate new SSH host keys, to generate (CSR) or import the TOE's TLS server certificate (for the Web UI).

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

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

The section entitled "CSR generation" in the Admin Guide describes the process for generating a CSR on the TOE and the process for importing the signed certificate.

The section entitled "SFTP Authentication" describes the process for configuring the SSH public key used by the TOE to authenticate the SFTP server.

Component Testing Assurance Activities: The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that

access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

The evaluator logged out of the TOE and verified that without prior authentication, no cryptographic keys changes can be performed on the TOE.

The successful generation of a new CSR and private key and the subsequent import of a signed certificate by an authorized administrator is demonstrated in NDcPP22e:FIA_X509_EXT.3-t1.

2.4.6 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1)

2.4.6.1 NDcPP22e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_SSL_EXT.1, FTA_SSL.3, FTA_TAB.1, FMT_MOF.1(1)/ManualUpdate, FMT_MOF.1(4)/AutoUpdate (if included in the ST), FIA_AFL.1, FIA_X509_EXT.2.2 (if included in the ST), FPT_TUD_EXT.1.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(2)/Services, and FMT_MOF.1(3)/Functions (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

(containing also requirements on Guidance Documentation and Tests)

The evaluator shall examine the TSS, Guidance Documentation and the TOE as observed during all other testing and shall confirm that the management functions specified in FMT_SMF.1 are provided by the TOE. The evaluator shall confirm that the TSS details which security management functions are available through which interface(s) (local administration interface, remote administration interface).

The evaluator shall examine the TSS and Guidance Documentation to verify they both describe the local administrative interface. The evaluator shall ensure the Guidance Documentation includes appropriate warnings for the administrator to ensure the interface is local.

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

Section 6.4 Security management (NDcPP22e:FMT_SMF.1) of the ST states that the TOE provides all the capabilities necessary to securely manage the TOE. The administrative user can connect to the TOE using the CLI to perform the below functions via SSHv2, using the TOE's Web UI, or at the local console.

The Security Administrator (admin) has the following privileges:

- Ability to administer the TOE locally and remotely;
- Ability to configure the access banner;
- Ability to configure the session inactivity time before session termination or locking;
- Ability to update the TOE, and to verify the updates using [digital signature] capability prior to installing those updates;
- Ability to configure the authentication failure parameters for FIA_AFL.1;
- Ability to configure audit behaviour (e.g. changes to storage locations for audit; changes to behaviour when local audit storage space is full);
- Ability to manage the cryptographic keys,
- Ability to configure the cryptographic functionality;
- Ability to enable or disable automatic checking for updates or automatic updates;
- Ability to set the time which is used for time-stamps;
- Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors;
- Ability to import X509v3 certificates to the TOE's trust store,
- Ability to manage the trusted public keys database

Section "Local Console" in the Admin Guide describes how to Log in through the RJ-45 CONSOLE port to establish a CLI session. This section provides sufficient guidance for the administrator to ensure that the interface is local.

All security management functions and the corresponding configuration information are identified or referenced throughout this AAR with the requirement to which they apply.

Component Guidance Assurance Activities: See TSS Assurance Activities

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

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

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

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

2.4.7.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.7.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.7.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Section 6.4 Security management (NDcPP22e:FMT_SMR.2) of the ST states that the TOE maintains the following user role: Security Administrator (Admin). The Security Administrator can manage the TOE both locally and remotely.

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

The section entitled "Accessing the appliance to make configuration changes" in the Admin Guide indicates that the TOE can be managed using 1) a CLI that is accessed through a local console or via SSH 2) the Web GUI which is protected by TLS/HTTPS. The Guidance assurance activities associated with FCS_SSHS_EXT.1, and FCS_TLSS_EXT.1 identify the relevant sections of the Admin Guide which provide the cryptographic capabilities of the TOE's SSH and TLS servers.

