www.GossamerSec.com

Assurance Activity Report for Aruba ClearPass Policy Manager 6.11

Version 1.0
03/21/23

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

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

Revision History

Revision	Date	Authors	Summary
Version 0.1	03/08/23	Cummins	Initial draft
Version 1.0	03/21/23	Cummins	Minor ECR updates

The TOE Evaluation was Sponsored by:
Aruba, a Hewlett Packard Enterprise company
6280 America Center Drive
San Jose, CA 95002

Evaluation Personnel:

- Cody Cummins
- Matai Spivey

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 Aruba ClearPass Policy Manager 6.11 evaluation. The TOE claims conformance to the following protection profiles:

- collaborative Protection Profile for Network Devices Version 2.2e, 23 March 2020 (NDcPP22e)
- Application Software Protection Profile Extended Package for Authentication Servers', Version 1.0, 07 August 2015 (AUTHSRVEP10)

This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

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

1.1.1 Evaluated Platform Equivalence

The TOE is the Aruba ClearPass Policy Manager Version 6.11.0 running in one of the following appliances: C1000, C2000, C2010, C2020 C3000, C3010 or C1000V.

Each platform differs in CPU performance (e.g., number of cores), available memory, disk performance and storage capacity, and power consumption/supply.

Appliance Model	CPU
C1000	Intel Atom C2758 (Rangeley)
C2000	Intel Xeon E3-1240 v5 (Skylake)
C2010	Intel Xeon E-2274G (Coffee Lake)
C2020	Intel Xeon Gold 5118 (Skylake)
C3000 (legacy only)	Intel Xeon E5-2620 v3 (Haswell)
C3010	Intel Xeon Gold 5118 (Skylake)
C1000V	ESXi 7.0 on Intel Xeon E-2254ML (Coffee Lake)

The evaluators executed the entire test suite on the C1000 and C1000V appliance model running ClearPass Policy Manager version 6.11.

Each ClearPass Policy Manager device is a rack-mountable appliance with one or more Intel Atom or Xeon CPUs running a version of RHEL 8 to host the applications designed to provide the network access control capabilities provided by the TOE. The hardware models that comprise the TOE have common hardware characteristics and contain similar processors from the Intel Xeon and Atom families. The exact same software image is used on all hardware platforms and there are no security relevant functions implemented by the network drivers. Since the same software is installed on all the hardware platforms and all security functions provided by the TOE are implemented in software; the TOE security behavior is the same on all the hardware models for each of the SFRs defined by the NDcPP22E/AuthSrvEP10. These SFRs are instantiated by the same version of the TOE software and in

the same way on every platform Any differing characteristics in the hardware affect only non-TSF relevant functionality such as throughput, processing speed, number and type of network connections supported and amount of storage.

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

1.1.2 CAVP CERTIFICATE JUSTIFICATION

The TOE has been CAVP tested. The following functions have been CAVP tested to meet the associated SFRs.

Requirements	Functions	Cert
	Cryptographic key generation	
FCS_CKM.1	RSA schemes using cryptographic key sizes of 2048-bit and 3072-bit	A3295
	ECC schemes using 'NIST curves' P-256 and P-384	A3295
	FFC schemes using cryptographic key sizes of 2048-bit (DSA)	A3295
	Cryptographic key establishment	
FCS_CKM.2	RSA-based key establishment schemes	Vendor Affirmed
	Elliptic curve-based key establishment schemes (KAS ECC)	A3295
	Finite field-based key establishment schemes (KAS FFC)	A3299
	Encryption/Decryption	
FCS_COP.1/Data Encryption	AES CBC (128 and 256 bits)	A3272
	AES GCM (128 and 256 bits)	A3286
	AES CTR (128 and 256 bits)	A3272
	Cryptographic signature services	
FCS_COP.1/SigGen	RSA Digital Signature Algorithm (rDSA) (2048 bits & 3072 bits)	A3295

	Elliptic Curve Digital Signature Algorithm (ECDSA) with an elliptical curve size of 256 or 384	A3295
	Cryptographic hashing	
FCS_COP.1/Hash	SHA-1/256/384/512 (digest sizes 160, 256, 384 bits and 512 bits)	A3295
	Keyed-hash message authentication	
FCS_COP.1/KeyedHash	HMAC-SHA-1 (block size 512 bits, key and digest size 160 bits) HMAC-SHA-256 (block size 512 bits, key and digest size 256 bits) HMAC-SHA-384 (block size 1024 bits, key and digest size 384 bits), HMAC-SHA-512 (block size 1024 bits, key and digest size 512 bits)	A3295
	Random bit generation	
FCS_RBG_EXT.1	CTR_DRBG (AES) with S/W based noise source	A3272

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. The following evidence was used to complete the Assurance Activities:

- Aruba ClearPass Policy Manager 6.11 Security Target, Version 1.0, 03/21/2023 [ST]
- Common Criteria Configuration Guidance Aruba ClearPass Policy Manager 6.11, March 2023 [Guide]

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 in the **ST** states that audit records include date and time of the event, type of event, user identity that caused the event to be generated, and the outcome of the event. For any auditable events related to cryptographic key operations, the key or certificate name is logged.

The TOE is not distributed.

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

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

Appendix A in the **Admin Guide** lists all of the auditable events and provides a reference and link to the ClearPass Policy Manager User guide (Administration section under “Export Event Format Types – Examples”) which provides the syslog format for each audit record along with a brief description of each field. The auditable event examples are provided in the syslog format.

Locally, these audit events can be viewed via the Web UI depending on their location which is identified for each audit event as either the Event Viewer, the Audit Viewer or the Access Tracker.

All administrative functions are included in the list of auditable events, covered under FMT_SMF.1. Refer to the writeup in FMT_SMF.1 which maps out where each management function steps are documented.

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 and collected these audit events when running the other security functional tests described by the protection profiles. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. The evaluator collected the required audits for all TOE components, via syslog, during the course of testing. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

The TOE is not distributed.

2.1.2 User identity association (NDcPP22e:FAU_GEN.2)

2.1.2.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See FAU_GEN.1.

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

See FAU_GEN.1.

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

For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated

for building up a secure channel between two TOE components. If for some reason (could be e.g. TSS or Guidance Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

See FAU_GEN.1. 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 of the ST states that there are three locations in the Web UI where audit records are stored and can be viewed: Access Tracker, Audit Viewer, and Event Viewer.

By default, the Access Tracker and Audit Viewer store the logs for 7 days after which time they will be deleted automatically. The automatic clean up period can be configured by the administrator to be longer or shorter as may be necessary for a given deployment. The Audit Viewer storage can be configured via the cleanup parameter “Old Audit Records Cleanup Interval”. The Access Tracker storage can be configured via the cleanup parameter “Cleanup interval for Session Log details in database”. The Event Viewer records are stored for seven (7) days after which time they will be deleted automatically. There is no user configurable setting to modify the Event Viewer log storage.

The number and size of log files may be specified based on observed logging levels. The default number of log files is 12 and the default size of each log file is 50MB.The specific capacity of the audit storage is dependent on the disk drive capability of the TOE. The default disk capacity has been designed so that in a typical deployment the available space will not be exhausted within the default retention periods. Disk usage settings will notify the administrator if the system is running with low disk space.

The TOE can also be configured to send audit records to a trusted third-party SYSLOG server in the operational environment. The TOE can be configured to use TLS to protect the communication channel between itself and the remote SYSLOG server.

The TOE is a standalone TOE that stores audit data locally and transfers audit data to an external syslog server periodically. ClearPass does not transfer syslog messages in real time. Messages are queued to a syslog buffer that then transfers all messages to the syslog server every 120 seconds. This value may be reduced to a minimum of every 30 seconds, but will default to every 120 seconds.

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

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

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

Section “Configure Audit Export” in the **Admin Guide** states ClearPass has limited storage space to retain logs. It is recommended to export all audit logs to an external source. The recommended process to accomplish this is via syslog export. This section describes how to configure the TOE to communicate with a syslog server via the System Level tab in the Web UI. The IP address of the external syslog server that will receive audit messages from ClearPass is specified, along with all the appropriate audit events to be sent. ClearPass does not transfer syslog messages in real time.

Section “FAU_STG_EXT.1” in the **Admin Guide** states messages are queued to a syslog buffer that then transfers all messages to the syslog server every 120 seconds. This value may be reduced to a minimum of every 30 seconds, but will default to every 120 seconds. The potential delay in message queue and receipt by the remote server should be noted to comply with Common Criteria evaluated settings.

Local audit records are stored for seven (7) days prior to automatic cleanup (deletion). The settings can be adjusted by modifying the value on the Cleanup Intervals tab. The Access Tracker events can be modified by adjusting the Cleanup interval for Session log details in the database parameter (default value is 7 days). The general audit (such as the Accounting) events can be modified by adjusting the Old Audit Records cleanup interval parameter (default value is 7 days). Event Viewer records are stored for seven (7) days prior to automatic cleanup. There is no user configurable setting to modify the Event Viewer log storage. Audit records that exceed cleanup intervals will be deleted from the file system and the space reclaimed to write new audit events. The number and size of log files may be specified based on observed logging levels. The default number of log files is 12 and the default size of each log file is 50MB. The number and size limits apply to all log file settings. Modifying these values will affect the log files that contain information created by RADIUS, Policy, and other services. Reducing the capacity may decrease the information available to less than seven (7) days. Increasing may cause issues with system free disk space thresholds.

The “FCS_IPSEC_EXT.1” section in the **Admin Guide** describes how to configure IPsec using the “Create IPsec Tunnel” interface in the Web UI between ClearPass and a remote device such as the syslog server using the “Create IPsec Tunnel” dialog box in the Web UI. This dialog box identifies a field to enter the remote IP address of the syslog server that the TOE will be connecting to as well as fields for entering the encryption algorithms and the type of authentication that will be used.

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 external audit server was utilizing rsyslogd version 8.16.0. The evaluator established a connection and demonstrated that the syslog messages were successfully sent to the syslog server through the encrypted channel. All of the audits demonstrated in NDcPP22E:FAU_GEN.1 were collected from the syslog server.

Test 2: For the three locations in the Web UI where audit records are stored and can be viewed (Access Tracker, Audit Viewer, and Event Viewer), the evaluator verified that after the number of days configured via the cleanup interval setting for the Audit Viewer and Access Tracker, the audit logs were deleted. The evaluator also verified that audit logs greater than 7 days old were deleted from the Event Viewer.

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

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

2.2 Communication (FCO)

2.2.1 Selective Proof of Origin (AUTHSRVEP10:FCO_NRO.1)

2.2.1.1 AUTHSRVEP10:FCO_NRO.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.1.2 AUTHSRVEP10:FCO_NRO.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.1.3 AUTHSRVEP10:FCO_NRO.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol to ensure that RADIUS encapsulated EAP Message Authenticators conform to RFC 3579.

Section 6.2 of the ST states that the TOE implements the RADIUS protocol in order to service authentication requests from associated NAS devices. The TOE requires RADIUS encapsulated EAP Message Authenticators that conform to RFC 3579 and each Access-Request from a NAS must have the correct Message Authenticator so that the NAS can be determined to be authentic. In response, the TOE includes its own identifier, Response Authenticator (conforming to RFC 2865), and the response packet with the requested authentication results.

Component Guidance Assurance Activities: The evaluator shall verify that the guidance contains all necessary instructions to configure RADIUS and encapsulated EAP on the TOE, in order to ensure that evidence of origin for all incoming RADIUS Access-Request packets is collected and preserved.

Section "FTP_ITC.1(1)" in the **Admin Guide** describes how to configure the ClearPass RADIUS service via the Web UI. The RADIUS service is configured via the Service tab in Configuration > Services, while the EAP-TLS authentication method is configured via the Authentication tab.

Section "Add Network Access Devices" in the **Admin Guide** describe how to add network access devices to the system before conducting RADIUS authentication events and how to configure a RADIUS shared secret key/password and to enable RadSec. To comply with Common Criteria evaluated status, all RADIUS communications should be encrypted between ClearPass and the NAD(s). When the communication between NAD and Policy Manager occurs over RadSec, the shared secret key/password is automatically set to "radsec" in compliance with RFC behavior. RadSec sessions will use certificate validation to establish communication and this is configured on the RadSec Settings tab.

Section "Configuring RadSec" in the **Admin Guide** states that Network Access Devices (NAD) can be configured to use either RADIUS or RadSec. When the option to Enable RadSec is selected on the NAD, Policy Manager will not accept communication from that device using RADIUS, RADIUS Accounting, or RADIUS Dynamic Authorization ports.

Section "FCS_IPSEC_EXT.1" in the **Admin Guide** details the basic information to establish IPsec tunnels which can also be used to protect RADIUS communication. If ports are restricted to RADIUS, ensure that RADIUS Accounting and RADIUS Dynamic Authorization are also allowed to pass through the IPsec tunnel to comply with CC evaluation configuration.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to update the ClearPass Server RADIUS and HTTPS certificates by installing certificates that are signed by a trusted certificate authority (CA). This is done via Administration -> Certificates -> Trust List in the Web UI.

The "FIA_X509_EXT.3.1" section of the **Admin Guide** provides details for creating a Certificate Signing Request including the following information fields to enter in the request: Common Name, Organization, Organizational Unit and Country. This is consistent with the claims for this requirement in the ST. A CSR can be generated of a selected type. The default will be RADIUS/EAP Server Certificate. Other valid selections include HTTPS Server Certificate, RadSec Server Certificate, and Database Server Certificate.

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

Test 1: The evaluator shall send a RADIUS Access-Request, from a NAS with which the TOE does not share a RADIUS secret, with NAS identification attributes correctly indicating the originating NAS, containing an encapsulated EAP-response message and a valid message-authenticator attribute. The evaluator shall verify that the TOE discards the request without responding.

Test 2: The evaluator shall send a RADIUS Access-Request, from a NAS with which the TOE does not share a RADIUS secret, with NAS identification attributes falsely indicating a NAS with which the TOE does share a RADIUS secret, containing an encapsulated EAP-response message and a valid message-authenticator attribute. The evaluator shall verify that the TOE discards the request without responding.

Test 1: The evaluator sent the TOE a RADIUS Access-Request from an eapol client device which was configured with valid identification attributes, but with a preshared key that did not match the RADIUS Shared secret configured for the client device on the TOE. The request failed and there was no response from the TOE.

Test 2: The evaluator sent the TOE a RADIUS Access-Request from an eapol client device with NAS identification attributes falsely indicating a NAS with which the TOE does share a RADIUS secret. The request failed and there was no response from the TOE.

2.2.2 Selective Proof of Receipt (AUTHSRVEP10:FCO_NRR.1)

2.2.2.1 AUTHSRVEP10:FCO_NRR.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.2.2 AUTHSRVEP10:FCO_NRR.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.2.3 AUTHSRVEP10:FCO_NRR.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol to ensure that RADIUS Response Authenticators conform to RFC 2865.

Section 6.2 of the ST states that in response to each Access-Request from a NAS determined to be authentic, the TOE includes its own identifier, Response Authenticator (conforming to RFC 2865), and the response packet with the requested authentication results.

Component Guidance Assurance Activities: The evaluator shall verify that the guidance contains all necessary instructions to configure RADIUS and encapsulated EAP on the TOE, in order to ensure that evidence of receipt of all incoming RADIUS Access-Request packets is generated and transmitted correctly.

Section "FTP_ITC.1(1)" in the **Admin Guide** describes how to configure the ClearPass RADIUS service via the Web UI. The RADIUS service is configured via the Service tab in Configuration > Services, while the EAP-TLS authentication method is configured via the Authentication tab.

Section "Add Network Access Devices" in the **Admin Guide** describe how to add network access devices to the system before conducting RADIUS authentication events and how to configure a RADIUS shared secret key/password and to enable RadSec. To comply with Common Criteria evaluated status, all RADIUS communications should be encrypted between ClearPass and the NAD(s). When the communication between NAD and Policy Manager occurs over RadSec, the shared secret key/password is automatically set to "radsec" in compliance with RFC behavior. RadSec sessions will use certificate validation to establish communication and this is configured on the RadSec Settings tab.

Section "Configuring RadSec" in the **Admin Guide** states that Network Access Devices (NAD) can be configured to use either RADIUS or RadSec. When the option to Enable RadSec is selected on the NAD, Policy Manager will not accept communication from that device using RADIUS, RADIUS Accounting, or RADIUS Dynamic Authorization ports.

Section "FCS_IPSEC_EXT.1" in the **Admin Guide** details the basic information to establish IPsec tunnels which can also be used to protect RADIUS communication. If ports are restricted to RADIUS, ensure that RADIUS Accounting and RADIUS Dynamic Authorization are also allowed to pass through the IPsec tunnel to comply with CC evaluation configuration.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to update the ClearPass Server RADIUS and HTTPS certificates by installing certificates that are signed by a trusted certificate authority (CA). This is done via Administration -> Certificates -> Trust List in the Web UI.

The "FIA_X509_EXT.3.1" section of the **Admin Guide** provides details for creating a Certificate Signing Request including the following information fields to enter in the request: Common Name, Organization, Organizational

Unit and Country. This is consistent with the claims for this requirement in the ST. A CSR can be generated of a selected type. The default will be RADIUS/EAP Server Certificate. Other valid selections include HTTPS Server Certificate, RadSec Server Certificate, and Database Server Certificate.

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

Test 1: The evaluator shall send a RADIUS Access-Request containing an encapsulated EAP-response message of type Identity, specifying a valid user account, a service for which the user is authorized, and containing all information required to authenticate the user. The evaluator shall verify that the TOE returns an Access-Challenge, and that the MD5 hash of the concatenated Code + ID + Length + Request Authenticator of the Access-Request + Attributes + Secret matches the response authenticator.

Test 1: The evaluator sent the TOE a RADIUS Access-Request from an eapol client device which was configured to check the MD5 Message-Authenticator and to report it as either Valid or Invalid. TOE responded with an Access-Challenge and indicated that the MD5 Message-Authenticator was found to be valid and the connection was successful.

2.3 Cryptographic support (FCS)

2.3.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.3.1.1 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.

Table 6-1 in Section 6.3 of the ST identifies the TOE's cryptographic functions including cryptographic algorithms and associated key sizes. The FCS_CKM.1 requirement specifies the RSA, ECC, FFC schemes and FFC Schemes using Diffie-Hellman. Table 6-1 identifies the RSA Digital Signature Algorithm (rDSA) with a key size of 2048 bits or greater, the Elliptic Curve Digital Signature Algorithm (ECDSA) with NIST Curve sizes 256 and 384 bits and the FFC scheme using cryptographic key sizes of 2048 bits or greater. The TOE also supports key establishment using Diffie Hellman group 14 that meets Section 3 of RFC 3526. Table 6-2 in Section 6.3 further identifies where each of these schemes is used. IPsec, TLS, SSH and RADSEC use these schemes to create trusted communication channels for administration and communication with syslog, NTP and authentication services.

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section "FMT_SMR.2.3" in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The ClearPass console access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides instructions for importing a new certificate or for generating a new Certificate Signing Request (CSR). The type of certificate (RSA or ECDSA) used will influence which ciphers are available later. For example, RSA certificates will not be able to perform ECDSA based ciphers, so those encryption options will automatically be disabled.

The following is a list of all the allowed hash and encryption types that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode:

- Encryption: RSA
  - Size: 2048-bit, 3072-bit, or 4096-bit
  - Hash: SHA1, SHA256, SHA384, or SHA512
- Encryption: ECDSA
  - Size: NIST/SECG curve over 384-bit prime field
  - Hash: SHA1, SHA256, SHA384, or SHA512

Attempts to generate a CSR or load a certificate with sizes below the specified thresholds will fail. The type of key will be used to automatically determine the available cipher suites available.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. ClearPass uses its HTTPS certificate for IPSec. The Create IPsec Tunnel dialog box provides a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Type. The other cryptographic parameters to be configured are IPSec Mode, IKE Version, PRF, Encryption Algorithm, Hash Algorithm, Diffie Hellman Group, IKE Lifetime, Lifetime and Peer Certificate Subject DN. The encryption algorithms and hash algorithms available can be selected only one time and applied across the Security Association (SA).