Component Testing Assurance Activities: In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

All of the administrative interfaces were used throughout the course of testing including the HTTPS, the SSH, and the Local Console channels for the TOE. The evaluator tested the communication channels by applying the

FCS_TLSS and the FCS_SSHS tests to the relevant channels on the TOE. The Local Console was tested as part of all of the SFRs specific to the Local Console (for example, FTA_SSL_EXT.1).

2.5 Protection of the TSF (FPT)

2.5.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1)

2.5.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Section 6.5 Protection of the TSF (NDcPP22e:FPT_APW_EXT.1) of the ST states that all passwords are stored in a secure directory that is not readily accessible to administrators through any interface. The passwords are stored as SHA-512 salted hash.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.2.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.5 Protection of the TSF (NDcPP22e:FPT_SKP_EXT.1) of the ST states the TOE stores all pre-shared keys, symmetric keys and private keys in plaintext in a secure storage location that is not accessible through an interface to administrators.

Refer to NDcPP22e:FCS_CKM.4 for more details on key storage and destruction.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.3.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.3.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.5 Protection of the TSF (NDcPP22e:FPT_STM_EXT.1) of the ST states The TOE provides reliable time stamps. The clock function is reliant on the system clock provided by the underlying hardware.

The following security functions make use of the time:

- Audit events
- Session inactivity
- Certificate validation (for validity/expiration)

'Obtain time from the underlying virtualization system' is not selected.

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

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

The section entitled "Setting the clock manually" in the Admin Guide contains instructions for setting the time on the TOE manually.

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

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

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

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

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

Test 1: The evaluator manually changed the time and observed that the TOE's time changed.

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

Test 3: Not applicable. The TOE does not obtain time from an underlying VS.

2.5.4 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.5.4.1 NDCPP22E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

For distributed TOEs the evaluator shall examine the TSS to ensure that it details which TOE component performs which self-tests and when these self-tests are run.

Section 6.5 Protection of the TSF (NDcPP22e:FPT_TST_EXT.1) of the ST states all crypto algorithms used by the TOE go through power up self-tests (KAT) before they can be used to provide service. The TOE executes the following power-on self-tests using Known Answer Tests (in which the TOE tests the algorithm using known inputs [plaintext, keys, messages, seed, etc.] and verifies that the algorithm produces the expected/known output [signature, ciphertext, hash, random output, etc.]):

- SHA KAT
- AES KAT
- HMAC SHA KAT
- DRBG KAT

- ECDH KAT
- GCM KAT
- RSA KAT
- ECDSA KAT

When device detects a failure during one or more of the self-tests, it raises an alarm. The administrator can attempt to reboot the TOE to clear the error. If rebooting the device does not resolve the issue, then the administrator should contact their next level of support or their Corelight support group for further assistance. All power up self-tests execution is logged for both successful and unsuccessful completion.

The Software Integrity Test is run automatically on start-up, and whenever the system images are loaded. These tests are sufficient to verify that the correct version of the TOE software is running as well as that the cryptographic operations are all performing as expected.

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

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

The section entitled "Self-Tests" in the Admin Guide explains that failure of any self-test during the start-up process requires the admin to restart the system and allow the tests to run again. If the failure persists, the system may be compromised, and the appliance must be reinstalled.

Component Testing Assurance Activities: It is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE

b) Verification of the correct operation of the cryptographic functions necessary to fulfill any of the SFRs.

Although formal compliance is not mandated, the self-tests performed should aim for a level of confidence comparable to:

a) FIPS 140-2, chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

b) FIPS 140-2, chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions. Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

The evaluator shall either verify that the self tests described above are carried out during initial start-up or that the developer has justified any deviation from this.

For distributed TOEs the evaluator shall perform testing of self-tests on all TOE components according to the description in the TSS about which self-test are performed by which component.

The evaluator initiated a reboot and confirmed that the status messages show that the FIPS self-tests were executed successfully.

2.5.5 TRUSTED UPDATE (NDCPP22E:FPT_TUD_EXT.1)

2.5.5.1 NDCPP22E:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.2 NDCPP22E:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.3 NDCPP22E:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describe how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the TSS needs to describe how and when the inactive version becomes active. The evaluator shall verify this description.

The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system firmware and software (for simplicity the term 'software' will be used in the following although the requirements apply to firmware and software). The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital

signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

If the options 'support automatic checking for updates' or 'support automatic updates' are chosen from the selection in FPT_TUD_EXT.1.2, the evaluator shall verify that the TSS explains what actions are involved in automatic checking or automatic updating by the TOE, respectively.

For distributed TOEs, the evaluator shall examine the TSS to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component. Alternatively, this description can be provided in the guidance documentation. In that case the evaluator should examine the guidance documentation instead.

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

Section 6.5 Protection of the TSF (NDcPP22e:FPT_TUD_EXT.1) of the ST states Security Administrators can query the current version of the TOE and they are able to perform manual software updates. The currently active version of the TOE can be queried by issuing the “corelight-client information get” command.

When software updates are become available, the administrator can obtain and install the updates. The TOE can alternatively automatically download new updates as described in NDcPP22e:FMT_MOF.1/AutoUpdate.

The software images are digitally signed using RSA digital signature mechanism. The TOE will use a public key in order to verify the digital signature, upon successful verification the image will be loaded onto the TOE. If the images cannot be verified, the image will not be loaded onto the TOE.

Section 6.4 Security management (NDcPP22e:FMT_MOF.1/AutoUpdate) of the ST states the TOE permits the administrator (either through the TOE’s Web UI or through its CLI) to enable the TOE to contact Corelight’s servers to automatically checked for new firmware releases. Additionally, the administrator can further specify whether the TOE should automatically download and install new firmware release, as they become available.

The TOE is not distributed.

Public hash verification is not claimed.

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

The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall

include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

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

For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT_TUD_EXT.1, how all TOE components are updated, and the error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. The guidance documentation only has to describe the procedures relevant for the Security Administrator; it does not need to give information about the internal communication that takes place when applying updates.

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

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

The section entitled "Appliance Software Updates" in the Admin Guide explains that security updates are signed by the vendor and verified during the update installation. This section also provides instructions to determine the currently running version.

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

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

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

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

2) An image that has not been signed

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

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

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

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

2) The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value to the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to

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

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

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

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

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

Test 1: The evaluator displayed the version of the TOE and then installed an update. The signature verified and the update was successfully installed.

Test 2: The evaluator attempted to upload three images: a corrupted image, an image with an invalid signature and an image missing a signature. In each case, the TOE checked the integrity of the image and correctly detected an invalid image file and rejected the update.

Test 3: Not applicable. Published hash is not used to verify the integrity of the TOE updates.

2.6 TOE ACCESS (FTA)

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

2.6.1.1 NDCPP22E:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 TOE access (NDcPP22e:FTA_SSL.3) of the ST states a Security Administrator can configure maximum inactivity times for administrative sessions through the TOE local CLI and remote SSH interfaces. The default inactivity time period is 60 minutes for both the CLI and SSH interfaces. The configuration of inactivity period is applied on a per interface basis. A configured inactivity period will be applied to both local and remote sessions in the same manner. When the interface has been idle for more than the configured period, the session will be terminated and will require authentication to establish a new session.

The TOE will terminate the local/SSH administrative session after a Security Administrator defined period of inactivity. The inactivity time-out for CLI can be configured by the following commands:

```
corelight-client configuration update --security.auto_logout.enable=1
corelight-client configuration update --security.auto_logout.timeout=1
```

The TOE maintains separate settings for its Web UI, which allow the Security Administrator to configure the idle user session timeout (in minutes). By default, the TOE does not enforce an inactivity time period, but once the admin enabled the auto logout, the TOE allows the admin to configure the timeout in minutes. The configuration of inactivity period is applied to all Web UI admin sessions. When the interface has been idle for more than the configured period, the TOE will automatically logout (effectively terminate) the administrator from their Web UI session and require authentication to establish a new session.

The TOE's Web UI (under "Login Behaviors") allows an administrator to "Enable auto logout for idle user sessions" and then specify the "Timeout (min)".