Section "FCS_IPSEC_EXT.1.11" in the **Admin Guide** states that Diffie Hellman (DH) Groups are limited to group 14, 19, and 20.

Section “FCS_TLSS_EXT.2.1” in the **Admin Guide** lists the available TLS cipher suites consistent with those identified in FCS_TLSS_EXT.2, FCS_EAP-TLS_EXT.1 and FCS_RADSEC_EXT.1.

The “FCS_SSHS_EXT.1.2” section in the **Admin Guide** provides instructions for configuring SSH public key authentication using ssh-rsa and ecdsa-sha2-nistp256. This is consistent with the public key algorithms specified in FCS_SSHS_EXT.1.5 in the ST.

Section “FCS_EAP-TLS_EXT.1” in the **Admin Guide** specifies the cipher suites that are not available for RADIUS sessions under any circumstance.

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 the table “TOE CAVP Certificates” in Section 1.1.2.

2.3.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.3.2.1 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)

In section 6.3 of the ST, three key establishment schemes are identified in Table 6-1, RSA based key establishment, Elliptic Curve based key establishment and Finite field based key establishment schemes. The FCS_CKM.1 requirement specifies the RSA, ECC, FFC schemes and FFC Schemes using Diffie-Hellman. Table 6-1 identifies the RSA Digital Signature Algorithm (rDSA) with a key size of 2048 bits or greater, the Elliptic Curve Digital Signature Algorithm (ECDSA) with NIST Curve sizes P-256 and P-384 and the FFC scheme using cryptographic key sizes of 2048 bits or greater. The TOE generally fulfills all of the NIST SP 800-56A and Section 7.2 of RFC 3447, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1" requirements without extensions. For finite-field based key establishment, the TOE implements the following sections of SP 800-56A: 5.6 and all subsections. The TOE also supports key establishment using Diffie-Hellman group 14 that meets Section 3 of RFC 3526.

Table 6-2 in Section 6.3 further identifies where each of these schemes is used. IPsec, TLS, SSH and RADSEC use these schemes to create trusted communication channels for administration and communication with syslog, NTP and authentication services.

Security Function	Communication Type	Key Establishment Methods
Administration	TLS	RSA Schemes ECC Schemes FFC Schemes DH-14
Administration	SSH	ECC Schemes FFC Schemes DH-14

Trusted Channels for Syslog, NTP, Authentication Services	IPsec	ECC Schemes FFC Schemes DH-14
Trusted Channels for Authentication Services	RADSEC	RSA Schemes ECC Schemes FFC Schemes DH-14

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section "FMT_SMR.2.3" in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The ClearPass console access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides instructions for importing a new certificate or for generating a new Certificate Signing Request (CSR). The type of certificate (RSA or ECDSA) used will influence which ciphers are available later. For example, RSA certificates will not be able to perform ECDSA based ciphers, so those encryption options will automatically be disabled.

The following is a list of all the allowed hash and encryption types that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode:

- Encryption: RSA
  - Size: 2048-bit, 3072-bit, or 4096-bit
  - Hash: SHA1, SHA256, SHA384, or SHA512
- Encryption: ECDSA
  - Size: NIST/SECG curve over 384-bit prime field
  - Hash: SHA1, SHA256, SHA384, or SHA512

Attempts to generate a CSR or load a certificate with sizes below the specified thresholds will fail. The type of key will be used to automatically determine the available cipher suites available.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. ClearPass uses its HTTPS certificate for IPSec. The Create IPsec Tunnel dialog box provides a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Type. The other

cryptographic parameters to be configured are IPSec Mode, IKE Version, PRF, Encryption Algorithm, Hash Algorithm, Diffie Hellman Group, IKE Lifetime, Lifetime and Peer Certificate Subject DN. The encryption algorithms and hash algorithms available can be selected only one time and applied across the Security Association (SA).

Section "FCS_IPSEC_EXT.1.11" in the **Admin Guide** states that Diffie Hellman (DH) Groups are limited to group 14, 19, and 20.

Section "FCS_TLSS_EXT.2.1" in the **Admin Guide** lists the available TLS cipher suites consistent with those identified in FCS_TLSS_EXT.2, FCS_EAP-TLS_EXT.1 and FCS_RADSEC_EXT.1.

The "FCS_SSHS_EXT.1.2" section in the **Admin Guide** provides instructions for configuring SSH public key authentication using ssh-rsa and ecdsa-sha2-nistp256. This is consistent with the public key algorithms specified in FCS_SSHS_EXT.1.5 in the ST.

Section "FCS_EAP-TLS_EXT.1" in the **Admin Guide** specifies the cipher suites that are not available for RADIUS sessions under any circumstance.

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 the table "TOE CAVP Certificates" in Section 1.1.2.

2.3.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.3.3.1 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 6.3 in the ST identifies the Critical Security Parameters and keys that are subject to key destruction and indicates where they are stored and how they are cleared. Zeroization occurs as follows: 1) when deleted from the encrypted drive, the previous value is overwritten once with zeroes; 2) when added or changed on the encrypted drive, any old value is overwritten completely with the new value; and, 3) the zeroization of values in RAM is achieved by overwriting once with zeroes. All operations on the encrypted drive and RAM utilize standard file system APIs or memory management APIs.

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

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

Section “FCS_CKM.4 – Cryptographic Key Destruction” in the **Admin Guide** states that cryptographic key destruction is performed automatically. There are no administrator requirements to meet this requirement. There are no circumstances that do not strictly conform to the key destruction requirement and there are no situations where key destruction may be delayed at the physical layer.

Component Testing Assurance Activities: None Defined

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

2.3.4.1 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.

Table 6-1 in Section 6.3 of the ST identifies the TOE performs encryption and decryption using AES in CBC, CTR, and GCM mode with key sizes of either 128 or 256. The TOE supports IPsec for both transport and tunnel mode. For ESP encryption and the encrypted payload in IKEv1 the TOE supports 128 and 256-bit AES-CBC. For ESP encryption and the encrypted payload in IKEv2, the TOE supports 128 and 256-bit AES-CBC or 128 and 256-bit AES_GCM. The TOE supports SSHv2 with aes128-cbc, aes256-cbc, aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com encryption algorithms. The TOE supports TLSv1.2 with AES (CBC and GCM) 128 or 256-bit ciphers.

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section "FMT_SMR.2.3" in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The ClearPass console access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides instructions for importing a new certificate or for generating a new Certificate Signing Request (CSR). The type of certificate (RSA or ECDSA) used will influence which ciphers are available later. For example, RSA certificates will not be able to perform ECDSA based ciphers, so those encryption options will automatically be disabled.

The following is a list of all the allowed hash and encryption types that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode:

Encryption: RSA

Size: 2048-bit, 3072-bit, or 4096-bit

Hash: SHA1, SHA256, SHA384, or SHA512

Encryption: ECDSA

Size: NIST/SECG curve over 384-bit prime field

Hash: SHA1, SHA256, SHA384, or SHA512

Attempts to generate a CSR or load a certificate with sizes below the specified thresholds will fail. The type of key will be used to automatically determine the available cipher suites available.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. ClearPass uses its HTTPS certificate for IPSec. The Create IPsec Tunnel dialog box provides a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Type. The other cryptographic parameters to be configured are IPSec Mode, IKE Version, PRF, Encryption Algorithm, Hash Algorithm, Diffie Hellman Group, IKE Lifetime, Lifetime and Peer Certificate Subject DN. The encryption algorithms and hash algorithms available can be selected only one time and applied across the Security Association (SA).

Section "FCS_TLSS_EXT.2.1" in the **Admin Guide** lists the available TLS cipher suites consistent with those identified in FCS_TLSS_EXT.2, FCS_EAP-TLS_EXT.1 and FCS_RADSEC_EXT.1.

The "FCS_SSHS_EXT.1.4" section in the **Admin Guide** provides instructions for configuring SSH transport encryption algorithms.

Section "FCS_EAP-TLS_EXT.1" in the **Admin Guide** specifies the cipher suites that are not available for RADIUS sessions under any circumstance.

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 the table "TOE CAVP Certificates" in Section 1.1.2.

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

2.3.5.1 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.

Table 6-1 in Section 6.3 of the ST specifies the hash functions: SHA-1, SHA-256, SHA-384 and SHA-512 with message digest sizes 160, 256, 384 and 512 bits.

Section 6.3 of the ST states that the TOE's TLS implementations (TLS, EAP-TLS, RADSEC) support TLSv1.2 with AES (CBC and GCM) 128- or 256-bit ciphers, in conjunction with SHA-1, SHA-256, and SHA-384 using RSA and ECDSA for authentication. The TOE supports SSHv2 with aes128-cbc, aes256-cbc, aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com encryption algorithms, in conjunction with HMAC-SHA-1, HMAC-SHA2-256 and HMAC-SHA2-512 for data integrity and the following key exchange methods: diffie-hellman-group14-sha1 and ecdh-sha2-nistp256. The TOE also supports NTP and can authenticate an NTP server using a SHA1 key or can utilize NTP within an authenticated IPsec tunnel.

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides a list of all the allowed hash and encryption types that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode:

Encryption: RSA

Size: 2048-bit, 3072-bit, or 4096-bit

Hash: SHA1, SHA256, SHA384, or SHA512

Encryption: ECDSA

Size: NIST/SECG curve over 384-bit prime field

Hash: SHA1, SHA256, SHA384, or SHA512

Attempts to generate a CSR or load a certificate with sizes below the specified thresholds will fail.

Section "FMT_SMR.2.3" in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The ClearPass console access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The available TLS cipher suites are listed in section "FCS_TLSS_EXT.2.1" in the **Admin Guide**. These ciphersuites contain the hash algorithms identified in the ST.

Section FCS_SSHS_EXT.1.2 and section FCS_SSHS_EXT.1.5 in the **Admin Guide** identify the SSH public key authentication keys supported by the TOE which are ssh-rsa and ecdsa-sha2-nistp256. No administrator settings are available to configure.

Section FCS_SSHS_EXT.1.6 in the **Admin Guide** identifies the data integrity algorithms used for the SSH transport implementation which are: hmac-sha1, hmac-sha2-256, or hmac-sha2-512 MAC algorithms. No administrator settings are available to configure.

Section FCS_SSHS_EXT.1.7 in the **Admin Guide** identifies the SSH key exchange methods available in the TOE which are diffie-hellman-group14-sha1 and ecdh-sha2-nistp256. No administrator settings are available to configure.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. The Create IPsec Tunnel dialog box provides a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Method. The encryption algorithms and hash algorithms

available can be selected only one time and applied across the Security Association (SA). ClearPass uses it HTTPS certificate for IPSec.

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 the table "TOE CAVP Certificates" in Section 1.1.2.

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

2.3.6.1 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.

Table 6-1 in Section 6.3 of the ST specifies the keyed hash functions: HMAC-SHA-1 (block size 512 bits, key and digest size 160 bits), HMAC-SHA-256 (block size 512 bits, key and digest size 256 bits), HMAC-SHA-384 (block size 1024 bits, key and digest size 384 bits) and HMAC-SHA-512 (block size 1024 bits, key and digest size 512 bits).

Section 6.3 of the ST indicates that the TOE supports IPsec for both transport and tunnel mode and supports HMAC-SHA1, HMAC-SHA-256, HMAC-SHA384 and HMAC-SHA512 for keyed hashing. The TOE supports SSHv2 with aes128-cbc, aes256-cbc, aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com encryption algorithms, in conjunction with HMAC-SHA-1, HMAC-SHA2-256 and HMAC-SHA2-512 for data integrity and the following key exchange methods: diffie-hellman-group14-sha1 and ecdh-sha2-nistp256.

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides a list of all the allowed hash and encryption types that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode:

Encryption: RSA

Size: 2048-bit, 3072-bit, or 4096-bit

Hash: SHA1, SHA256, SHA384, or SHA512

Encryption: ECDSA

Size: NIST/SECG curve over 384-bit prime field

Hash: SHA1, SHA256, SHA384, or SHA512

Attempts to generate a CSR or load a certificate with sizes below the specified thresholds will fail.

Section "FMT_SMR.2.3" in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The ClearPass console access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The available TLS cipher suites are listed in section "FCS_TLSS_EXT.2.1" in the **Admin Guide**. These ciphersuites contain the hash algorithms identified in the ST.

Section FCS_SSHS_EXT.1.2 and section FCS_SSHS_EXT.1.5 in the **Admin Guide** identify the SSH public key authentication keys supported by the TOE which are ssh-rsa and ecdsa-sha2-nistp256. No administrator settings are available to configure.

Section FCS_SSHS_EXT.1.6 in the **Admin Guide** identifies the data integrity algorithms used for the SSH transport implementation which are: hmac-sha1, hmac-sha2-256, or hmac-sha2-512 MAC algorithms. No administrator settings are available to configure.

Section FCS_SSHS_EXT.1.7 in the **Admin Guide** identifies the SSH key exchange methods available in the TOE which are diffie-hellman-group14-sha1 and ecdh-sha2-nistp256. No administrator settings are available to configure.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. The Create IPsec Tunnel dialog box provides a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Method. The encryption algorithms and hash algorithms available can be selected only one time and applied across the Security Association (SA). ClearPass uses it HTTPS certificate for IPSec.

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 the table "TOE CAVP Certificates" in Section 1.1.2.

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

2.3.7.1 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 6.3 of the ST states the TOE supports the use of RSA with 2048 bit key sizes, and ECDSA with P-256 and P-384 curves for cryptographic signatures.

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 "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to update the ClearPass Server RADIUS and HTTPS certificates by installing certificates that are signed by a trusted certificate authority (CA). This is done via Administration -> Certificates -> Trust List in the Web UI.

Section "FIA_X509_EXT.3.1 in the **Admin Guide** provides instructions for generating a Certificate Signing Request (CSR) on ClearPass.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. The *Create IPsec Tunnel* dialog box provides a field to enter the remote server IP address and a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Method. The encryption algorithms and hash algorithms available can be selected only one time and applied across the Security Association (SA). ClearPass uses it HTTPS certificate for IPSec.

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 the table “TOE CAVP Certificates” in Section 1.1.2.

2.3.8 Extended: Extensible Authentication Protocol - Transport Layer Security (EAP-TLS) (AUTHSRVEP10:FCS_EAPTLS_EXT.1)

2.3.8.1 AUTHSRVEP10:FCS_EAPTLS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 AUTHSRVEP10:FCS_EAPTLS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.3 AUTHSRVEP10:FCS_EAPTLS_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.4 AUTHSRVEP10:FCS_EAPTLS_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.5 AUTHSRVEP10:FCS_EAPTLS_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.6 AUTHSRVEP10:FCS_EAPTLS_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that optional characteristics (e.g., extensions supported) are specified as well as the

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

Section 6.3 of the ST states that the TOE supports TLSv1.2 with AES (CBC and GCM) 128- or 256-bit ciphers, in conjunction with SHA-1, SHA-256, SHA-384 and RSA and ECDSA. Any other SSL/TLS versions are not supported by the TOE and such connection attempts will be rejected. No optional extensions are identified.

The TOE implements RADIUS to encapsulate EAP messages to and from associated NAS devices in accordance with RFCs 2865, 2869, and 3579. Furthermore, the TOE supports only EAP-TLS to support mutual authentication in accordance with RFC 5216. The EAP-TLS authentication method provides Compare Distinguished Name (DN) or Compare Common Name (CN) and Compare Subject Alternate Name (SAN) options. When these options are selected, RADIUS authentication request will be rejected if User Name does not match the DN or CN or SAN field in the certificate depending on the option configured. For EAP-TLS, the TOE supports the evaluated ciphersuites listed in the AUTHRVEP10:FCS_EAP-TLS_EXT.1 requirement which are as follows:

- TLS_RSA_WITH_AES_128_CBC_SHA
- *TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,*
- *TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,*
- *TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256 as defined in RFC 5246,*
- *TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,*
- *TLS_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*

Component Guidance Assurance Activities: The evaluator shall check that the operational guidance contains instructions for the administrator to configure the list of Certificate Authorities that are allowed to sign certificates used by the authentication server that will be accepted by the TOE in the EAP-TLS exchange, and instructions on how to specify the algorithm suites that will be proposed and accepted by the TOE during the EAP-TLS exchange. The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

Section “FCS_CKM.1 – Enable FIPS 140-2 Mode” and section “Enable Common Criteria Mode” in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section “FMT_SMR.2.3” in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The available cipher suites are listed in section “FCS_TLSS_EXT.2.1” in the **Admin Guide**. These ciphersuites are consistent with those identified in the ST.

Section FCS_TLSS_EXT.1.2 in the **Admin Guide** states that in Common Criteria deployments, the use of older SSL based protocols are not available and TLSv1.0 and TLSv1.1 are disabled by default. This can be verified in the Web UI. provides instructions for disabling TLS v1.0. This disables TLSv1.0 for HTTPS connections as well as all RADIUS related connections including EAP-TLS. On the General tab, the values for Disable TLSv1.0 support and Disable TLSv1.1 support are both set to All. This will prevent TLS versions prior to v1.2 from use in any component (such as RADIUS, RadSec, or WebUI). This setting cannot be modified in CC operating mode.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to install certificates that are signed by a trusted certificate authority (CA) via Administration -> Certificates -> Trust List in the Web UI. It also describes how to update the ClearPass Server RADIUS and HTTPS certificates.

Section "FTP_ITC.1" in the **Admin Guide** describes how to configure the ClearPass RADIUS service via the Web UI. The RADIUS service is configured via the Service tab in Configuration > Services, while the EAP-TLS authentication method is configured via the Authentication tab.

Section "FCS_EAP-TLS_EXT.1" in the **Admin Guide** states that when operating in Common Criteria mode, ClearPass will only use the cipher suites specified in section FCS_TLSS_EXT.2.1. TLS_ECDSA ciphers will not be used without an ECDSA key available for RADIUS. The following cipher suites are not available for RADIUS sessions under any circumstance:

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS-ECDHE-RSA-AES128-GCM-SHA256
TLS_ECDHE-RSA-AES256-GCM-SHA384

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

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 EAP session. It is sufficient to observe (on the wire) 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 in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The following test is repeated for each supported certificate signing algorithm supported. The evaluator shall attempt to establish the connection such that the client certificate contains the Client Authentication purpose in the extended KeyUsage field and the Key Agreement bit is set in the KeyUsage field and verify that a connection is established. The evaluator will then verify that connection is not established with an otherwise valid client certificate that lacks the Client Authentication purpose in the extended KeyUsage field.

Test 3: The evaluator shall follow the administrative guidance to configure the list of ciphersuites to be proposed during EAP-TLS negotiations that is limited to only those specified by the first element of this component. The evaluator shall have the EAP-TLS client propose a set of ciphersuites and show that the TOE will only negotiate the configured ciphers and ignore any others when proposed by a client. If the initial list is not a subset of the total set

of ciphersuites proposed by the client, the evaluator shall repeat the test specifying a proper subset of the ciphersuites used in the initial test. (TD0171 applied)

Test 1: The evaluator successfully established an EAP TLS connection between the TOE and an eapol client with all claimed ciphersuites.

Test 2: The evaluator sent a Radius Access-Request message from an eapol client configured to use a client certificate with the Client Authentication purpose set and the Key Agreement bit set in the extended KeyUsage field. The Access-Request was accepted and the connection was successful.

The evaluator then sent a Radius Access-Request message from an eapol client configured to use a client certificate with no Client Authentication purpose set in the extended KeyUsage field. The Access-Request was rejected and the connection failed.

Test 3: In conjunction with Test 1, the evaluator confirmed that the TOE would only negotiate configured ciphers and would reject all others.

2.3.9 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.3.9.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.3 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 6.3 of the ST states that the TOE implements HTTPS using TLS and is compliant with RFC 2818. The TOE requires the peer to initiate the connection and the TOE can be configured to require mutual authentication and when so configured requires a valid certificate to be provided by the peer. The TOE will not establish a connection when an invalid certificate is presented.

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.

Section "Passwords and Accounts" of the **Admin Guide** describes how to setup an administrator account at initial setup to access the WebUI

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides instructions for importing a new certificate or for generating a new Certificate Signing Request (CSR) which can be used for the HTTP Server (WebUI) The type of certificate (RSA or ECDSA) used will influence which ciphers are available later.

Section "FCS_TLSS_EXT.2.2 describes how to configure the HTTPS server for mutual authentication.

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.

This testing was demonstrated as part of the testing for FCS_TLSS_EXT.2 and FIA_X509_EXT.1/Rev where the evaluator started a packet capture and proceeded to log into the Web UI using HTTPS. A successful connection to the TOE was established. The evaluator tested this connection with both valid certificates and certificates not having a valid certification path used during the TLS connection.

2.3.10 IPsec Protocol - per TD0633 (NDcPP22e:FCS_IPSEC_EXT.1)

2.3.10.1 NDcPP22e:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in

RFC 4301.

As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Section 6.3 of the ST states that the TOE supports the definition of “IPsec Traffic Selector Rules”. The default behavior for IPsec rules is to encrypt all traffic between ClearPass and VPN peer. Traffic can be separated on a per-port and/or per-protocol level for encrypt, bypass, or drop actions. When implementing IKEv1, only one (1) rule of each type may be created. When implementing IKEv2, a maximum of ten (10) rules may be created for each IPsec tunnel.

The actions associated with each rule type are:

- **Encrypt Rules** - All packets matching these rules will be encrypted through the IPsec tunnel. When no subordinate actions are specified, this is the default for all traffic between hosts.
- **Bypass Rules** - All packets matching these rules will bypass the IPsec tunnel and flow to the remote peer outside of the VPN. This is commonly known as traffic “in the clear”, even though it may already be encrypted. When using bypass rules, both peers must be configured to bypass the selected traffic or the remote end will not appropriately process the packets.
- **Drop Rules** - All packets matching these rules will be dropped.
- **Final Rule** - An implicit rule is created with all IPsec traffic selection that will drop any traffic not processed. This rule will create a behavior where all traffic that should be encrypted or dropped between peers will always be blocked when the VPN is inactive. Bypass traffic is unaffected by tunnel status.

The defined IPsec rules are processed using both order and specificity. Order is established beginning by rule position starting with the first rule and descending within a rule group. Specificity is established based on the exactness of a rule to match against. Rules with specific ports and protocols will be evaluated prior to more general rules that apply to all ports or protocols prior to rules that catch “any” traffic.

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases â€“ a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

Section "FCS_IPSEC_EXT.1" in the **Admin Guide** provides instructions for using the *Create IPSec Tunnel* dialog box in the Web UI to configure the IPSec SPD including configuring the rules for processing a packet via the 'Traffic Selector" tab. Without configuration, the default is to encrypt all traffic (protocol and port) between the two hosts. The SPD is further described in Appendix B.

Appendix B in the **Admin Guide** indicates that traffic can be separated on a per port and/or per protocol level for Encrypt, Bypass or Drop Actions. There is also an implicit Final Rule which will drop any outbound traffic that is not processed. IPsec rules are processed using both order and specificity. Order is established beginning with rule position #1 and descending within a rule group. Specificity is established based on the exactness of a rule to match against. Rules with specific ports and protocols will be evaluated prior to more general rules that apply to all ports or protocols prior to rules that catch "any" traffic.

Testing Assurance Activities: The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

a) Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behaviour: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

b) Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and guidance documentation. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation.

Test 1 – The evaluator created rules for each type of SPD policy. The evaluator then verified the rules by establishing an IPsec tunnel and successfully establishing an administrator connection. It was observed that the Bypass, Protect and Discard rules for IPsec traffic were successfully enforced against the traffic going through the

IPsec tunnel. The evaluator also tested the Default Deny functionality and used the packet captures to verify and confirm. Overall the following scenarios were covered:

--Bypass, Protect and Discard rules
--Implicit and Explicit Discard/Drop
-- specificity of rules (e.g. rules with specific ports and protocols)

Test 2: FCS_IPSEC_EXT.1.1-t1 above included tests which cover most scenarios for packet processing based on the TOE's SPD. Specifically, FCS_IPSEC_EXT.1.1-t1 covered the following:

--Bypass, Protect and Discard rules
--Implicit and Explicit Discard/Drop
-- specificity of rules (e.g. rules with specific ports and protocols)

2.3.10.2 NDcPP22e:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a 'TOE create'' final entry that discards packets that do not match any previous entries). The evaluator sends the packet and observes that the packet was dropped.

The evaluator performed this test in conjunction with FCS_IPSEC_EXT.1.1. The evaluator tested both the implicit Default Deny rule and the explicit Discard rule. In both cases, the traffic was discarded as expected.

2.3.10.3 NDcPP22e:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: The evaluator checks the TSS to ensure it states that the VPN can be established to operate in transport mode and/or tunnel mode (as identified in FCS_IPSEC_EXT.1.3).

Section 6.3 of the ST states the TOE supports IPsec for both tunnel and transport mode. This matches the FCS_IPSEC_EXT.1.3 requirement.

Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions on how to configure the connection in each mode selected.

The "FCS_IPSEC_EXT.1" and section in the **Admin Guide** describes how to configure the IPSec tunnel including how to specify either transport or tunnel mode. IPSec tunnels are configured via Administration » Server Manager » Server Configuration – Network. The *Create IPSec Tunnel* dialog box contains a drop down menu for selecting the IPSec mode – tunnel or transport.

Section "FCS_IPSEC_EXT.1.3" in the **Admin Guide** states that IPsec VPNs may be configured to use either Transport or Tunnel by selecting the IPsec Mode. Tunnel mode is the default IPsec Mode.

Testing Assurance Activities: The evaluator shall perform the following test(s) based on the selections chosen:

Test 1: If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

Test 2: If transport mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

Test 1 and Test 2: For this test, the evaluator alternately configured a test peer to require only tunnel mode or only transport mode. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful in both modes which are supported by the TOE. This test was iterated for both IKEv1 and IKEv2.

2.3.10.4 NDcPP22e:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in

FCS_COP.1(4)/KeyedHash Cryptographic Operations (for keyed-hash message authentication) and if the SHA-based HMAC function truncated output is utilized it must also be described.

Section 6.3 in the ST states that the TOE supports IPsec for both transport and tunnel mode. For ESP encryption and the encrypted payload in IKEv1 the TOE supports 128 and 256-bit AES-CBC. For ESP encryption and the encrypted payload in IKEv2, the TOE supports 128 and 256-bit AES-CBC or 128 and 256-bit AES_GCM. Similarly, HMAC-SHA1, HMAC-SHA-256 and HMAC-SHA384 are supported for keyed hashing.

Guidance Assurance Activities: The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure the IPSec tunnel and describes all of the parameters that can be configured in the *Create IPsec Tunnel* dialog box. This includes configuration of the claimed encryption algorithms which are identified and listed and are consistent with those algorithms claimed in the ST.

Testing Assurance Activities: The evaluator shall configure the TOE as indicated in the guidance documentation configuring the TOE to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

This test was iterated for both IKEv1 and IKEv2.

For IKEv1, the evaluator alternately configured a test peer to accept only the claimed ESP ciphers AES-CBC-128 and AES-CBC-256. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful.

For IKEv2, the evaluator alternately configured a test peer to accept the claimed ESP ciphers AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful.

2.3.10.5 NDcPP22e:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

For IKEv1 implementations, the evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

Section 6.3 of the ST indicates that both IKEv1 and IKEv2 are supported and notes that aggressive mode is not used with IKEv1, only main mode is supported.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and how to configure the TOE to perform NAT traversal for the following test (if selected).

If the IKEv1 Phase 1 mode requires configuration of the TOE prior to its operation, the evaluator shall check the guidance documentation to ensure that instructions for this configuration are contained within that guidance.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** explains how to configure the parameters for both IKEv1 and IKEv2. To configure the IPsec tunnel the user is instructed to select Administration » Server Manager » Server Configuration – Network. On the Network tab, the user should click on 'Create IPSec Tunnel'. In the *Create IPsec Tunnel* dialog box, there is a drop down menu to select IKEv1 or IKEv2. The "FCS_IPSEC_EXT.1.5" section indicates that support for NAT traversal is included in IPsec.

Testing Assurance Activities: Tests are performed in conjunction with the other IPsec evaluation activities.

a) Test 1: If IKEv1 is selected, the evaluator shall configure the TOE as indicated in the guidance documentation and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.

b) Test 2: If NAT traversal is selected within the IKEv2 selection, the evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

Test 1: The evaluator alternately configured a test peer to use Main Mode and Aggressive Mode for IKEv1. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was only accepted by the TOE in Main Mode.

Test 2: The evaluator configured the connection between the TOE and a test peer so that NAT was required. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful regardless of NAT routing.

2.3.10.6 NDcPP22e:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms chosen in the selection of the requirement are included in the TSS discussion.

Section 6.3 of the ST states that the TOE supports IPsec for both transport and tunnel mode. For ESP encryption and the encrypted payload in IKEv1 the TOE supports 128 and 256-bit AES-CBC. For ESP encryption and the encrypted payload in IKEv2, the TOE supports 128 and 256-bit AES-CBC or 128 and 256-bit AES_GCM. This is consistent with the requirements.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure the IPsec tunnel and describes all of the parameters that can be configured in the 'Create IPsec Tunnel' dialog box. This includes configuration of the encryption algorithms which include AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256.

Testing Assurance Activities: The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

The evaluator alternately configured a test peer to accept only the claimed IKE ciphers. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to be successful only if the configured IKE cipher was supported on the TOE.

This test was iterated for both IKEv1 and IKEv2.

For IKEv1, the evaluator alternately configured a test peer to accept only the claimed IKE ciphers AES-CBC-128 and AES-CBC-256. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful.

For IKEv2, the evaluator alternately configured a test peer to accept the claimed IKE ciphers AES-CBC-128, AES-CBC-256, AES-GCM-128 and AES-GCM-256. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful.

2.3.10.7 NDcPP22e:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 1 SA lifetime and/or the IKEv2 SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Section 6.3 of the ST states that IKEv1 Phase 1 SA and IKEv2 SA lifetime limits can be configured to be up to 24 hours by a Security Administrator. This is consistent with the selection made in FCS_IPSEC_EXT.1.7. After SAs are established as part of a connection, each SA is renegotiated and re-established each time its configured lifetime is reached.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g. configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not

permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

The “FCS_IPSEC_EXT.1” and “FCS_IPSEC_EXT.1.7” sections in the **Admin Guide** describe how to configure the IPsec tunnel and describe all of the parameters that can be configured in the *Create IPsec Tunnel* dialog box. This includes configuration of the IKEv1 and IKEv2 SA lifetimes by setting the ‘IKE Lifetime’ and ‘Lifetime’ fields.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If 'number of bytes' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish a SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the Phase 1 SA lifetime on the TOE.

Test 1: Not applicable. Byte lifetime is not supported by the TOE.

Test 2: The evaluator configured the TOE to have a 24 hour IKE and 8 hour ESP limits and the test peer was configured to have 25 hour IKE and 9 hour ESP limits. The evaluator then successfully connected the IPsec VPN between the test peer and the TOE being tested and then waited for over 24 hours before terminating the test. Every hour, the date was injected into the test log, an ipsec status command was issued to show all the connections were still active, and each TOE device was pinged from the test peer. The evaluator verified that both lifetimes expired and were renegotiated prior to their respective lifetimes being reached. This test was repeated for both IKEv1 and IKEv2.

2.3.10.8 NDcPP22e:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 2 SA lifetime and/or the IKEv2 Child SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Section 6.3 of the ST states that IKEv1 Phase 2 SA and IKEv2 Child SA lifetime limits can be configured up to 8 hours by a Security Administrator. This is consistent with the selection made in FCS_IPSEC_EXT.1.8. After SAs are established as part of a connection, each SA is renegotiated and re-established each time its configured lifetime is reached.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g. configure a time value of 7h 45min to ensure the actual rekey is performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

The “FCS_IPSEC_EXT.1” and “FCS_IPSEC_EXT.1.7” sections in the **Admin Guide** describe how to configure the IPsec tunnel and describe all of the parameters that can be configured in the *Create IPsec Tunnel* dialog box. This includes configuration of the IKEv1 and IKEv2 SA lifetimes by setting the ‘IKE Lifetime’ and ‘Lifetime’ fields.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

Test 1: If 'number of bytes' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish a SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE.

Test 1: Not applicable. Byte lifetime is not supported by the TOE.

Test 2: The 8 hour phase 2 lifetime was demonstrated alongside the 24 hour phase 1 lifetime as part of testing in FCS_IPSEC_EXT.1.7, test 2.

2.3.10.9 NDcPP22e:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating 'x'. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of 'x' meets the stipulations in the requirement.

Section 6.3 in the ST states that the TOE generates the secret value x used in the IKEv1/IKEv2 Diffie-Hellman key exchange ('x' in gx mod p) using the FIPS validated RBG specified in FCS_RBG_EXT.1 and having possible lengths of 224, 256 or 384 bits (for DH Groups 14, 19, and 20, respectively). When a random number is needed for a nonce, the probability that a specific nonce value will be repeated during the life of a specific IPsec SA is less than 1 in 2^{112}, 2^{128}, or 2^{192}, corresponding to the respective DH group. For IKEv2, the nonces used in the IKE exchanges are generated by the TOE's random bit generator with lengths of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.10.10 NDcPP22e:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

Section 6.3 of the ST states that the TOE generates the secret value x used in the IKEv1/IKEv2 Diffie-Hellman key exchange ('x' in gx mod p) using the FIPS validated RBG specified in FCS_RBG_EXT.1 and having possible lengths of 224, 256 or 384 bits (for DH Groups 14, 19, and 20, respectively). When a random number is needed for a nonce,

the probability that a specific nonce value will be repeated during the life of a specific IPsec SA is less than 1 in 2^{112}, 2^{128}, or 2^{192} corresponding to the respective DH group. For IKEv2, the nonces used in the IKE exchanges are generated by the TOE's random bit generator with lengths of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

b) Test 2: If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

This test requirement is a TSS activity that has been addressed in the TSS assurance activity above.

2.3.10.11 NDcPP22e:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Section 6.3 in the ST states that DH groups 14, 19, and 20 are supported for both IKEv1 and IKEv2. The TOE selects the DH group by selecting the largest group configured by an administrator that is offered by the VPN gateway.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** explains how to configure the IPsec tunnel. The user is instructed to select Administration » Server Manager » Server Configuration – Network. On the Network tab, the user should click on 'Create IPsec Tunnel'. In the *Create IPsec Tunnel* dialog box, there is a drop down menu to select the dh group which includes the dh groups claimed in the ST: group 14, group 19 and group 20.

Section "FCS_IPSEC_EXT.1.11" in the **Admin Guide** further clarifies that the supported DH groups are limited to group 14, group 19 and group 20.

Testing Assurance Activities: For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

Test 1 - The evaluator made an IPsec connection to an IPsec peer using each of the claimed DH groups (14, 19, and 20). The evaluator confirmed via packet capture that the connections were successful for each claimed DH group. This test was iterated for both IKEv1 and IKEv2.

2.3.10.12 NDcPP22e:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

Section 6.3 of the ST states that the TOE supports IPsec for both transport and tunnel mode. For ESP encryption and the encrypted payload in IKEv1 the TOE supports 128 and 256-bit AES-CBC. For ESP encryption and the encrypted payload in IKEv2, the TOE supports 128 and 256-bit AES-CBC or 128 and 256-bit AES_GCM. Similarly, HMAC-SHA1, HMAC-SHA-256 and HMAC-SHA384 are supported for keyed hashing. When configuring ciphers, there is only one setting that applies to both phase 1 and phase 2, this ensures that the IKE and ESP ciphers are the same and hence have the same security strength.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator simply follows the guidance to configure the TOE to perform the following tests.

a) Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

b) Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

c) Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

d) Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

Test 1 – The evaluator alternately configured a test peer to accept only the claimed IKE hash algorithms. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connections were successful for each claimed IKE hash algorithm. This test was iterated for both IKEv1 and IKEv2. Note that the range of ciphersuites has been tested according to FCS_IPSEC_EXT.1.6.

Test 2 – The evaluator configured a test peer to use a 128-bit key size for IKE and 256-bit key size for ESP. The evaluator then attempted to connect the IPsec VPN between the test peer and confirmed that the connection was rejected by the TOE. This test was iterated for both IKEv1 and IKEv2.

Test 3- The evaluator alternately configured a test peer as follows: 1) Control test with all supported ciphers and hashes, 2) restrict the IKE cipher to 128-bit blowfish, 3) restrict the IKE hash to MD5, and 4) restrict the ESP cipher to 128-bit blowfish. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was only successful for the control case. This test was iterated for both IKEv1 and IKEv2.

Test 4 – This test has been performed as part of Test 3 above. The evaluator attempted to establish a connection with an unsupported ESP algorithm. The connection attempt failed.

2.3.10.13 NDcPP22e:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(2)/SigGen Cryptographic Operations (for cryptographic signature).

If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

In Section 6.3 of the ST, it states that RSA and ECDSA certificates are supported for both IKEv1 and IKEv2. The TOE supports both certificate and pre-shared key IPsec authentication. Section 6.4 states that in the case of pre-shared keys, the administrator types in and confirms the pre-shared key. The TOE does not claim to be able to generate pre-shared keys.

Guidance Assurance Activities: The evaluator ensures the guidance documentation describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

The evaluator shall check that the guidance documentation describes how pre-shared keys are to be generated and established. The description in the guidance documentation shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

The evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA and ensure a valid certificate for that CA is loaded into the TOE and marked 'trusted'.

The “FCS_IPSEC_EXT.1” section in the **Admin Guide** explains how to configure the IPsec tunnel for both Certificate and Preshared key authentication. To configure the IPsec tunnel the user is instructed to select Administration » Server Manager » Server Configuration – Network. On the Network tab, the user should click on ‘Create IPsec Tunnel’. In the *Create IPsec Tunnel* dialog box, there is a drop down menu to select the Authentication type as either certificate or preshared key.

Section FCS_IPSEC_EXT.1 in the **Admin Guide** further states that if Certificate is selected as the authentication type, a certificate issued by a public authority should be used. If “Certificate” is selected as the Authentication Type, when specifying the value of Peer Certificate Subject DN, the specified distinguished name must be an exact match to the certificate that the remote device is using. If this is not exactly matched, the tunnel will fail to negotiate. ClearPass will use its HTTPS certificate for IPsec identity, but the CA from the remote peer must also be included in the ClearPass trust list or validation will not occur. HTTPS certificates can be either RSA or ECDSA certificates.

Section “FIA_PSK_EXT.1” in the **Admin Guide** states that it is recommended to use any key of at least 22 characters. ClearPass supports PSK values of up to 128 character length. PSK values should make use of a mixture of password character types to maximize the entropy and minimize attack capabilities. The pre-shared key value is entered into the ‘IKE Shared Secret’ and ‘Verify IKE Shared Secret’ fields.

The “FIA_X509_EXT.1/Rev (Install Certificates)” section of the **Admin Guide** provides instructions for enabling a CA in the Certificate Trust List or importing a CA to the Certificate Trust List. It further describes how to import the ClearPass Server certificate and how to create a Certificate Signing Request. It also provides a list of all the allowed hash and encryption types that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode. These algorithms are consistent with those identified in the ST.

Testing Assurance Activities: For efficiency sake, the testing is combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1.

Test 1 – Pre-shared Key authentication was demonstrated throughout the other IPSec testing, an example would be FCS_IPSEC_EXT.1.3 where the evaluator configured the TOE to perform pre-shared key authentication and established a successful connection with a peer.