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

The section entitled "Enabling Inactivity Timeout" in the Admin Guide explains that the administrator can configure the number of minutes of inactivity after which the TOE logs the user out and terminates the session.

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

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

For each method of remote administration on the TOE, the evaluator configured two different idle timeout values. The evaluator then logged into the device and observed that after the configured amount of time, the TOE terminated the evaluator's session.

2.6.2 User-initiated Termination (NDcPP22e:FTA_SSL.4)

2.6.2.1 NDcPP22e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.6 TOE access (NDcPP22e:FTA_SSL.4) of the ST states the Security Administrator can terminate their local CLI and remote SSH sessions by typing “exit” at the prompt. Likewise, the administrator can terminate their remote Web UI session by closing their web browser or selecting “logout”.

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

The section entitled "Session Termination" in the Admin Guide explains that a user can log out of an active session using the logout or exit commands.

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

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

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

Test 1: This test was performed as part of FIA_UIA_EXT.1-t1 where the evaluator performed a logout for each interactive console session after a successful authentication attempt.

Test 2: This test was performed as part of FIA_UIA_EXT.1-t1 where the evaluator performed a logout for each remote interactive session after a successful authentication attempt.

2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.6.3.1 NDcPP22e:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details whether local administrative session locking or termination is supported and the related inactivity time period settings.

Section 6.6 TOE access (NDcPP22e:FTA_SSL_EXT.1) of the ST states a Security Administrator can configure maximum inactivity times for administrative sessions through the TOE local CLI and remote SSH interfaces. The default inactivity time period is 60 minutes for both the CLI and SSH interfaces. The configuration of inactivity period is applied on a per interface basis. A configured inactivity period will be applied to both local and remote sessions in the same manner. When the interface has been idle for more than the configured period, the session will be terminated and will require authentication to establish a new session.

The TOE will terminate the local/SSH administrative session after a Security Administrator defined period of inactivity. The inactivity time-out for CLI can be configured by the following commands:

```
corelight-client configuration update --security.auto_logout.enable=1
corelight-client configuration update --security.auto_logout.timeout=1
```

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

The section entitled "Enabling Inactivity Timeout" in the Admin Guide explains that the administrator can configure the number of minutes of inactivity after which the TOE logs the user out and terminates the session.

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

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

Test 1: The evaluator followed operational guidance to set the console timeout on the TOE to both 3 and 6 minutes and observed that the user was logged out automatically in 3 and 6 minutes, respectively.

2.6.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.6.4.1 NDcPP22e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access, and might be configured during initial configuration (e.g. via configuration file).

Section 6.6 TOE access (NDcPP22e:FTA_TAB.1) of the ST states Security Administrators can create a customized login banner that will be displayed at the following interfaces:

- Local CLI
- Remote CLI via SSH v2
- Remote Web UI via TLS

This banner will be displayed prior to allowing Security Administrator access through those interfaces.

The banner will be same for local CLI and SSH remote methods of access and can be configured during initial configuration. The TOE maintains a separate banner for the Web UI , which that admin can configure and update.

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

The section entitled "Login Banner" in the Admin Guide provides instructions for setting a login banner for all TOE interfaces. The login banner is displayed on all TOE interfaces prior to logging in.

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

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

Test 1: The evaluator followed the steps outlined in the guidance to configure a login banner, then logged into the TOE via each possible login method to see the login banner updated and displayed properly.

2.7 Trusted path/channels (FTP)

2.7.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1)

2.7.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 NDcPP22e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 Trusted path/channels (NDcPP22e:FTP_ITC.1) of the ST states the TOE supports secure communication to the following IT entities: Audit server (via SFTP). The TOE protects communications between the TOE and the SFTP audit server using SFTP (FTP over the SSH v2 protocol). The TOE acts as a client in all cases. The protocols listed are consistent with those included in the requirements in the ST.

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

The section entitled "Audit Log Export" in the Admin Guide includes steps to configure the TOE to connect to an external SFTP server to export logs using SSH.