2.3.10.14 NDcPP22e:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN). If the TOE simultaneously supports the same identifier type in the CN and SAN, the TSS shall describe how the TOE prioritizes the comparisons (e.g. the result of comparison if CN matches but SAN does not). If the location (e.g. CN or SAN) of non-DN identifier types must explicitly be configured as part of the reference identifier, the TSS shall state this. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which

that type is compared to the peer's presented certificate, including what field(s) are compared and which fields take precedence in the comparison.

Section 6.3 of the ST states that when an IPsec connection is configured, the administrator can define the DN for the peer. When the connection is made, the configured DN is compared against that in the peer certificate and the connection succeeds only if they match exactly.

Guidance Assurance Activities: The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE does not guarantee unique identifiers, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

Sections “FCS_IPSEC_EXT.1” and “FCS_IPSEC_EXT.1.14” in the **Admin Guide** describe how to configure the expected DN for the connection. If “Certificate” is selected as the Authentication Type, when specifying the value of Peer Certificate Subject DN, the specified distinguished name must be an exact match to the certificate that the remote device is using. If this is not exactly matched, the tunnel will fail to negotiate. The peer certificate should be specified as stated in the client certificate beginning with the CN= field until the end of the DN is met. When applied to IPsec VPN configurations, the SAN extension in the certificate is not used to match against. ClearPass will use its HTTPS certificate for IPsec identity, but the CA from the remote peer must also be included in the ClearPass trust list or validation will not occur. HTTPS certificates can be either RSA or ECDSA certificates.

Testing Assurance Activities: In the context of the tests below, a valid certificate is a certificate that passes FIA_X509_EXT.1 validation checks but does not necessarily contain an authorized subject.

The evaluator shall perform the following tests:

Test 1: (conditional) For each CN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes CN checking over SAN (through explicit configuration of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the SAN so it contains an incorrect identifier of the correct type (e.g. the reference identifier on the TOE is example.com, the CN=example.com, and the SAN:FQDN=otherdomain.com) and verify that IKE authentication succeeds.

Test 2: (conditional) For each SAN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes SAN checking over CN (through explicit specification of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the CN so it contains an incorrect identifier formatted to be the same type (e.g. the reference identifier on the TOE is DNS-ID; identify certificate has an identifier in SAN with correct DNS-ID, CN with incorrect DNS-ID (and not a different type of identifier)) and verify that IKE authentication succeeds.

Test 3: (conditional) For each CN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with the CN so it contains the valid identifier followed by ''. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the evaluator shall configure the SAN so it matches the reference identifier.

b) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the CN without the '' and verify that IKE authentication fails.

Test 4: (conditional) For each SAN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with an incorrect identifier in the SAN. The evaluator shall configure a string representation of the correct identifier in the DN. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the addition/modification shall be to any non-CN field of the DN. Otherwise, the addition/modification shall be to the CN.

b) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the correct identifier (expected in the SAN) and verify that IKE authentication fails.

Test 5: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 6: (conditional) If the TOE supports DN identifier types, to demonstrate a bit-wise comparison of the DN, the evaluator shall create the following valid certificates and verify that the IKE authentication fails when each certificate is presented to the TOE:

a) Duplicate the CN field, so the otherwise authorized DN contains two identical CNs.

b) Append '' to a non-CN field of an otherwise authorized DN.

Test 1 – Not applicable. The TOE does not support CN identifier types.

Test 2 – Not applicable. The TOE does not support SAN identifier types.

Test 3 – Not applicable. The TOE does not support CN identifier types.

Test 4- Not applicable. The TOE does not support SAN identifier types.

Test 5 – The evaluator configured a test peer to first send an authentication certificate with an authorized DN and confirmed that the connection succeeded. This test was iterated with 4 variations: IKEv1:RSA, IKEv1:ECDSA, IKEv2:RSA and IKEv2:ECDSA.

Test 6a - The evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate but with a DN containing a duplicate CN. In each case, the evaluator attempted to establish an IPsec connection and confirmed that the TOE rejected the certificate containing a duplicate CN. This test was iterated for IKEv1:RSA, IKEv1:ECDSA, IKEv2:RSA and IKEv2:ECDSA.

Test 6b - The evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate in which the Organization field of the DN has a trailing null character (64) appended. In each case, the evaluator attempted to establish an IPsec connection, and confirmed that the TOE rejected the certificate containing a DN Organization containing an appended null character. This test was iterated for IKEv1:RSA, IKEv1:ECDSA, IKEv2:RSA and IKEv2:ECDSA.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.11 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1)

2.3.11.1 NDcPP22e:FCS_NTP_EXT.1.1

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

Section 6.3 of the ST states that the TOE supports NTPv4, while rejecting all broadcast and multicast time updates. The TOE can authenticate an NTP server using a SHA1 key or can utilize NTP within an authenticated IPsec tunnel. The TOE can be configured to identify as many as 5 NTP servers from which time update are accepted.

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

The “Configure System Time” section in the **Admin Guide** provides instructions for configuring the time manually or using the option ‘Synchronize time with NTP server’. NTP servers must support NTPv4 to work. The WebUI allows a minimum of one (1) NTP server to be used, but it is recommended to specify at least three (3) NTP servers.

The WebUI allows the specification of 1-5 NTP servers. When configuration is performed with NTP the communication between appliance and NTP server should be configured to a secure key and the SHA-1 hash algorithm to ensure the communication is not modified. When the date and/or time are modified, the system will restart services and require a re-login to the UI. The NTP service does not accept multicast or broadcast NTP information, there are no configuration options to change this behavior.

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

A successful sync using NTPv4 was demonstrated in FCS_NTP_EXT.1.2.

2.3.11.2 NDcPP22e:FCS_NTP_EXT.1.2

TSS Assurance Activities: None Defined

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

Assurance Activity Note:

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

The “Configure System Time” section in the **Admin Guide** provides instructions for configuring the time manually or using the option ‘Synchronize time with NTP server’. NTP servers must support NTPv4 to work. The WebUI allows a minimum of one (1) NTP server to be used, but it is recommended to specify at least three (3) NTP servers. The WebUI allows the specification of 1-5 NTP servers. When configuration is performed with NTP the communication between appliance and NTP server should be configured to a secure key and the SHA-1 hash algorithm to ensure the communication is not modified. When the date and/or time are modified, the system will restart services and require a re-login to the UI. The NTP service does not accept multicast or broadcast NTP information, there are no configuration options to change this behavior.

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

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

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

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

NTP over a trusted channel was demonstrated as part of NDcPP22:FTP_ITC.1.

The evaluator configured the NTP server to use an invalid MD5 NTP key, and the TOE to use a SHA1 key. The evaluator verified that the TOE did not synchronize with the time source and that the time did not change. The evaluator then configured the NTP server to use the valid SHA1 NTP key and verified that the time was synchronized correctly, and found from the packet capture that NTP version 4 and the SHA1 message digest algorithm were used.

2.3.11.3 NDcPP22e:FCS_NTP_EXT.1.3

TSS Assurance Activities: None Defined

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

The “Configure System Time” section in the **Admin Guide** states that the NTP service does not accept multicast or broadcast NTP information, there are no configuration options to change this behavior.

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

The evaluator configured the NTP server to support periodic time updates to broadcast address. The evaluator observed the time on the NTP server prior to the broadcast updates. The evaluator then verified that the NTP server sent broadcast packets and that the TOE timestamp did not change to the NTP server time above. This test was repeated with the NTP server configured to support periodic updates to multicast address and the results similarly indicate that the TOE timestamp did not change to the NTP server time when multicast packets were sent.

2.3.11.4 NDcPP22e:FCS_NTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

(TD0528 applied)

Test 1: The evaluator successfully configured 3 valid NTP servers on the TOE and attempted to sync the TOE to the NTP servers. The evaluator verified a sync was successful to one of the NTP servers.

Test 2: The evaluator successfully configured 3 NTP servers on the TOE and set up an unconfigured NTP server to broadcast directly to the TOE. The evaluator verified the TOE does not sync to the unconfigured broadcasting NTP server.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.12 Extended: RADIUS (AUTHSRVEP10:FCS_RADIUS_EXT.1)

2.3.12.1 AUTHSRVEP10:FCS_RADIUS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.12.2 AUTHSRVEP10:FCS_RADIUS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.12.3 AUTHSRVEP10:FCS_RADIUS_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 that RADIUS is specified as the protocol by which all communication between the TOE and the NAS is conducted. The evaluator shall examine the TSS to ensure that EAP is specified as the authentication protocol to be used between the TOE and the NAS, that TLS is the means of mutual authentication to be carried out over EAP, and that other authentication frameworks are disallowed. The evaluator shall check the description of the implementation of this protocol to ensure that RADIUS encapsulated EAP Message Authenticators conform to RFC 3579.

Section 6.2 of the ST states that the TOE implements the RADIUS protocol in order to provide network authentication services for RADIUS clients, but not for local administrator authentication. The TOE requires RADIUS encapsulated EAP Message Authenticators that conform to RFC 3579 and each Access-Request from a NAS must have the correct Message Authenticator so that the NAS can be determined to be authentic.

Section 6.3 of the ST further confirms that the TOE implements RADIUS to encapsulate EAP messages to and from associated NAS devices in accordance with RFCs 2865, 2869, and 3579. Furthermore, the TOE supports only EAP-TLS to support mutual authentication in accordance with RFC 5216. The EAP-TLS authentication method provides Compare Distinguished Name (DN) or Compare Common Name (CN) and Compare Subject Alternate Name (SAN) options. When these options are selected, RADIUS authentication request will be rejected if User Name does not match the DN or CN or SAN field in the certificate depending on the option configured.

Component Guidance Assurance Activities: The evaluator shall verify that the guidance contains all necessary instructions to configure RADIUS and encapsulated EAP-TLS on the TOE, in accordance with RFCs 2865, 2869, 3579, and 5216.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides instructions for installing certificates to the ClearPass Trust List as well as importing the ClearPass Server certificate. It also provides a list of all the allowed hash and encryption types (RSA and ECDSA) that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode.

Section "FTP_ITC.1" in the **Admin Guide** describes how to configure the RADIUS service via the Configuration -> Services menu in the Web UI. The authentication method is configured on the Authentication tab. To conform with the Common Criteria evaluated configuration, only the EAP-TLS authentication method may be used.

Section "Add Network Access Devices" in the **Admin Guide** describes how to add network access devices to the system before conducting RADIUS authentication events and configuring a RADIUS shared secret key/password. To comply with Common Criteria evaluated status, all RADIUS communication between the TOE and the NAD should be through an established IPsec tunnel.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPsec using the "Create IPsec Tunnel" interface in the Web UI between ClearPass and a remote device using the "Create IPsec Tunnel" dialog box in the Web UI. This dialog box identifies a field to enter the remote IP address of the syslog server that the TOE will be connecting to as well as fields for entering the encryption algorithms and the type of authentication that will be used.

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

Test 1: The evaluator shall send RADIUS access-requests with encapsulated EAP-response messages to the TOE, from a NAS with which the TOE shares a RADIUS pre-shared key, and verify that the TOE responds appropriately according to RFCs 2865 and 3579:

- The evaluator shall verify that the TOE returns either an access-reject or an access-reject with an encapsulated EAP-response of type NAK. (TD0171 applied)

- Access-requests containing encapsulated EAP-response messages and each of the following attributes: User-password, CHAP-password, CHAP-challenge, ARAP-password, password-retry, reply-message, error-cause. The evaluator shall verify that in each case, the TOE discards the request without responding.

- An access-request containing an encapsulated EAP-response message, but no message-authenticator attribute. The evaluator shall verify that the TOE discards the request without responding.

- An access-request containing an encapsulated EAP-response message of type MD5-challenge. The evaluator shall verify that the TOE responds with an access-reject or access-challenge message of type Nak or expanded Nak. (TD0171 applied)

- An access-request containing an encapsulated EAP-response message of type Identity, specifying a valid user account, a service for which the user is authorized, and containing all information required to authenticate the user.

o The evaluator shall verify that the TOE returns an access-challenge with an encapsulated EAP-TLS start packet; i.e. an EAP-request with EAP-type set to EAP-TLS, the start bit set, and no data.

o The evaluator shall go on to complete the TLS handshake, presenting valid, untrusted, expired, and revoked client certificates to the TOE, and verify that the handshake completes successfully only for valid certificates, and unsuccessfully otherwise,

o The evaluator shall verify that the TOE indicates a successful TLS handshake with an access-accept with encapsulated EAP-success packet. The evaluator shall verify that the TOE indicates an unsuccessful TLS handshake with an access-reject with encapsulated EAP-failure packet.

o During an otherwise successful handshake, the evaluator shall send an access-request with encapsulated EAP-response with EAP-type set to anything but EAP-TLS, and verify that the TOE returns an access-challenge with encapsulated EAP-request of type EAP-TLS, indicating error-cause: invalid EAP type error (ignored), an access-reject message, or silently discard the request. The evaluator shall verify that subsequent handshake steps complete normally. (TD0171 applied)

o During an otherwise successful handshake, the evaluator shall send five or less invalid EAP packets, and verify that the TOE returns an access-reject with encapsulated EAP-failure packet after receiving an invalid packet. If the number of packets are configurable, the evaluator must follow the instructions in the operational guidance to verify the ability to set this value to 5 or less. (TD0171 applied)

- An access-request containing an encapsulated EAP-response message of type Identity, specifying a valid user account, and a service for which the user is not authorized. The evaluator shall verify that the TOE returns an access-reject.

- An access-request containing an encapsulated EAP-response message of type Identity, specifying an invalid user account. The evaluator shall verify that the TOE returns an access-reject.

- An Access-Request whose length field is incorrect. The evaluator shall verify that the TOE discards the request without responding.

- An Access-Request whose code field is invalid. The evaluator shall verify that the TOE discards the request without responding.

- An Access-Request containing an encapsulated EAP-response message and a message-authenticator attribute that does not match the request. The evaluator shall verify that the TOE discards the request without responding.

Test 1: The evaluator sent a RADIUS access-request to the TOE which was modified so that the embedded EAP message was an EAP-request rather than an EAP-response. The request failed as expected.

Test 2: The evaluator used an EAP-TLS client to attempt to connect to the TOE where an otherwise correct RADIUS request is modified so that the embedded EAP message includes the following attributes one at a time (i.e., each in a subsequent attempt):

a) User-password
b) CHAP-password
c) CHAP-challenge
d) ARAP-password
e) password-retry
f) reply-message
g) error-cause

The evaluator verified that all of these unsupported EAP message attributes resulted in failed requests.

Test 3: The evaluator used an EAP-TLS client to attempt to connect to the TOE where an otherwise correct RADIUS request with an EAP-response does not contain a message authenticator. The request failed as expected.

Test 4: The evaluator used an EAP-TLS client to attempt to connect to the TOE where an otherwise correct RADIUS request with an EAP-response with its type changed to MD5-challenge. The request failed as expected.

Test 5: The evaluator used an EAP-TLS client to attempt to connect to the TOE with each of the following scenarios and verified that only the first scenario resulted in a successful connection. All other attempts failed.

- A correct RADIUS request and valid client certificate is used.

- A correct RADIUS request and untrusted client certificate is used.

- A correct RADIUS request and expired client certificate is used.

- A correct RADIUS request and revoked client certificate is used.

- An otherwise correct RADIUS request and valid client certificate is used, but the EAP-message type is changed to an unknown type.

- An otherwise correct RADIUS request and valid client certificate is used, but the EAP-message is corrupted (several times during a session).

Test 6: The evaluator used an EAP-TLS client to attempt to connect to the TOE where a correct RADIUS request that is requesting access to a service that is not allowed to an otherwise valid user. The request failed as expected.

Test 7: The evaluator used an EAP-TLS client to attempt to connect to the TOE where a correct RADIUS request that is requesting access for an invalid user. The request failed as expected.

Test 8: The evaluator used an EAP-TLS client to attempt to connect to the TOE where an otherwise correct RADIUS request has a modified (incorrect) length. The request failed as expected.

Test 9: The evaluator used an EAP-TLS client to attempt to connect to the TOE where an otherwise correct RADIUS request has a modified (invalid) access-request code. The request failed as expected.

Test 10: The evaluator used an EAP-TLS client to attempt to connect to the TOE where an otherwise correct RADIUS request has a modified (incorrect) message-authenticator. The request failed as expected.

2.3.13 Extended: RadSec (AUTHSRVEP10:FCS_RADSEC_EXT.1)

2.3.13.1 AUTHSRVEP10:FCS_RADSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.13.2 AUTHSRVEP10:FCS_RADSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.13.3 AUTHSRVEP10:FCS_RADSEC_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 description includes the use of RADIUS over TLS, as described in RFC 6614.

If the TOE supports X.509v3 certificates, the evaluator shall ensure that the TSS description includes the use of client-side certificates for TLS mutual authentication. The evaluator shall also verify that the TSS describes how the DN or SAN in the certificate is compared to the expected identifier.

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. The evaluator shall also verify that the TSS contains a description of the denial of old SSL and TLS versions.

(TD0459 applied)

Section 6.3 of the ST states that the TOE supports the use of RADIUS over TLS (compliant with RFC 6614) for use with specified network peers. The TOE RADSEC implementation supports authentication of a peer using x509 certificates only (use of preshared keys is not supported). The SAN or CN in the certificate presented by the peer must match the expected identifier for a RADSEC connection to be fully established. As with other TLS implementations in this product, for RADSEC the TOE supports only TLSv1.2, rejecting all earlier version of SSL and TLS. Additionally, the TOE supports the use of only the algorithms, hashes and key exchanges defined by the ciphersuites listed in the AUTHSRVEP10:FCS_RADSEC_EXT.1 requirement which are as follows:

TLS_RSA_WITH_AES_128_CBC_SHA]
TLS_RSA_WITH_AES_256_CBC_SHA,
TLS_RSA_WITH_AES_128_CBC_SHA256,
TLS_RSA_WITH_AES_256_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256,
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

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

The evaluator shall verify that the guidance includes instructions for configuring certificates for TLS mutual authentication. If the DN is not compared automatically to the Domain Name or IP address, username, or email address, then the evaluator shall ensure that the guidance includes configuration of the expected DN or the directory server for the connection.

The evaluator shall also 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).

(TD0459 applied)

Section “FCS_CKM.1 – Enable FIPS 140-2 Mode” and section “Enable Common Criteria Mode” in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section “FCS_TLSS_EXT.2.1” in the **Admin Guide** lists the available TLS cipher suites consistent with those identified FCS_RADSEC_EXT.1.

Section “Add Network Access Devices” in the **Admin Guide** describes how to add network access devices to the system before conducting authentication events via the Configuration -> Network -> Devices menu in the Web UI. A Network Access Device (NAD) can be configured to use either RADIUS or RadSec. When communication between NAD and the TOE will occur over RadSec the shared secret key/password is automatically set to “radsec” in compliance with the RFC behavior. RadSec sessions will use certificate validation to establish communication.

Certificates may be selected on the RadSec Settings tab which also includes settings for configuring the Validate Certificate option to *Validate with CN or SAN*. When using RadSec, only TCP port 2088 is used for all communication between NAD and ClearPass.

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

Test 1: The evaluator shall attempt to establish a non-TLS RADIUS session, and verify that the TOE does not process such requests, and that it signals rejection to the originator.

Test 2: [conditional] If 'X.509v3 certificates' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall configure the server to send a certificate request to the client and shall attempt a connection without sending a certificate from the client. The evaluator shall verify that the connection is denied.

Test 3: [conditional] If 'X.509v3 certificates' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

Test 4: [conditional] If 'X.509v3 certificates' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

Test 5: [conditional] If 'X.509v3 certificates' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall present a client certificate that does not chain to one of the Certificate Authorities (either a Root or Intermediate CA) in the server's Certificate Request message. The evaluator shall verify that the attempted connection is denied.

Test 6: [conditional] If 'X.509v3 certificates' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall present a client certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

Test 7: [conditional] If 'X.509v3 certificates' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall perform the following modifications to the traffic:

o Modify a byte in the client's certificate. The evaluator shall verify that the server rejects the connection.

o Modify a byte in the client's Certificate Verify handshake message. The evaluator shall verify that the server rejects the connection.

Test 8: [conditional] If 'X.509v3 certificates' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

Test 9: [conditional] If 'pre-shared keys' is selected in FCS_RADSEC_EXT.1.2, the evaluator shall generate an invalid key and demonstrate that a client cannot successfully complete a protocol negotiation using this key.

Test 10: 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 in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 11: 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 12: The evaluator shall use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

Test 13: The evaluator shall perform the following modifications to the traffic:

o Modify at a byte in the client's nonce in the Client Hello handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

o Modify the signature block in the Client's Key Exchange handshake message, and verify that the server rejects the client's Certificate Verify handshake message (if using mutual authentication) or that the server denies the client's Finished handshake message.

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

o After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.

o Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

Test 14: The evaluator shall send a Client Hello requesting a connection with version SSL 1.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 2.0, SSL 3.0, TLS 1.0, and any non-selected TLS versions.

Test 15 & 16: Removed by TD0459

Test 1 - The evaluator configured the TOE for only Radius over TLS and then sent a RADIUS-request using EAP-TLS and viewed that the TOE discarded the attempt due to not using RadSec.

Test 2 - The evaluator configured a test client to connect to the TOE using RadSec and attempted a connection where the client did not provide a certificate as requested by the TOE. The client was unable to connect when the server required a certificate and none was provided.

Test 3 - The evaluator configured a test client to connect to the TOE using RadSec with an unsupported signature algorithm (md5). The connection attempt failed.

Test 4 - For the invalid chain, the evaluator attempted to connect the test client to the RadSec TOE server with a cert that chained to an invalid CA and confirmed that the connection failed. The successful connection with a valid cert chain is demonstrated as part of Test 10 below.

Test 5 – The testing of a certificate that does not chain to a CA in the certificate request was demonstrated in Test 4 above.

Test 6 - The evaluator configured the TOE to connect to a test client and attempted two connections. During the first TLS negotiation the test client sent a valid certificate chaining to a CA known by the TOE. The certificate included the Client Authentication extended key usage (EKU) field. The PCAP for this part of the test shows that the server issued the Change Cipher Spec and Encrypted Handshake messages, which signify that the TLS connection is successful. During the second connection, the client presented a certificate chaining to a CA known by the TOE, however, the certificate did not include the Client Authentication extended key usage (EKU) field. In the PCAP for this part of the test, the server generates a fatal alert and closes the connection.

Test 7 - The evaluator alternately configured a test client to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator then attempted to connect the test client to the RadSec TOE server and confirmed that the connection only succeeded if the certificate was not modified/corrupted.

Test 8 - The evaluator configured a test client to connect to the TOE using RadSec. The evaluator alternately used a valid client certificate and a certificate with an identifier not configured on the TOE but is otherwise valid (e.g., chains to the root configured on the TOE) and the control connection succeeded while the second connection failed.

Test 9- Not applicable. The TOE does not support pre shared key for RadSec.

Test 10 - The evaluator attempted to connect to the TOE using each of the ciphersuites identified by the Protection Profile and confirmed that the connection was successful for each ciphersuite specified by the requirement.

Test 11 - The evaluator attempted to connect to the TOE using all ciphers except those defined in the PP and observed that the connection failed in each case. The evaluator attempted to connect to the TOE using the TLS_NULL_WITH_NULL_NULL ciphersuite and confirmed that the connection failed.

Test 12 – The evaluator used a client to send a key exchange message of an invalid type (a TLS_RSA key exchange message is substituted with a TLS_ECDHE key exchange message) and confirmed that connection failed.

Test 13 – The evaluator initiated a RadSec connection from a test client to the TOE and alternately performed the following modifications of the traffic and confirmed that in each case the connection was rejected.

a. Modified a byte in the client's nonce in the Client Hello handshake message
b. Modified the signature block in the Client's Key Exchange handshake message
c. Modified a byte in the Client Finished handshake message
d. After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, a Client Hello with the session identifier from the previous test was sent
e. This test was performed in part c above

Test 14 - The evaluator alternately attempted to connect to the TOE using SSL2.0, SSL3.0, TLSv1.0, TLSv1.1 and TLSv1.2 and confirmed that only TLS v1.2 resulted in a successful connection as expected.

2.3.14 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.3.14.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.14.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

Table 6-1 in Section 6.3 of the ST specifies the FCS_RBG_EXT.1 requirement using CTR_DRBG (AES) with a software based noise source. The TOE uses one software based noise source – Jitter Entropy daemon.

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

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

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

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 the table "TOE CAVP Certificates" in Section 1.1.2.

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

2.3.15.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.15.2 NDcPP22e:FCS_SSHS_EXT.1.2

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

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

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

Section 6.3 of the ST states that the TOE's implementation of SSHv2 supports both public-key and password-based authentication; and packets are limited to 256K bytes. SSH public key authentication supports the ssh-rsa and ecdsa-sha2-nistp256 algorithms. X509 certificates are not supported for SSH. This is consistent with the algorithms selected in FCS_SSHS_EXT.1.5.

Guidance Assurance Activities: None Defined

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

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

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

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

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

Test 1: The evaluator generated an RSA 2048 bit key pair on the test server and then uploaded the key to the TOE and configured the admin user with the public key. The evaluator then attempted to login to the TOE using this public key and observed that the login was successful. The evaluator repeated this for an ecdsa-sha2-nistp256 key.

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

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

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

2.3.15.3 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 6.3 of the ST states that SSHv2 supports both public-key and password based authentication that can be configured. Packets are limited to 256K bytes. As SSH packets are being received, the TOE uses a buffer to build all

packet information. Once complete, the packet is checked to ensure it can be appropriately decrypted. However, if it is not complete when the buffer becomes full (256K bytes) the packet will be dropped.

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

2.3.15.4 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 6.3 of the ST indicates that the TOE supports SSHv2 with aes128-cbc, aes256-cbc aes128-ctr, aes256-ctr, aes128-gcm@openssh.com, and aes256-gcm@openssh.com encryption algorithms, in conjunction with HMAC-SHA-1, HMAC-SHA2-256 and HMAC-SHA2-512. No optional characteristics are identified. The ciphers match the requirement.

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section "FMT_SMR.2.3" in the **Admin Guide** indicates that once Common Criteria Mode is enabled, available algorithms and ciphersuites are limited to those specified within the Security Target (ST). ClearPass CLI access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The "FCS_SSHS_EXT.1.2" and "FCS_SSHS_EXT.1.5" sections in the **Admin Guide** provide instructions for configuring SSH public key authentication using ssh-rsa and ecdsa-sha2-nistp256. This is consistent with the public key algorithms specified in FCS_SSHS_EXT.1.5 in the ST. No administrator settings are available to configure.

Section “FCS_SSHS_EXT.1.4” in the **Admin Guide** identifies the encryption algorithms, AES128-CBC, AES256-CBC, AES128-CTR, AES256-CTR, AES128-GCM or AES256- GCM, that are supported by the TOE. This is consistent with the encryption algorithms identified in FCS_SSHS_EXT.1.4 in the ST. Administrators may select AES-CTR (AES128-CTR or AES256-CTR), AES-GCM (AES128-GCM or AES256- GCM), or All (AES-CBC, AES-CTR, and AES-GCM) options. There is no configuration option to select between 128- and 256-bit algorithms.

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

The evaluator established an SSH connection using each claimed algorithm and confirmed that each attempt was successful.

2.3.15.5 NDcPP22e:FCS_SSHS_EXT.1.5

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

Section 6.3 of the ST identifies the ssh-rsa, rsa-sha2-256, rsa-sha2-521 and ecdsa-sha2-nistp256 algorithms. This matches the requirement.

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

Section “FCS_CKM.1 – Enable FIPS 140-2 Mode” and section “Enable Common Criteria Mode” in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section “FMT_SMR.2.3” in the **Admin Guide** indicates that once Common Criteria Mode is enabled, available algorithms and ciphersuites are limited to those specified within the Security Target (ST). ClearPass CLI access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved

cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The "FCS_SSHS_EXT.1.5" section in the **Admin Guide** states the TOE supports SSH host key algorithms of ssh-rsa, rsa-sha2-256, rsa-sha2-512, and ecdsa-sha2-nistp256. This is consistent with the host key algorithms specified in FCS_SSHS_EXT.1.5 in the ST. No administrator settings are available to configure.

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

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

Has effectively been moved to FCS_SSHS_EXT.1.2.

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

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

Test 1: The evaluator established an SSH connection with the TOE using each claimed host public key algorithm and observed that the connection was successful in each case. The evaluator then attempted to establish an SSH connection with a disallowed authentication algorithm and verified that the connection failed.

Test 2: The evaluator attempted to connect to the TOE using an SSH client alternately using a host pubkey not claimed by TOE. The evaluator found that an unsupported pubkey would not be used to establish an SSH session.

2.3.15.6 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 6.3 of the ST identifies SSHv2 with HMAC-SHA-1, HMAC-SHA2-256 and HMAC-SHA2-512. This matches the requirement. The ST also notes that when aes*-gcm@openssh.com is negotiated as the encryption algorithm, the MAC algorithm field is ignored and GCM is implicitly used as the MAC.

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

Section “FCS_CKM.1 – Enable FIPS 140-2 Mode” and section “Enable Common Criteria Mode” in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section “FMT_SMR.2.3” in the **Admin Guide** indicates that once Common Criteria Mode is enabled, available algorithms and ciphersuites are limited to those specified within the Security Target (ST). ClearPass CLI access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The “FCS_SSHS_EXT.1.2” and “FCS_SSHS_EXT.1.5” sections in the **Admin Guide** provide instructions for configuring SSH public key authentication using ssh-rsa and ecdsa-sha2-nistp256. This is consistent with the public key algorithms specified in FCS_SSHS_EXT.1.5 in the ST. No administrator settings are available to configure.

Section “FCS_SSHS_EXT.1.6” in the **Admin Guide** identifies the integrity algorithms, hmac-sha1, hmac-sha2-256, or hmac-sha2-512, that are supported by the TOE. This is consistent with the integrity algorithms identified in FCS_SSHS_EXT.1.6 in the ST. No administrator settings are available to configure.

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 and observed a successful connection using each claimed integrity algorithm. The evaluator then attempted to establish an SSH connection using a disallowed MAC algorithm and confirmed that the connection was rejected.

Test 2: This test has been performed in Test 1 where a disallowed MAC algorithm is also tested.

2.3.15.7 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 6.3 of the ST identifies ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp512 as the key exchange methods supported. This matches the requirement.

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section "FMT_SMR.2.3" in the **Admin Guide** indicates that once Common Criteria Mode is enabled, available algorithms and ciphersuites are limited to those specified within the Security Target (ST). ClearPass CLI access does not require further changes to access it in this mode. Most modern web browsers support the available ciphers without further configuration. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

The "FCS_SSHS_EXT.1.2" and "FCS_SSHS_EXT.1.5" sections in the **Admin Guide** provide instructions for configuring SSH public key authentication using ssh-rsa and ecdsa-sha2-nistp256. This is consistent with the public key algorithms specified in FCS_SSHS_EXT.1.5 in the ST. No administrator settings are available to configure.

Section "FCS_SSHS_EXT.1.7" in the **Admin Guide** identifies the SSH key exchange methods, diffie-hellman-group14-sha1 and ecdh-sha2-nistp256, that are supported by the TOE. This is consistent with the key exchange methods identified in FCS_SSHS_EXT.1.7 in the ST. No administrator settings are available to configure.

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 and Test 2: The evaluator attempted to connect to the TOE using an SSH client alternately using an unallowable key exchange algorithm which fails and then using key exchange algorithm(s) that can be claimed. The evaluator confirmed that a connection attempt using diffie-hellman-group1-sha1 key exchange failed and that the connection was successful with the claimed key exchange method (diffie hellman-group14-sha1).

2.3.15.8 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 6.3 of the ST states that once an SSH session is established, the TOE starts a timer and keeps track of data exchanged. Once either 128MB of data is transferred or 1 hour elapses, the TOE issues a rekey message causing news keys to be exchanged between the TOE and the SSH client and the timer and data counters are reset.

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 "FCS_SSHS_EXT.1.8" section in the **Admin Guide** states that SSH rekey events are initiated every 128MB of data sent over the connection, or every 60 minutes (1 hour). Two (2) events will occur for rekey events. The first is ClearPass sending clients updated keys. The second is ClearPass receiving updated client keys. SSH rekey events will occur for either one (1) hour or 128 megabyte (MB) of data transferred, whichever event occurs first. There is no configuration needed.

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

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

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

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

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

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

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

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

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

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

The evaluator attempted to connect to the TOE using a SSH client alternately waiting an hour and generating 1GB of data. The evaluator confirmed that a rekey happened before each of those thresholds.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.3.16.1 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 6.3 of the ST states that the TOE supports TLSv1.2 with AES (CBC and GCM) 128- or 256-bit ciphers, in conjunction with SHA-1, SHA-256, SHA-384 and RSA and ECDSA. Any other SSL/TLS versions are not supported by the TOE and such connection attempts will be rejected. No optional extensions are identified. The TOE acts as a TLS server with mutual authentication and requires no additional configuration to support the evaluated ciphersuites listed in the NDcPP21:FCS_TLSS_EXT.2 requirement which are as follows:

- TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268,

- TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268,
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 3268,
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 3268,
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,
- TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,
- TLS_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,
- TLS_DHE_RSA_WITH_AES_128_CBC_ SHA256 as defined in RFC 5246,
- TLS_DHE_RSA_WITH_AES_256_CBC_ SHA256 as defined in RFC 5246,
- TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,
- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289

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).

Section “FCS_CKM.1 – Enable FIPS 140-2 Mode” and section “Enable Common Criteria Mode” in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section “FMT_SMR.2.3” in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The available cipher suites are listed in section “FCS_TLSS_EXT.2.1” in the **Admin Guide**. These ciphersuites are consistent with those identified in the ST.

Section FCS_TLSS_EXT.2.2 in the **Admin Guide** provides instructions for how to configure Web UI sessions using certificate identification through TLS mutual authentication. This is configured with the aid of a setup wizard which is available to administrators in the Web UI.

Section FCS_TLSS_EXT.1.2 in the **Admin Guide** states that in Common Criteria related deployments, the use of TLS v1.0 and TLS v1.1 is disabled by default. This prevents TLS versions prior to v1.2 from use in any component (such as RADIUS, RadSec, or WebUI). This setting cannot be modified in CC operating mode.

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

Test 2: The evaluator shall 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 ciphersuites identified by the Protection Profile. A packet capture was obtained for each connection attempt. The evaluator verified that successful connections were only established with the claimed ciphersuites. These tests were repeated for 2 variations as follows: RSA ciphers and ECDSA ciphers.

Test 2 - The evaluator attempted to connect to the TOE using all ciphers except those defined in the PP and observed that the connection failed. The evaluator used the openssl s_client to attempt to connect to the TOE using the TLS_NULL_WITH_NULL_NULL ciphersuite and observed that the connection failed.

Test 3a - The evaluator attempted a connection to the TOE where the evaluator modified a byte in the Client Finished handshake message, verified that the server rejected the connection attempt after receiving the client Finished message and no application data was exchanged.

Test 3b - The evaluator viewed a successful TLS connection and saw 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 examined the Finished message and confirmed that it did 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, which it did not.

2.3.16.2 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 6.3 of the ST states that the TOE TLSv1.2 with AES (CBC and GCM) 128- or 256-bit ciphers, in conjunction with SHA-1, SHA-256, and SHA-384 and RSA and ECDSA. Any other SSL/TLS versions are not supported by the TOE and such connection attempts 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.

Section “FCS_TLSS_EXT.1.2” in the **Admin Guide** indicates that support for older SSL based protocols (SSL 1.0, SSL 2.0 or SSL 3.0) is no longer available in any ClearPass operating configuration. This section further states that in Common Criteria related deployments, the use of TLS v1.0 and TLS v1.1 is disabled by default. This prevents TLS versions prior to v1.2 from use in any component (such as RADIUS, RadSec, or WebUI). This setting cannot be modified in CC operating mode.

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 alternately attempted to connect to the TOE using SSL2.0, SSL3.0, TLSv1.0, TLSv1.1 and TLSv1.2 and confirmed that the connection was rejected in all cases except the claimed protocol version (TLSv1.2).

2.3.16.3 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.

The TOE performs key establishment, depending on the TLS cipher suite that is negotiated, using RSA with key sizes of 2048, 3072 or 4096 bits, ECDSA with secp256r1 or secp384r1 NIST curves, or using Diffie-Hellman with 2048 or 3072 bits.

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

Section "FCS_CKM.1 – Enable FIPS 140-2 Mode" and section "Enable Common Criteria Mode" in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section "FMT_SMR.2.3" in the **Admin Guide** states that once Common Criteria Mode is enabled, the list of ciphers available for use is limited to those specified within the Security Target (ST). The available cipher suites are listed in section "FCS_TLSS_EXT.2.1" in the **Admin Guide**. These ciphersuites are consistent with those identified in the ST.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides a list of all the allowed hash and encryption types that may be used for either HTTPS or RADIUS server certificates when operating in CC Mode:

```
Encryption: RSA
Size: 2048-bit, 3072-bit, or 4096-bit
Hash: SHA1, SHA256, SHA384, or SHA512

Encryption: ECDSA
Size: NIST/SECG curve over 384-bit prime field
Hash: SHA1, SHA256, SHA384, or SHA512
```

Section FCS_TLSS_EXT.2.2 in the **Admin Guide** provides instructions for how to configure Web UI sessions using certificate identification through TLS mutual authentication. This is configured with the aid of a setup wizard which is available to administrators in the Web UI.

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 when the evaluator's client specified only one ECDHE key exchange method in the Client Hello. The evaluator observed that the TOE successfully established connections using all of the claimed ECDHE key exchanges.

Test 2 - The evaluator attempted to establish a TLS session with the TOE when the evaluator's client specified only one DHE key exchange method in the Client Hello. The evaluator observed that the TOE successfully established connections using all of the claimed DHE key exchanges.

Test 3 - The evaluator attempted to establish a TLS session with the TOE when the evaluator's client specified only one RSA key exchange method in the Client Hello. The evaluator observed that the TOE successfully established connections using all of the claimed RSA key exchanges.

2.3.16.4 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 6.3 of the ST states the TLS server implementation of the TOE supports session tickets used for TLS session resumption and are encrypted using symmetric algorithms consistent with FCS_COP.1/DataEncryption claims in this ST – AES used in CBC and GCM modes and key sizes of 128 and 256 bits. The session tickets adhere to the structural format provided in section 4 of RFC 5077.

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: This test is not applicable as the TOE supports session resumption with session tickets.

Test 2: This test is not applicable as the TOE supports session resumption with session tickets.

Test 3 – The evaluator first attempted to resume a session using a valid session ticket. The server correctly reuses the client's proposed session ticket and the session is successfully resumed.

The evaluator then attempted a second connection and observed the session ticket. The evaluator then attempted to reuse a modified version of the session ticket to resume the connection and verified that the session was not resumed as expected.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.17 TLS Server Support for Mutual Authentication (NDcPP22e:FCS_TLSS_EXT.2)

2.3.17.1 NDcPP22e:FCS_TLSS_EXT.2.1

TSS Assurance Activities: The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

The evaluator shall verify the TSS describes how the TSF uses certificates to authenticate the TLS client. The evaluator shall verify the TSS describes if the TSF supports any fallback authentication functions (e.g. username/password, challenge response) the TSF uses to authenticate TLS clients that do not present a certificate. If fallback authentication functions are supported, the evaluator shall verify the TSS describes whether the fallback authentication functions can be disabled.

Section 6.3 in the ST states that The TOE acts as a TLS server with mutual authentication and requires no additional configuration to support the evaluated ciphersuites listed in the NDcPP22e:FCS_TLSS_EXT.2 requirement. The TOE will not establish a connection when an invalid client certificate is presented and no fallback authentication method is supported.