The section entitled "Connection Loss Recovery" in the Admin Guide states that the TOE will reconnect the SSH pathway after an unintentional disconnect.

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

The evaluator shall perform the following tests:

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

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

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

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

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

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

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

Further assurance activities are associated with the specific protocols.

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

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

Test 1: Refer to NDcPP22e:FCS_SSHC_EXT.1 where the SSH Channel is fully tested.

Test 2: Refer to NDcPP22e:FTP_ITC.1-t4 where the TOE initiating the trusted channel has been demonstrated.

Test 3: Refer to NDcPP22e:FTP_ITC.1-t4 where the TOE using an encrypted trusted channel has been demonstrated.

Test 4:

MAC Layer: The evaluator connected the TOE to the test peer/server and began to send some traffic between the devices. The connection was physically disrupted and after about 90-120 seconds the connection was restored. After the restoration, the evaluator observed the SSH connection had to be reestablished and data remained protected.

APP Layer: The evaluator connected the TOE to the test peer/server and began to send some traffic between the devices. The connection was physically disrupted and after about 5-6 minutes the connection was restored. After the restoration, the evaluator observed that the SSH connection had to be reestablished and data remained protected.

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

2.7.2.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.7 Trusted path/channels (NDcPP22e:FTP_ITC.1/Admin) of the ST states the TOE supports SSH v2.0 for secure remote CLI administration and supports HTTPS/TLSS for secure remote WebUI. SSH v2.0 session is encrypted using AES encryption to protect confidentiality and uses HMACs to protect integrity of traffic.

The TOE also supports TLS for secure remote Web UI administration of the TOE. The TLS v1.2 session is also encrypted using AES to protect confidentiality and uses HMAC or GCM to protect integrity of traffic. The protocols listed are consistent with those specified in the requirement.

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

The section entitled "Accessing the appliance to make configuration" in the Admin Guide details authentication methods for each type of remote administrator.

The section entitled "Web UI" in the Admin Guide states that the TOE uses TLS protocol version 1.2 and it lists the supported ciphers, key exchange methods and algorithms allowed in the evaluated configuration.

The section entitled “Key-based Authentication with SSH” in the Admin Guide details how to access the TOE's SSH interface and it lists the allowed ciphers, key exchange methods and algorithms allowed in the evaluated configuration.

The Web GUI (protected with TLS) and the SSH interfaces are the only two remote administrative interfaces identified by the Security Target and by evaluators during testing.

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

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

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

Further assurance activities are associated with the specific protocols.

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

Test 1 and test 2: The remote administration methods were tested throughout the course of the evaluation. The successful testing of the HTTPS channel and the demonstration of its encryption can be found in FCS_TLSS_EXT.1.

The successful testing of the SSH channel and the demonstration of its encryption can be found in FCS_SSHS_EXT.1.

3. Protection Profile SAR Assurance Activities

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

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

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

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

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

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

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

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

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

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

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

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

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

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

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. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

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

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

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

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

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

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

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

As identified throughout this AAR, the Admin Guide provides instructions for configuring the TOE's cryptographic security functions. The Admin Guide provides instructions for configuring the cryptographic algorithms and parameters used for the evaluated configuration. The Admin Guide is clear that no other cryptographic configuration has been evaluated or tested. There are also warnings and notes throughout the Admin Guide

regarding use of functions that are and are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP22e:FPT_TUD_EXT.1.