Guidance Assurance Activities: If the TSS indicates that mutual authentication using X.509v3 certificates is used, the evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for TLS mutual authentication.

The evaluator shall verify the guidance describes how to configure the TLS client certificate authentication function. If the TSF supports fallback authentication functions, the evaluator shall verify the guidance provides instructions for configuring the fallback authentication functions. If fallback authentication functions can be disabled, the evaluator shall verify the guidance provides instructions for disabling the fallback authentication functions.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to install certificates that are signed by a trusted certificate authority (CA) via Administration -> Certificates -> Trust List in the Web UI. It also describes how to update the ClearPass Server RADIUS and HTTPS certificates.

Section "FCS_TLSS_EXT.2.2" in **the Admin Guide** provides instructions for how to configure Web UI sessions using certificate identification through TLS mutual authentication. This is configured with the aid of a setup wizard which is available to administrators in the Web UI. The option to "require a client certificate from the user" is presented during this configuration process. This process requires that all DNS entries be configured correctly prior to establishment. Fully qualified domain names (FQDN) must be resolvable from the client. Additionally, client systems need to have the ClearPass WebUI public certificate available locally, along with any required CA intermediate certificates.

No fallback authentication is supported.

Testing Assurance Activities: Test 1a [conditional]: If the TOE requires or can be configured to require a client certificate, the evaluator shall configure the TOE to require a client certificate and send a Certificate Request to the client. The evaluator shall attempt a connection while sending a certificate_list structure with a length of zero in the Client Certificate message. The evaluator shall verify that the handshake is not finished successfully and no application data flows.

Test 1b [conditional]: If the TOE supports fallback authentication functions and these functions cannot be disabled. The evaluator shall configure the fallback authentication functions on the TOE and configure the TOE to send a Certificate Request to the client. The evaluator shall attempt a connection while sending a certificate_list structure with a length of zero in the Client Certificate message. The evaluator shall verify the TOE authenticates the connection using the fallback authentication functions as described in the TSS.

Note: Testing the validity of the client certificate is performed as part of X.509 testing.

Test 2 [conditional]: If TLS 1.2 is claimed for the TOE, the evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

Test 3: The aim of this test is to check the response of the server when it receives a client identity certificate that is signed by an impostor CA (either Root CA or intermediate CA). To carry out this test the evaluator shall configure the client to send a client identity certificate with an issuer field that identifies a CA recognised by the TOE as a trusted CA, but where the key used for the signature on the client certificate does not correspond to the CA certificate trusted by the TOE (meaning that the client certificate is invalid because its certification path does not terminate in the claimed CA certificate). The evaluator shall verify that the attempted connection is denied.

Test 4: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

Test 5: The evaluator shall perform the following modifications to the traffic:

a) Configure the server to require mutual authentication and then connect to the server with a client configured to send a client certificate that is signed by a Certificate Authority trusted by the TOE. The evaluator shall verify that the server accepts the connection.

b) Configure the server to require mutual authentication and then modify a byte in the signature block of the client's Certificate Verify handshake message (see RFC5246 Sec 7.4.8). The evaluator shall verify that the server rejects the connection.

Note: Testing the validity of the client certificate is performed as part of X.509 testing.

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

Test 6: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds, and a trusted channel can be established.

Test 7: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

Test 8 [conditional]: The purpose of this test is to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

Test 1: The evaluator configured the TOE to send a certificate request to the client and then attempted a connection with a client that had no client certificates to offer. The evaluator confirmed that the server disconnected and rejected the connection attempt.

The TOE does not support fallback authentication for cert auth.

Test 2: The evaluator configured the TOE to send a certificate request to the client and then attempted a connection with a client certificate signed with an unsupported signature algorithm (md5). The evaluator confirmed that the server disconnected and rejected the connection attempt.

Test 3: This test is performed in FIA_X509_EXT.1, test 1 where the trusted root availability is tested.

Test 4: The evaluator attempted to connect to the TOE using a client certificate signed by an imposter CA that matched the identity of the CA in the TOE's certificate request, but had a different key. The evaluator confirmed that the server disconnected and rejected the connection attempt.

Test 5a: This test is performed in FIA_X509_EXT.1.1, test 5 where multiple certificate modifications are tested.

Test 5b: The evaluator attempted to connect to the TOE using a client certificate with a modified Certificate Verify handshake message. The evaluator observed that the connection failed.

Test 6: The successful connection with a valid CA chain was demonstrated in FCS_TLSS_EXT.2.4 test 1.

Test 7: The testing of the various certificate failures is performed throughout FCS_TLSS_EXT.2.4 and FIA_X509_EXT.1/Rev.

Test 8: Not applicable. The TOE does not support administrative override of certificate failures.

2.3.17.2 NDcPP22e:FCS_TLSS_EXT.2.2

TSS Assurance Activities: The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

The evaluator shall verify the TSS describes how the TSF uses certificates to authenticate the TLS client. The evaluator shall verify the TSS describes if the TSF supports any fallback authentication functions (e.g. username/password, challenge response) the TSF uses to authenticate TLS clients that do not present a certificate. If fallback authentication functions are supported, the evaluator shall verify the TSS describes whether the fallback authentication functions can be disabled.

See FCS_TLSS_EXT.2.1 above as the requirements are shared by the two SFRs.

Guidance Assurance Activities: If the TSS indicates that mutual authentication using X.509v3 certificates is used, the evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for TLS mutual authentication.

The evaluator shall verify the guidance describes how to configure the TLS client certificate authentication function. If the TSF supports fallback authentication functions, the evaluator shall verify the guidance provides instructions for configuring the fallback authentication functions. If fallback authentication functions can be disabled, the evaluator shall verify the guidance provides instructions for disabling the fallback authentication functions.

See FCS_TLSS_EXT.2.1 above as the requirements are shared by the two SFRs.

Testing Assurance Activities: Test 1a [conditional]: If the TOE requires or can be configured to require a client certificate, the evaluator shall configure the TOE to require a client certificate and send a Certificate Request to the client. The evaluator shall attempt a connection while sending a certificate_list structure with a length of zero in the Client Certificate message. The evaluator shall verify that the handshake is not finished successfully and no application data flows.

Test 1b [conditional]: If the TOE supports fallback authentication functions and these functions cannot be disabled. The evaluator shall configure the fallback authentication functions on the TOE and configure the TOE to send a Certificate Request to the client. The evaluator shall attempt a connection while sending a certificate_list structure with a length of zero in the Client Certificate message. The evaluator shall verify the TOE authenticates the connection using the fallback authentication functions as described in the TSS.

Note: Testing the validity of the client certificate is performed as part of X.509 testing.

Test 2 [conditional]: If TLS 1.2 is claimed for the TOE, the evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

Test 3: The aim of this test is to check the response of the server when it receives a client identity certificate that is signed by an impostor CA (either Root CA or intermediate CA). To carry out this test the evaluator shall configure the client to send a client identity certificate with an issuer field that identifies a CA recognised by the TOE as a trusted CA, but where the key used for the signature on the client certificate does not correspond to the CA certificate trusted by the TOE (meaning that the client certificate is invalid because its certification path does not terminate in the claimed CA certificate). The evaluator shall verify that the attempted connection is denied.

Test 4: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose.

Test 5: The evaluator shall perform the following modifications to the traffic:

a) Configure the server to require mutual authentication and then connect to the server with a client configured to send a client certificate that is signed by a Certificate Authority trusted by the TOE. The evaluator shall verify that the server accepts the connection.

b) Configure the server to require mutual authentication and then modify a byte in the signature block of the client's Certificate Verify handshake message (see RFC5246 Sec 7.4.8). The evaluator shall verify that the server rejects the connection.

Note: Testing the validity of the client certificate is performed as part of X.509 testing.

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

Test 6: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds, and a trusted channel can be established.

Test 7: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

Test 8 [conditional]: The purpose of this test is to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

See FCS_TLSS_EXT.2.1 above as the requirements are shared by the two SFRs.

2.3.17.3 NDcPP22e:FCS_TLSS_EXT.2.3

TSS Assurance Activities: The evaluator shall verify that the TSS describes which types of identifiers are supported during client authentication (e.g. Fully Qualified Domain Name (FQDN)). If FQDNs are supported, the evaluator shall verify that the TSS describes that corresponding identifiers are matched according to RFC6125. For all other types of identifiers, the evaluator shall verify that the TSS describes how these identifiers are parsed from the certificate, what the expected identifiers are and how the parsed identifiers from the certificate are matched against the expected identifiers.

Section 6.3 in the ST states that the SAN or CN in the certificate presented by the peer must match the expected identifier (user-name) or the TOE will not establish the TLS connection. FQDNs are not supported as an identifier for the WebUI.

Guidance Assurance Activities: The evaluator shall ensure that the AGD guidance describes the configuration of expected identifier(s) for X.509 certificate-based authentication of TLS clients. The evaluator ensures this description includes all types of identifiers described in the TSS and, if claimed, configuration of the TOE to use a directory server.

Section "FCS_TLSS_EXT.2/FCS_TLSS_EXT.2.5" in the **Admin Guide** provides instructions for how to configure Web UI sessions using certificate identification through TLS mutual authentication. This is configured with the aid of a setup wizard and a service template which is available to administrators in the Web UI. This process requires that all DNS entries be configured correctly prior to establishment. It is critical to ensure that fully qualified domain names (FQDN) are resolvable from the client. This section explains that when configuring the service, the 'Enforcement Details' tab allows you to select attributes (e.g. Subject-CN) from the certificate to match against enforcements.

Testing Assurance Activities: The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

The evaluator configured the enforcement policy on the TOE to define the set of allowable DNs. The evaluator then attempted to connect using a TOE client certificate that did not match an allowed DN in the login enforcement policy. The connection attempt was rejected.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: For all tests in this chapter the TLS client used for testing of the TOE shall support mutual authentication.

All FCS_TLSS_EXT.2 related testing was performed with both the TOE's web UI and the client supporting TLS mutual authentication.

2.4 IDENTIFICATION AND AUTHENTICATION (FIA)

2.4.1 AUTHENTICATION FAILURE HANDLING (AUTHSRVEP10:FIA_AFL.1)

2.4.1.1 AUTHSRVEP10:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.2 AUTHSRVEP10: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.

Section 6.4 of the ST states that when authentication fails, the TOE increments a per-user counter. The per-user counter is reset to 0 upon successful authentication. If the per-user counter reaches the configured limit, the account is locked. For SSH-based CLI logins the per-user counter is reset to 0 when the account is explicitly unlocked or after the configured period, whichever occurs first. If the configured authentication threshold is exceeded on the Web UI, the account is locked out until an administrator resets the account to re-enable Web UI login for that account. Accounts are never locked out on the local console.

Component Guidance Assurance Activities: The evaluator shall also examine the operational guidance to ensure that instructions for configuring the authentication failure threshold and the TOE's response to the threshold being met (if configurable), 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 trusted path used to access the TSF (see FTP_TRP.1), all must be described.

Section “FIA_AFL.1.2” in the **Admin Guide** provides instructions for configuring the authentication failure threshold for SSH access. SSH access can be locked after a specified number of failed attempts for a configurable length of time. By default, SSH lockout is not enabled. To enable SSH lockout, the ‘ssh lockout count’ and ‘ssh lockout duration’ commands are used. To extend the lockout capability to include SSH public key authentication, the ‘ssh lockout mode advanced command’ should be used. Advanced mode will apply the same conditions to both username password authentication and SSH public key authentication. When Advanced mode is enabled, the only way to unlock the account is by waiting for the duration to expire or to execute the unlock command from the console or previously established SSH session. The administrator uses the ‘ssh unlock’ command to reset the SSH lockout.

The “Disable Admin User and Local User Account” section in the **Admin Guide** provides instructions on configuring the authentication failure threshold for Web UI access. WebUI access can be locked out for administrators after a specified number of failed attempts. The time duration for these events is permanent until unlocked by another administrator. The number of failed attempts can be configured through the WebUI by navigating to Administration > Users and Privileges > Admin Users, selecting the Account Settings link, and then selecting the Disable Accounts tab. The ‘failed attempts count’ field may be populated with the desired number of failed login attempts. Re-enabling accounts can be done from the same screen by clicking on the Reset button.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which remote administrators access the TOE: Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE. The evaluator shall test that once the limit is reached for a given remote administrator account, subsequent attempts with valid credentials are not successful.

Test 2: [conditional] If the TSS indicates that administrative action is necessary to re-enable an account that was locked out due to excessive authentication failures, the evaluator shall perform the steps in Test 1 to lock out an account, follow the operational guidance to manually re-enable the locked out administrator account, and observe that it is once again able to successfully log in.

Test 3: [conditional] If the TSS indicates that an administrator-configurable time period must elapse in order to automatically re-enable an account that was locked out due to excessive authentication failures, the evaluator shall perform the steps in Test 1 to lock out an account, follow the operational guidance to configure a time period of their choosing, and observe through periodic login attempts that the account cannot successfully log in until the configured amount of time has elapsed. The evaluator shall then repeat this test for a different time period of their choosing.

These tests were all demonstrated as part of NDcPP21:FIA_AFL.1.

2.4.2 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.4.2.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.2.2 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 6.4 of the ST states that when authentication fails, the TOE increments a per-user counter. The per-user counter is reset to 0 upon successful authentication. If the per-user counter reaches the configured limit, the account is locked. For SSH-based CLI logins the per-user counter is reset to 0 when the account is explicitly unlocked or after the configured period, whichever occurs first. If the configured authentication threshold is exceeded on the Web UI, the account is locked out until an administrator resets the account to re-enable Web UI login for that account. Accounts are never locked out on the local console.

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

The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be

maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

Section “FIA_AFL.1.2” in the **Admin Guide** provides instructions for configuring the authentication failure threshold for SSH access. SSH access can be locked after a specified number of failed attempts for a configurable length of time. By default, SSH lockout is not enabled. To enable SSH lockout, the ‘ssh lockout count’ and ‘ssh lockout duration’ commands are used. To extend the lockout capability to include SSH public key authentication, the ‘ssh lockout mode advanced command’ should be used. Advanced mode will apply the same conditions to both username password authentication and SSH public key authentication. When Advanced mode is enabled, the only way to unlock the account is by waiting for the duration to expire or to execute the unlock command from the console or previously established SSH session. The administrator uses the ‘ssh unlock’ command to reset the SSH lockout.

The “Disable Admin User and Local User Account” section in the **Admin Guide** provides instructions on configuring the authentication failure threshold for Web UI access. WebUI access can be locked out for administrators after a specified number of failed attempts. The time duration for these events is permanent until unlocked by another administrator. The number of failed attempts can be configured through the WebUI by navigating to Administration > Users and Privileges > Admin Users, selecting the Account Settings link, and then selecting the Disable Accounts tab. The ‘failed attempts count’ field may be populated with the desired number of failed login attempts. Re-enabling accounts can be done from the same screen by clicking on the Reset button.

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 and Test 2: On the Web UI, the evaluator configured the limit for the maximum number of failed login attempts at 3. The evaluator then logged in using incorrect credentials three times. Following this, the evaluator attempted to login using the correct credentials, but found the account was locked. The evaluator then followed the guidance to unlock the user and verified that the user was successfully able to log in again.

Over SSH, the evaluator configured the SSH lockout count to 5. The evaluator then made 5 bad password attempts and confirmed that the account was locked. The evaluator then attempted to login with the correct password, but the attempt failed further confirming the account was locked. On the local console, the evaluator logged in as administrator and followed the guidance to unlock the account and verified that the user was successfully able to log in again.

Over SSH, the evaluator configured both the limit for the maximum number of failed login attempts at 2 and the duration for 1 minute. The evaluator made two bad password attempts followed by a third attempt with a good password. The account was locked after the 2nd attempt. After waiting for 1 minute, the tester attempted to log in with the correct credentials, and witnessed a successful attempt.

2.4.3 Password Management (NDcPP22e:FIA_PMG_EXT.1)

2.4.3.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it contains the lists of the supported special character(s) and minimum and maximum number of characters supported for administrator passwords.

Section 6.4 of the ST states The TOE password authentication mechanism enforces password composition rules. Passwords can contain alphabetic (upper or lower case) characters, numeric characters, and special characters such as any of '!', '@', '#', '$', '%', '^', '&', '*', '(', ')' and they are case-sensitive. The TOE supports the configuration of password composition policies such as:

- No password complexity requirement;
- At least one uppercase and one lowercase letter;
- At least one digit;
- At least one letter and one digit;

- At least one of each: uppercase letter, lowercase letter, digit;
- At least one symbol; and
- At least one of each: uppercase letter, lowercase letter, digit, and symbol.

Additionally, disallowed characters and words can be defined along with even more checks such as disallowing repeating character four times or containing the user identity either forward or backwards. All of the configured policies are enforced whenever a user changes their password.

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 "Passwords and Accounts" section in the **Admin Guide** states the following recommendations to administrators for establishing secure passwords:

- Require a minimum password length of at least 15 characters
- Make use of upper case, lower case, numerical values, and allowed special characters in all passwords
- Passwords are not based on dictionary words (unless passphrases longer than 22 characters are used)
- Secure common passwords (such as CLI users) in a secure location with restricted access.

Examples of special characters include: "!", "@", "#", "$", "%", "^", "&", "*", "(", and ")".

The "Establish Password Policy Enforcement" section in the **Admin Guide** states that the password policy allows passwords with six (6) to one hundred (100) characters and provides instructions for configuring the minimum password length via the Policy Tab in the Account Settings dialog box.

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.

The evaluator attempted to set/change a password for a user's account using several attempts. Throughout those attempts, every upper case letter, lower case letter, digit, and special character (as specified by the SFR in [ST]) were used in a password. The evaluator also confirmed that a minimum length of 15 was required by attempting to set passwords with 14 characters (and observing the TOE reject the password) and of 15 characters (and observing that the TOE accepted the password change).

2.4.4 Extended: Pre-Shared Key Composition (AUTHSRVEP10:FIA_PSK_EXT.1)

2.4.4.1 AUTHSRVEP10:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.4.2 AUTHSRVEP10:FIA_PSK_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 identifies all protocols that allow text-based pre-shared keys and states that text-based pre-shared keys of 22 characters are supported. For each protocol identified by the requirement. The evaluator shall also verify that the selection of IPsec or RadSec matches the selection in FTP_ITC.1.

Section 6.4 of the ST identifies RADIUS and IPsec as the protocols using pre-shared keys. The pre-shared key can be up to 128 characters in length (e.g., including 22 characters). This matches the selection of IPsec in the FTP_ITC.1(1) requirement.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on the composition of strong text-based pre-shared keys and (if the selection indicates keys of various lengths can be entered) that it provides information on the range of lengths supported. The guidance must specify the allowable characters for pre-shared keys and that list must be a super-set of the list contained in FIA_PSK_EXT.1.2.

The "FIA_PSK_EXT.1" section in the **Admin Guide** provides guidance to administrators on the composition of strong text-based pre-shared keys. When IPsec VPNs are established using a pre-shared key (PSK) it is recommended to use a key of at least 22-characters. ClearPass supports PSK values of up to 128-character length. As with any other human derived password, it is recommended that PSK values make use of a mixture of password character types to maximize the entropy and minimize attack capabilities. Upper case, lower case, numerical, and special characters that are supported by both VPN peers are recommended to be used in any PSK. Note that this guidance for strong pre-shared keys mitigates the risk for CVE-2018-5389 which is not a ClearPass vulnerability, but rather a potential vulnerability within the IKEv1 standard itself which affects all products using IKEv1.

Section "Add Network Access Devices" in the **Admin Guide** describe how to add network access devices to the system before conducting RADIUS authentication events and how to configure a RADIUS shared secret key/password. Each NAD should use an agreed upon RADIUS shared secret key/password that is secured using established password security requirements. It is recommended that all shared secrets be at least 22 characters, and that each NAD uses a unique shared secret.

Component Testing Assurance Activities: The evaluator shall also perform the following tests for each protocol (or instantiation of a protocol, if performed by a different implementation on the TOE). Note that one or more of these tests can be performed with a single test case.

Test 1: The evaluator shall compose a pre-shared key of 22 characters that contains a combination of the allowed characters in accordance with the operational guidance, and demonstrates that a successful protocol negotiation can be performed with the key.

Test 2 [conditional]: If the TOE supports pre-shared keys of multiple lengths, the evaluator shall repeat Test 1 using the minimum length; the maximum length; a length inside the allowable range; and invalid lengths beyond the supported range (both higher and lower). The minimum, maximum, and included length tests should be successful, and the invalid lengths must be rejected by the TOE.

Test 1 and Test 2: The evaluator configured a VPN connection using a 22 character pre-shared key and observed that the connection was successful for both IPsec and Radius. The evaluator then attempted connections with the following:

- Maximum Length of 128 characters – the connection succeeded.
- Invalid Length beyond the Maximum – the connection failed
- Invalid Length below the Minimum – the connection failed
- Minimum Length – the connection succeeded.

2.4.5 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.4.5.1 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.

No configuration I needed to ensure authentication data is not revealed.

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

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

Test 1- The evaluator observed during testing that passwords are obscured on the console login.

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

2.4.6.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

See FIA_UIA_EXT.1

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

See FIA_UIA_EXT.1

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

See FIA_UIA_EXT.1

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

2.4.7.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.7.2 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 6.4 of the ST indicates that the TOE authenticates administrative users accessing the TOE via the command-line interface (local serial console or SSH) or web interface (Web UI) in the same manner using its own password-based authentication mechanism. The TOE also supports public-key based authentication of users through the SSH-based CLI interface and supports certificate authentication for the Web UI. In order for an administrative user to access the TOE (i.e., to perform any functions except to see a configured login banner or to access network access control services, including processing RADIUS, Web, and other authentication requests from external entities), an administrative user account must be created for the user with an assigned privilege level. The TOE offers no TSF-mediated functions except display of a login banner until the user is identified and authenticated.

Section 6.8 of the ST states that the TOE provides a trusted path for its remote administrative users accessing the TOE via the Ethernet ports provided on the TOE using either a command line interface using SSH or Web-based graphical user interface using TLS/HTTPS. Local console access via a serial port is also supported for command line access. However, this access is protected by physical protection of the serial interface along with the TOE itself.

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 "Passwords and Accounts" section in the **Admin Guide** states that during initial setup, administrators should specify the initial password for use with the CLI and the Web UI accounts. After initial setup, the administrator should create individual accounts for all administrators and no longer use the default WebUI account or password. In the event that a weak password is initially used, it is recommended to immediately change the password to a more secure option for the default account(s). Permissions to the administrator functions are limited to users with appropriate roles. Only administrators should have access to the security management functionality on the system. General users are not required to have local accounts defined.

The **Admin Guide** refers to the Managing Admin Users section of the *ClearPass Policy Manager User guide* which describes how to create and modify administrator accounts. A web link to this guide is provided in the introductory section on page 2 of the **Admin Guide**.

Section "FMT_SMF.1.1" in the **Admin Guide** describes the local administration available on the TOE using the following interfaces:

- Peripherals (monitor and keyboard directly attached
- RS-232 terminal (serial console)

- Management Ethernet port

Because the Ethernet port may also be used for other communications, section “FMT_SMF.1.1” further describes how to restrict access for both CLI (secure shell) and Administrative WebUI by whitelisting an IP address for each resource.

The “Establish Password Policy Enforcement” section in the **Admin Guide** provides instructions for configuring the password policy including complexity requirements and the password minimum length. The “Passwords and Accounts” section in the **Admin Guide** provides guidance to security administrators regarding the composition of strong passwords.

The “Verify Local User Repository is available” section in the **Admin Guide** describes how to ensure that there is a Local User Repository available while performing initial deployment until all remote authentication sources are able to be validated. The authentication sources must include a Local User Repository prior to enabling Common Criteria mode or an administrator may be locked out.

Section “FIA_X509_EXT.1/Rev” in the **Admin Guide** indicates that valid certificates (including intermediate Certificate Authorities) must be installed prior to enabling Common Criteria Mode.

The “FIA_X509_EXT.1/Rev (Install Certificates)” section of the **Admin Guide** provides instructions for importing a new certificate or for generating a new Certificate Signing Request (CSR). The type of certificate (RSA or ECDSA) used will influence which ciphers are available later. For example, RSA certificates will not be able to perform ECDSA based ciphers, so those encryption options will automatically be disabled.

Section “FCS_CKM.1 – Enable FIPS 140-2 Mode” and section “Enable Common Criteria Mode” in the **Admin Guide** describe how to configure the TOE in these modes such that the cryptographic schemes and algorithms available are limited to those specified for the evaluation.

Section “FMT_SMR.2.3” in the **Admin Guide** indicates that once Common Criteria Mode is enabled, the list of algorithms and ciphersuites available for use is limited to those specified within the Security Target. The ClearPass console access does not require further changes to access it in this mode. SSH clients that are not configured to support only FIPS140-2 approved cryptographic algorithms will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

Section FIA_UIA_EXT.1 in the **Admin Guide** provides instructions for blocking a port to prevent inbound connections when ClearPass is being deployed in a stand-alone environment.

Section “FCS_TLSS_EXT.2.2” in the **Admin Guide** provides instructions for how to configure Web UI sessions using certificate identification through TLS mutual authentication. This is configured with the aid of a setup wizard which is available to administrators in the Web UI.

The “FCS_SSHS_EXT.1.2” section in the **Admin Guide** provides instructions for configuring SSH public key authentication.

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

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

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

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

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

Test 1 – The evaluator logged in locally via the console and remotely via the Web UI and via the SSH CLI and performed unsuccessful and successful logons of each type. The evaluator observed that a session was only established when the correct authentication credentials were used.

Test 2 - As part of Test 1, the evaluator saw that the only action available prior to login was the viewing of the TOE access banner.

Test 3 - As part of Test 1, the evaluator saw that the only action available prior to login was the viewing of the TOE access banner.

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

2.4.8 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/Rev)

2.4.8.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto

the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

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

Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE's trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

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

c) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates - conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the peer certificate and revocation of the peer intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore the revoked certificate(s) used for testing shall not be a trust anchor.

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

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

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

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

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

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

Test 8a: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a valid chain of EC certificates (terminating in a trusted CA certificate), where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

Test 8b: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a chain of EC certificates (terminating in a trusted CA certificate), where the intermediate certificate in the certificate chain uses an explicit format version of the Elliptic Curve parameters in the public key information field, and is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

Test 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

(TD0527 12/2020 update applied)

These tests are all iterated for 2 variations as follows: IPSec and TLS/HTTPS.

Test 1 – IPSec: The successful connection using a valid cert chain was demonstrated as part of FCS_IPSEC_EXT.1.14 test 5. To invalidate the chain the evaluator initiated a connection sending a cert that did not chain to a CA in the TOE's truststore and viewed that the connection failed as expected.

TLS/HTTPS: The successful connection with the valid CA was demonstrated in FCS_TLSS_EXT.2.4 test 1. The invalidation of the chain, sending a cert signed by an CA not in the TOE's truststore was demonstrated in FCS_TLSS_EXT.2.4 test 4 where the CA was an imposter.

Test 2 – IPSec: The evaluator alternately configured a test peer to send an authentication certificate 1) that is valid, 2) that is expired, and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection only succeeded if there were no expired certificates.

TLS/HTTPS: The evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that is expired, and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the test client to the TLSS TOE server and confirmed that the connection only succeeded if there were no expired certificates.

Test 3 –
IPSec: The evaluator alternately configured a test peer to send an authentication certificate 1) that is valid, 2) that is revoked, and 3) issued by an intermediate CA that is revoked. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection only succeeded if there were no revoked certificates.

TLS/HTTPS: The evaluator alternately configured a test client to send an authentication certificate 1) that is valid, 2) that is revoked, and 3) issued by an intermediate CA that is revoked. In each case, the evaluator then attempted to connect the test client to the TLSS TOE and confirmed that the connection only succeeded if there were no revoked certificates.

Test 4 –

IPSec: The evaluator alternately configured a test peer to send an authentication certificate 1) that is valid, and 2) issued by an intermediate CA whose issuer CA refers to an OCSP revocation server where the signer lacks OCSPSigning. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection only succeeded if all retrieved OCSP responses are signed using certificates with OCSPSigning.

TLS/HTTPS: The evaluator alternately configured a test client to send an authentication certificate 1) that is valid and 2) issued by an intermediate CA whose issuer CA refers to an OCSP revocation server where the signer lacks OCSPSigning. In each case, the evaluator then attempted to connect the test client and the TLSS TOE and confirmed that the connection only succeeded if all retrieved OCSP responses are signed using certificates with OCSPSigning.

Test 5 – IPSec: The evaluator alternately configured a test peer to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection only succeeded if the certificate is not modified/corrupted.

TLS/HTTPS: The evaluator alternately configured a test client to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator then attempted to connect the test client to the TLSS TOE server and confirmed that the connection only succeeded if the certificate is not modified/corrupted.

Test 6 – This test was performed as part of Test 5.

Test 7 – This test was performed as part of Test 5.

Test 8a – For both IPsec and TLS/HTTPS the evaluator configured the TOE and a network peer (a test peer) to attempt connections. The test peer presented a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where the ECDSA certificates have a supported named curve. The TOE accepted the certificate as valid.

Test 8b - For both IPsec and TLS/HTTPS the evaluator configured the TOE and a network peer (a test peer) to attempt connections. The test peer presented a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where an intermediate CA's ECDSA certificate has an explicit curve. The TOE rejected the certificate as invalid.

Test 8c - The evaluator attempted to load an intermediate CA ECDSA certificate with a supported named curve into the TOE trust store and observed that the certificate can be loaded. The evaluator attempted to load an intermediate CA ECDSA with an explicit curve and verified that the attempt failed.

2.4.8.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation). For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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

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

The evaluator shall repeat these tests for each distinct use of certificates. Thus, for example, use of certificates for TLS connection is distinct from use of certificates for trusted updates so both of these uses would be tested. But there is no need to repeat the tests for each separate TLS channel in FTP_ITC.1 and FTP_TRP.1/Admin (unless the channels use separate implementations of TLS).

Test 1: IPSec RSA: The evaluator alternately configured a test peer to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and confirmed that the connection was rejected in each case.

TLS/HTTPS: The evaluator first attempted to load a sub CA with no basic constraints to the TOE's truststore. The evaluator viewed that the attempt failed. The evaluator then configured the test client to send an authentication certificate issued by the Sub CA with no BasicConstraints. The evaluator viewed that the TOE rejected the connection.

Test 2: IPSec RSA: This test is performed in Test 1.

TLS/HTTPS: For this test, the evaluator first attempted to load a sub CA with basic constraints set to CA false to the TOE's truststore. The evaluator viewed that the attempt failed. The evaluator then configured the test client to send an authentication certificate issued by the Sub CA with no BasicConstraints. The evaluator viewed that the TOE rejected the connection.

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

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

Section 6.4 of the ST states that in the case of IPsec and TLS connections, the TOE uses X.509v3 certificates. During connection establishment, the TOE validates received authentication certificates. If the certificate appears to be valid (e.g., is properly constructed and can be decoded), the TOE then validates that it can construct a certificate path from the certificate through any intermediary CAs to a configured trusted root CA. If the path can be

constructed, the validity date and CA flag is checked in each CA certificate. If all of those checks succeed, the TOE finally checks the revocation status using OCSP of all certificates in the path. The TOE will reject any certificate for which it cannot determine validity and will reject the connection attempt.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

Section "FCS_TLSS_EXT.2.2 of the **Admin Guide** describes how to configure the HTTPS server for mutual authentication.

Component Testing Assurance Activities: None Defined

2.4.9 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2)

2.4.9.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.9.2 NDcPP22e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behaviour of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator

is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed.

Section 6.4 of the ST states that in the case of IPsec and TLS connection, the TOE uses X.509v3 certificates. During connection establishment, the TOE validates received authentication certificates. If the certificate appears to be valid (e.g., is properly constructed and can be decoded), the TOE then validates that it can construct certificate path from the certificate through any intermediary CAs to a configured trusted root CA. If the path can be constructed, the validity date and CA flag is checked in each CA certificate. If all of those checks succeed, the TOE finally checks the revocation status using OCSP of all certificates in the path. The TOE will reject any certificate for which it cannot determine validity and will reject the connection attempt.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the configuration required in the operating environment so the TOE can use the certificates. The guidance documentation shall also include any required configuration on the TOE to use the certificates. The guidance document shall also describe the steps for the Security Administrator to follow if the connection cannot be established during the validity check of a certificate used in establishing a trusted channel.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to update the ClearPass Server RADIUS and HTTPS certificates by installing certificates that are signed by a trusted certificate authority (CA). This is done via Administration -> Certificates -> Trust List in the Web UI.

Section "FIA_X509_EXT.3.1 in the **Admin Guide** provides instructions for generating a Certificate Signing Request (CSR) on ClearPass.

Section FIA_X509_EXT.2.2 in the **Admin Guide** states that if the validity of the certificate cannot be established, the default configuration is to not accept the certificate.

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

The evaluator performed these tests for both IPsec and TLS revocation checking.

Test 1:

IPSec: The evaluator first configured a test peer to send an authentication certificate that was valid and pointed to a reachable OCSP server. The evaluator viewed a successful connection. The evaluator then disabled the OCSP server and attempted to connect with the same cert. The validator viewed that the connection failed due to the unreachable revocation server.

TLS/HTTPS: The evaluator alternately configured a test client to send an authentication certificate that is valid with a reachable OCSP server. The evaluator viewed a successful connection. The evaluator then disabled the OCSP server and attempted to connect with the same cert. The validator viewed that the connection failed due to the unreachable revocation server.

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

2.4.10.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.10.2 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.

“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 “FIA_X509_EXT.1/Rev (Install Certificates)” section in the **Admin Guide** indicates that the Certificate Request Message can be generated via the Web UI by navigating to Administration -> Certificates -> Server Certificate and then clicking on the “Create Certificate Signing Request” link.

The “FIA_X509_EXT.3.1” section of the **Admin Guide** provides further details for creating a Certificate Signing Request. Upon accessing the ‘Create Certificate Signing Request’ link, the user is instructed to specify the Common Name, Organization, Organizational Unit, and Country fields. This is consistent with the selections made in the ST

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 followed the instructions in the **Admin Guide** and generated a certificate signing request as part of the IPsec testing. The evaluator downloaded the CSR and Private Key Files and confirmed that the certificate request and the private key were received.

Test 2 - The evaluator first attempted to import a server certificate without first loading a trusted CA needed to validate the server certificate. The attempt failed because the validation path in the certificate did not chain to a root CA recognized by the TOE. The evaluator then loaded a trusted CA needed to validate the server certificate into the Trust List on the TOE. The root CA was successfully added to the Trust List. The evaluator attempted to import the server certificate again. This time the server certificate was successfully validated and imported.

2.5 Security management (FMT)

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

2.5.1.1 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.

The TOE is not distributed.

Section 6.6 of the ST states that the TOE provides function to query the version and upgrade the software embedded in the TOE appliance. The TOE supports updating the TOE software using the Web UI. From the Web UI, an administrator can identify available updates and upgrades, download, and install or re-install them.

Subsequently updates and upgrades would be identified as 'installed' or 'install error' indicating there was a problem with the installation. If the update server is not accessible, the administrator can also import updates. Of course, this requires that the administrator has access to the update (e.g., previous download, access update server from an alternate machine) and can import it directly into the TOE. When installing updated software, digital signatures are used to authenticate the update to ensure it is the update intended and originated by Aruba.

The TOE supports automatic checking for updates when authenticated to the HPE Passport system. The TOE obtains credentials for the HPE Passport system from an administrator and uses those credentials to authenticate to the HPE Passport system to check for newer versions of the TOE that may be available. Upon detecting that a newer version of the code is available the TOE informs the administrator through a message presented on the software updates page of 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 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 TOE is not distributed. See TSS activity above.

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.

Testing in FIA_UIA_EXT.1.1 test 3 demonstrated no actions other than viewing the login banner were available prior to authentication by the Security Administrator. After logging in as the admin user the evaluator demonstrated that the automatic checking of updates could be enabled and disabled. The use of the automatic checking of updates is demonstrated as part of FPT_TUD_EXT.1.

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

2.5.2.1 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).

The TOE is not distributed.

Section 6.1 of the ST states that there are three locations in the Web UI where audit records are stored and can be viewed: Access Tracker, Audit Viewer, and Event Viewer.

By default, the Access Tracker and Audit Viewer store the logs for 7 days after which time they will be deleted automatically. The automatic clean up period can be configured by the administrator to be longer or shorter as may be necessary for a given deployment. The Audit Viewer storage can be configured via the cleanup parameter “Old Audit Records Cleanup Interval”. The Access Tracker storage can be configured via the cleanup parameter “Cleanup interval for Session Log details in database”. The Event Viewer records are stored for seven (7) days after which time they will be deleted automatically. There is no user configurable setting to modify the Event Viewer log storage.

The number and size of log files may be specified based on observed logging levels. The default number of log files is 12 and the default size of each log file is 50MB.The specific capacity of the audit storage is dependent on the disk drive capability of the TOE. The default disk capacity has been designed so that in a typical deployment the available space will not be exhausted within the default retention periods. Disk usage settings will notify the administrator if the system is running with low disk space.

The TOE can also be configured to send audit records to a trusted third-party SYSLOG server in the operational environment. The TOE can be configured to use TLS to protect the communication channel between itself and the remote SYSLOG server.

The TOE is a standalone TOE that stores audit data locally and transfers audit data to an external syslog server periodically. ClearPass does not transfer syslog messages in real time. Messages are queued to a syslog buffer that then transfers all messages to the syslog server every 120 seconds. This value may be reduced to a minimum of every 30 seconds, but will default to every 120 seconds.

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.

Section "Configure Audit Export" in the **Admin Guide** states ClearPass has limited storage space to retain logs. It is recommended to export all audit logs to an external source. The recommended process to accomplish this is via syslog export. This section describes how to configure the TOE to communicate with a syslog server via the System Level tab in the Web UI. The IP address of the external syslog server that will receive audit messages from ClearPass is specified, along with all the appropriate audit events to be sent. ClearPass does not transfer syslog messages in real time.

Section "FAU_STG_EXT.1" in the **Admin Guide** states messages are queued to a syslog buffer that then transfers all messages to the syslog server every 120 seconds. This value may be reduced to a minimum of every 30 seconds, but will default to every 120 seconds. The potential delay in message queue and receipt by the remote server should be noted to comply with Common Criteria evaluated settings.

Local audit records are stored for seven (7) days prior to automatic cleanup (deletion). The settings can be adjusted by modifying the value on the Cleanup Intervals tab. The Access Tracker events can be modified by adjusting the Cleanup interval for Session log details in the database parameter (default value is 7 days). The general audit (such as the Accounting) events can be modified by adjusting the Old Audit Records cleanup interval parameter (default value is 7 days). Event Viewer records are stored for seven (7) days prior to automatic cleanup. There is no user configurable setting to modify the Event Viewer log storage. Audit records that exceed cleanup intervals will be deleted from the file system and the space reclaimed to write new audit events. The number and size of log files may be specified based on observed logging levels. The default number of log files is 12 and the default size of each log file is 50MB. The number and size limits apply to all log file settings. Modifying these values will affect the log files that contain information created by RADIUS, Policy, and other services. Reducing the capacity may decrease the information available to less than seven (7) days. Increasing may cause issues with system free disk space thresholds.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPsec using the "Create IPsec Tunnel" interface in the Web UI between ClearPass and a remote device such as the syslog server using the "Create IPsec Tunnel" dialog box in the Web UI. This dialog box identifies a field to enter the remote IP address of the syslog server that the TOE will be connecting to as well as fields for entering the encryption algorithms and the type of authentication that will be used.