3.2.2 Preparative Procedures (AGD_PRE.1)

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

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

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

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

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

The preparative procedures must

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

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

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

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

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

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

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

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

3.3.2 TOE CM Coverage (ALC_CMS.1)

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

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

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

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

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

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

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

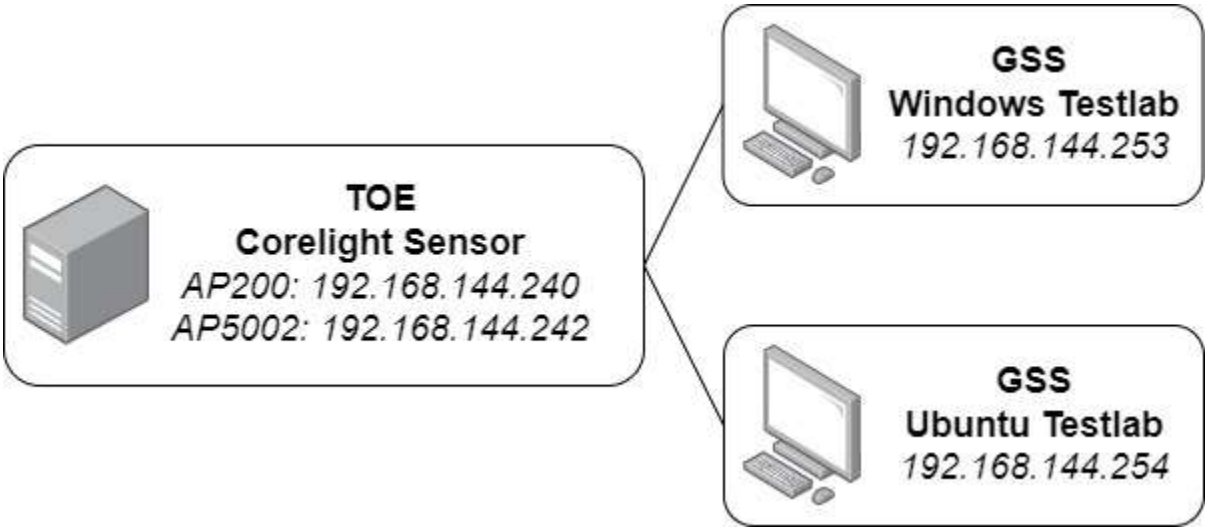
The evaluator should consult Appendix B when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

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

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

- AP200
- AP5002

Figure 1 of the DTR depicts the standard test network used during evaluation testing. The figure shows two Devices Under Test (DUTs), which represents the location of the TOEs in the network.



The Gossamer Test servers utilized both a windows and an Ubuntu environment. The test servers also acted as a syslog server. The Windows supporting software included the following:

- Windows 10 Pro
- Wireshark 4.0.3
- Windows SSH Client - Putty version 0.78 (used to connect to device console and SSH)

The Ubuntu supporting software included the following:

- OpenSSL 1.0.2g-fips
- Openssh client version 8.9
- Big Packet Putty, Openssh-client version 7.2
- Rsyslog version 8.16.0
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Stunnel 5.30

3.5 Vulnerability Assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different

evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

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

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

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

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

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

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

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

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

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

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

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

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

- National Vulnerability Database (https://web.nvd.nist.gov/vuln/search),
- Vulnerability Notes Database (http://www.kb.cert.org/vuls/),
- Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities),
- Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories),
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search),
- Offensive Security Exploit Database (https://www.exploit-db.com/)

The search was performed on December 20, 2024. The search was conducted with the following search terms: "Corelight", "Corelight Sensors", "Corelight Sensors with BroLin v28", "BroLin v28", "AP 200", "AP 5002", "AP 520", "AP 1001", "AP 1100", "AP 1200", "AP 3000", "AP 3100", "AP 3200", "AP 5200", "AP200", "AP5002", "AP520", "AP1001", "AP1100", "AP1200", "AP3000", "AP3100", "AP3200", "AP5200", "Intel Xeon Scalable Silver 4110", "Intel Xeon Scalable Gold 5317", "Intel Xeon Scalable Silver 4116", "Intel Xeon Scalable Silver 4314", "AMD EPYC 9254", "Intel Xeon Scalable Gold 6238", "Intel Xeon Scalable Gold 5318Y", "AMD EPYC 9534", "AMD EPYC 7742", "AMD EPYC 7713", "AMD EPYC 9754", "Linux Kernel version 5.4", "Linux Kernel 5.4", "SafeLogic", "SafeLogic OpenSSL 3.0", "OpenSSL 3.0".