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 '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. 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: Testing in FIA_UIA_EXT.1.1 test 3 demonstrated no actions other than viewing the login banner were available prior to authentication by the Security Administrator.

Test 2: The configuration of the transmission of audit data to an external syslog server by an authenticated Security Administrator was demonstrated as part of FTP_ITC.1 test 1.

Test 3: Testing in FIA_UIA_EXT.1.1 test 3 demonstrated no actions other than viewing the login banner were available prior to authentication by the Security Administrator.

Test 4: The configuration of the transmission of audit data to an external syslog server by an authenticated Security Administrator was demonstrated as part FTP_ITC.1 test 1. Before configuring, the current configuration can be determined.

2.5.3 MANAGEMENT OF SECURITY FUNCTIONS BEHAVIOUR (NDCPP22E:FMT_MOF.1/MANUALUPDATE)

2.5.3.1 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 "AGD_OPE.1" section in the **Admin Guide** indicates that updates can be obtained from the Aruba support website and provides the instructions for manually installing the update via the Web UI by navigating to Administration > Agents and Software Updates > Software Updates to install patches. To manually install patches, the patch must first be loaded to the ClearPass server by clicking the button 'Import Updates' under 'Firmware & Patch Updates'. The interface box will upload the patch to the appropriate directory for installation. Internet connected systems may download the patches through 'Firmware & Patch Updates' section in the Web UI by clicking the 'Download' button to download the patch, then 'Install' to install the patch. Most patches will require a reboot once installed.

Section "FPT_TUD_EXT.1.3" in the **Admin Guide** states that ClearPass makes use of a digital signature whenever updates/upgrades are applied to the system, regardless of the package size or intent. When a new package is to be installed on ClearPass it will be initially be loaded to the server. Package signatures are verified after the package is loaded, but prior to the installation process. The signature is verified using a locally stored copy of the public key. IF the cryptographic signatures are identical then the update process is allowed to proceed. If the signatures do not match, the package update will fail with an error message indicate that the package has failed validation prior to installation.

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.

Testing in FIA_UIA_EXT.1.1 test 3 demonstrated no actions other than viewing the login banner were available prior to authentication by the Security Administrator.

The successful manual update by the authenticated Security Administrator was demonstrated in FPT_TUD_EXT.1.

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

2.5.4.1 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 6.5 of the ST describes the TOE administrative interfaces which include a command-line interface (CLI) that can be accessed over the network using SSH or locally using the serial interface and a Web UI that can be accessed using a web browser via TLS/HTTPS. This section describes the security functions available to an authorized administrator and indicates that none of these functions are available to any user before being identified and authenticated.

Section 6.4 of the ST indicates that in order for an administrative user to access the TOE (i.e., to perform any functions except to see a configured login banner, an administrative user account must be created for the user with an assigned privilege level.

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 ST indicates that the TOE restricts access to manage TSF data that can affect security functions of the TOE to authorized administrators. The TSF data manipulating functions and the corresponding configuration information are identified or referenced throughout this AAR with the requirement to which they apply.

The TOE defines an administrator role that can be assigned more granular privileges via defined privilege levels. Each time a new administrative user is defined a user identifier, username, password, and privilege level must be assigned. There are a number of pre-defined privilege levels (e.g., Super Administrator, Network Administrator) while additional privilege levels can be defined by the TOE user as may be needed for a specific deployment.

The **Admin Guide** refers to the "Managing Admin Users" section of the *ClearPass Policy Manager User guide* which describes how to create and modify administrator accounts. A web link to this guide is provided in the introductory section on page 2 of the **Admin Guide**.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** provides instructions for the administrator to configure and maintain the trust store including importing a new certificate or generating a new Certificate Signing Request (CSR).

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

No separate testing for FMT_MTD.1/CoreData is required as the management functions have been exercised throughout testing of other SFRs. Refer to FMT_SMF.1 for the specific mapping.

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

2.5.5.1 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 6.3 of the ST states the TOE generates certificate requests and validates the CA used to sign the certificates.

Section 6.3 of the ST states that the TOE's implementation of SSHv2 supports both public-key and password-based authentication; and packets are limited to 256K bytes. SSH public key authentication supports the ssh-rsa and ecdsa-sha2-nistp256 algorithms.

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 "FIA_X509_EXT.1/Rev (Install Certificates)" section in the **Admin Guide** indicates that the Certificate Request Message can be generated via the Web UI by navigating to Administration -> Certificates -> Server Certificate and then clicking on the "Create Certificate Signing Request" link.

The "FIA_X509_EXT.3.1" section of the **Admin Guide** provides further details for creating a Certificate Signing Request. Upon accessing the 'Create Certificate Signing Request' link, the user is instructed to specify the Common Name, Organization, Organizational Unit, and Country fields. This is consistent with the selections made in the ST.

The "FCS_SSHS_EXT.1.2" section in the **Admin Guide** provide instructions for configuring SSH public key authentication using ssh-rsa and ecdsa-sha2-nistp256. This is consistent with the public key algorithms specified in FCS_SSHS_EXT.1.5 in the ST. No administrator settings are available to configure.

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.

Testing in FIA_UIA_EXT.1.1 test 3 demonstrated no actions other than viewing the login banner were available prior to authentication by the Security Administrator.

The successful management of crypto keys by an authorized Security Administrator was performed as a necessary set up for FIA_X509_EXT.1.

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

2.5.6.1 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 6.5 of the ST identifies all of the security management functions specified in FMT_SMF.1. The TOE administrative interfaces consist of network-based interfaces and a serial terminal-based interface. A command-line interface (CLI) can be accessed over the network using SSH or locally using the serial interface. The Web UI can be accessed using a web browser via TLS/HTTPS. The Web UI is the primary administrative interface, while many of the administrator commands are also available via the CLI. A list of which functions are provided by the CLI and which are provided by the Web UI is included.

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

See TSS Assurance Activities.

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

The management functions defined by FMT_SMF.1 have been addressed throughout testing. See mapping above.

2.5.7 Specification of Management Functions (AUTHSRVEP10:FMT_SMF.1(1))

2.5.7.1 AUTHSRVEP10:FMT_SMF.1.1(1)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: See NDcPP22E FMT_SMF.1.

See NDcPP22E:FMT_SMF.1.

Component Guidance Assurance Activities: See NDcPP22E FMT_SMF.1.

See NDcPP22E:FMT_SMF.1.

Component Testing Assurance Activities: See NDcPP22E FMT_SMF.1.

See NDcPP22E:FMT_SMF.1.

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

2.5.8.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.8.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.8.3 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 6.5 of the ST states the TOE defines an administrator role that can be assigned more granular privileges via defined privilege levels. Each time a new administrative user is defined a user identifier, username, password, and privilege level must be assigned. There are a number of pre-defined privilege levels (e.g., Super Administrator, Network Administrator) while additional privilege levels can be defined by the TOE user as may be needed for a specific deployment.

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.

As documented throughout this AAR (in particular, refer to NDCPP22E:FIA_UIA_EXT.1.2), the guidance contains instructions for managing all aspects of the TOE both locally and remotely. The Web UI is the primary administrative interface, while the CLI provides a limited set of administrative functions. Instructions are provided specifically to manage access to these available management interfaces. The Web UI is accessed using a web browser via TLS/HTTPS; and the CLI can be accessed locally via the serial ports or remotely using SSH.

Section “FMT_SMR.2.3” in the **Admin Guide** indicates that once Common Criteria Mode is enabled, the list of algorithms and ciphersuites available for use is limited to those specified within the Security Target. Most modern web browsers support the available ciphers without further configuration. The ClearPass console access does not require further changes to access it in this mode. SSH clients that are not configured to support only FIPS 140-2

approved cryptographic algorithms will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

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 available TOE interfaces, including local CLI, SSH and Web UI have been used throughout testing.

2.6 Protection of the TSF (FPT)

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

2.6.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.1.2 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 6.6 of the ST states that the TOE does not offer any functions that will disclose to any user a plain text password. Furthermore, locally defined passwords are not stored in plaintext form, they are stored hashed with PBKDF2 (1,000 iterations).

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.6.2.1 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 6.6 of the ST states that the TOE does not offer any functions that will disclose to any users a stored cryptographic key. Keys are generated during system bootstrapping and not exposed to users or administrators.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.6.3.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.3.2 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 6.6 of the ST explains that the TOE is a hardware appliance that includes a hardware-based real-time clock. The TOE's embedded OS manages the clock and exposes administrator clock-related functions. The TOE can be configured to periodically synchronize its clock with a time server, but the TOE can only ensure its own reliability and not that of an external time mechanism. Note that the clock is used primarily to provide timestamp for audit records, but is also used to supporting timing elements of cryptographic functions, certificate validity checks, session timeouts, and unlocking of administrator accounts locked as a result of authentication failure.

The claim '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.

In the "Configure System Time" section of the **Admin Guide** provides instructions for configuring the system time manually via the Web UI by navigating to Administration -> Server Manager -> Server Configuration and selecting the link 'Set Date & Time' in the upper right corner. The time can also be set by using the option to 'Synchronize time with NTP server'. NTP servers must support NTPv4 to work. The WebUI allows a minimum of one (1) NTP server to be used, but it is recommended to specify at least three (3) NTP servers. The WebUI allows the specification of 1-5 NTP servers. When configuration is performed with NTP the communication between appliance and NTP server should be configured to a secure key and the SHA-1 hash algorithm to ensure the communication is not modified. When the date and/or time are modified, the system will restart services and require a re-login to the UI. The NTP service does not accept multicast or broadcast NTP information, there are no configuration options to change this behavior.

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

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

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

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

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

Test 1: The evaluator set the local clock from the Web UI and observed the time change and corresponding audit record.

Test 2: The evaluator configured the TOE to synchronize time with an NTP server and observed the time change and corresponding audit record.

2.6.4 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.6.4.1 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 6.6 of the ST describes the self-tests. The TOE includes a number of built in diagnostic tests that are run during start-up to determine whether the TOE is operating properly. An administrator can configure the TOE to reboot or to stop, with errors displayed, when an error is encountered. When configured, the power-on self-tests comply with the FIPS 140-2 requirements for self-testing. The module performs Cryptographic algorithm known answer tests, firmware integrity tests using RSA signature verification and conditional self-tests for PRNG, Pair-wise consistency tests on generation of RSA keys, and a Firmware load test (RSA signature verification). Upon failing any of its FIPS mode power-on self-tests, the TOE will refuse to boot.

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 "FPT_TST_EXT.1 (self tests) in the **Admin Guide** states that ClearPass will execute self-tests on the cryptographic core when operating in Common Criteria mode. These tests are also executed as part of the FIPS operating mode. To ensure the integrity of the module and the correctness of the cryptographic functionality at start up, self-tests are run. In the event of a self-test error, the module will log the error and will halt, resulting in a failure to boot ClearPass. The module must be initialized into memory to resume function.

Power-on self-tests are executed automatically when the module is loaded into memory. The module verifies the integrity of the runtime executable using a HMAC-SHA1 digest computed at build time. If the fingerprints match, the power-up self-tests are then performed. If the power-up self-test is successful, a flag is set to place the module in FIPS mode. If any component of the power-up self-test fails, an internal global error flag is set to prevent subsequent invocation of any cryptographic function calls. Any such power-up self-test failure is a hard error that can only be recovered by reinstalling the module2. The power-up self-tests may be performed at any time by reloading the module. Additionally, the pair-wise consistency tests are run as a conditional test each time a key pair is generated.

The TOE is not distributed.

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.

During a reboot of the TOE, the evaluator confirmed that the TOE performed self tests to verify the firmware integrity and the cryptographic functions. The output of these tests was captured in a putty log and indicates that they were successful. The firmware integrity test passed and all other tests were successfully completed with no errors.

2.6.5 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.6.5.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.5.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.5.3 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 6.6 of the ST states that the TOE provides function to query the version and upgrade the software embedded in the TOE appliance. The TOE supports updating the TOE software using the Web UI. From the Web UI, an administrator can identify available updates and upgrades, download, and install or re-install them. Subsequently updates and upgrades would be identified as ‘installed’ or ‘install error’ indicating there was a problem with the installation. If the update server is not accessible, the administrator can also import updates. Of course, this requires that the administrator has access to the update (e.g., previous download, access update server from an alternate machine) and can import it directly into the TOE. When installing updated software, digital signatures are used to authenticate the update to ensure it is the update intended and originated by Aruba.

The TOE supports automatic checking for updates when authenticated to the HPE Passport system. The TOE obtains credentials for the HPE Passport system from an administrator and uses those credentials to authenticate to the HPE Passport system to check for newer versions of the TOE that may be available. Upon detecting that a

newer version of the code is available the TOE informs the administrator through a message presented on the software updates page of the Web UI.

Signing and verifying the update/upgrade images uses a cryptographic digest function. A 2048-bit RSA keypair (self-signed) is generated and the binary image is signed using the private key. The public key is shipped with the TOE and is used for verification of the signed.tar file. The tar file contains the signature and binary image (zip of binary + metafile). Once the tar file is extracted the TOE verifies whether the signature of the binary image and the extracted signature match. If it matches, verification is successful.

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

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

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

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

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

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

Section “FPT_TUD_EXT.1.3” in the **Admin Guide** states that ClearPass makes use of a digital signature whenever updates/upgrades are applied to the system, regardless of the package size or intent. When a new package is to be installed on ClearPass it will initially be loaded to the server. Package signatures are verified after the package is loaded, but prior to the installation process. The signature is verified using a locally stored copy of the public key. If

the cryptographic signatures are identical then the update process is allowed to proceed. If the signatures do not match, the package update will fail with an error message indicating that the package has failed validation prior to installation.

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 product and then installed an update. The signature verified and the update was successfully installed. The evaluator displayed the version again and confirmed that the new version was displayed.

The TOE also supports automatic checking of updates. The evaluator first enabled the automatic checking of updates. The evaluator provided the appropriate credentials so that the automatic update checking could begin and viewed that there were updates being installed. The evaluator initiated the outstanding updates found by the automatic checker and viewed that the updates were successful.

Test 2 – a). The evaluator attempted to load an image onto the TOE that has been modified with a hex editor. The TOE checks the integrity of the image after the download. The TOE correctly detects an invalid image file and rejects the download.

b). The evaluator loaded an image onto the TOE without a signature. The TOE checks the integrity of the image after the download. The TOE correctly detects an invalid image file and rejects the download.

c). The evaluator loaded an image onto the TOE with a bad signature. The TOE checks the integrity of the image after the download. The TOE correctly detects an invalid image file and rejects the download.

After the TOE rejects an update as illegitimate, the version does not change.

2.7 TOE ACCESS (FTA)

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

2.7.1.1 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 6.7 of the ST states the TOE is configured by an administrator to set a session timeout. A session (local console or remote SSH or Web/HTTPS) that is inactive (i.e., no commands issuing from the remote client) for the defined timeout value will be terminated. Upon exceeding the session timeout, the TOE logs the user off.

The user will be required to login in after any session has been terminated due to inactivity or after voluntary termination. Of course, administrators can logout of local or remote sessions at any time.

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

Section "FTA_SSL.3/FTA_SSL.4/FTA_SSL_EXT.1.1" in the **Admin Guide** provides instructions for configuring the inactivity time period for remote administrative session termination. For both SSH CLI and Web UI remote sessions, this is configured through the Web UI by navigating to Administration > Server Manager > Server Configuration and then selecting 'Cluster-Wide Parameters'. For remote administrative session termination, the timeout value is set for 'CLI Session Idle Timeout' and 'Admin Session Idle Timeout'.

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

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

Test 1: The evaluator configured the TOE for 5, 7 and 10 minute timeout periods and verified that the TOE session required re-authentication at the configured timeout period. This test was repeated on the SSH CLI and Web UI.

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

2.7.2.1 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 6.7 of the ST states the TOE provides the function to logout (i.e., terminate) both local and remote user sessions as directed by the user.

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

Section "FTA_SSL.3/FTA_SSL.4/FTA_SSL_EXT.1.1" in the **Admin Guide** states that termination of local console or CLI (SSH) sessions by the administrator is accomplished by entering the "exit" command to log out before idle

session timeout. WebUI screens may be triggered from the Menu list in the upper right corner and selecting "Logout".

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

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

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

Test 1 - The evaluator established a local session and manually exited per the command identified in the Guidance. The session was terminated and a logout audit record was logged.

Test 2 - The evaluator established a remote session and manually exited per the command identified in the Guidance. The session was terminated and a logout audit record was logged.

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

2.7.3.1 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 6.7 of the ST states the TOE is configured by an administrator to set a session timeout. A session (local console or remote SSH or Web/HTTPS) that is inactive (i.e., no commands issuing from the remote client) for the defined timeout value will be terminated. Upon exceeding the session timeout, the TOE logs the user off.

The user will be required to login in after any session has been terminated due to inactivity or after voluntary termination. Of course, administrators can logout of local or remote sessions at any time.

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

Section "FTA_SSL.3/FTA_SSL.4/FTA_SSL_EXT.1.1" in the **Admin Guide** provides instructions for configuring the inactivity time period for remote administrative session termination. For both SSH CLI and Web UI remote

sessions, this is configured through the Web UI by navigating to Administration > Server Manager > Server Configuration and then selecting 'Cluster-Wide Parameters'. For local administrative session termination, the timeout value is set for 'Console Session Idle Timeout'.

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 set the console time period to 5, 7 and 10 minutes and observed that the session terminated after each defined time period.

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

2.7.4.1 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 6.5 of the ST describes the methods of access (local and remote) available to the TOE administrator. The TOE administrative interfaces consist of network-based interfaces and a serial terminal-based interface. A command-line interface (CLI) can be accessed over the network using SSH or locally using the serial interface. The WebGUI can be accessed using a web browser via TLS/HTTPS. The WebGUI is the primary administrative interface, while many of the administrator commands are also available via the CLI.

Section 6.7 of the ST states that the TOE displays an administrator-configured login banner before authentication. In all cases (console, SSH, and web interface), the login banner is displayed on the display or login screen before an administrative user session is established.

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

Section "FTA_TAB.1" in the **Admin Guide** provides instructions for configuring an access banner by navigating to Administration > Server Manager > Server Configuration and selecting the option for 'Cluster-Wide Parameters'. The 'Login Banner Text' field is modified to include the desired text. This text will be applied to the local console, Web UI and SSH login events prior to the user 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 configured a Login banner on the TOE and then verified that the banner displayed appropriately for each login method.

2.7.5 TOE Session Establishment (AUTHSRVEP10:FTA_TSE.1)

2.7.5.1 AUTHSRVEP10:FTA_TSE.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 all of the attributes on which a user session can be denied are specifically defined.

The ST, section 6.7, states that the TOE can reject authentication requests based on time of day, account status, location, and role mapping.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it contains guidance for configuring each of the attributes identified in the TSS.

The section "FTA_TSE.1" section in the **Admin Guide** provides instructions for configuring the attributes upon which a user session can be denied which are: time of day, account status, location, and role mappings. These attributes are configured via the Web UI by navigating to Configuration -> Enforcement -> Policies. Examples of RADIUS policies that deny access based on these criteria are provided.

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

Test 1: The evaluator shall successfully establish a user session. The evaluator shall follow the operational guidance to configure the system so that that user's access is denied based on a specific value of an attribute. The evaluator shall then attempt to establish a session in contravention to the attribute setting while still providing valid authentication data. The evaluator shall observe that the access attempt fails. The evaluator shall repeat this test for each attribute indicated by the ST author.

The evaluator followed the guidance to configure the RADIUS server settings on the TOE Such that that user's access would be denied based on specific values of the following attributes: account status, time of day, location and role mapping. In each case the evaluator established a successful session with correct values for the attributes. Likewise, in each case, the evaluator found the connection failed when the incorrect values for the attributes were used.

2.8 Trusted path/channels (FTP)

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

2.8.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.3 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 6.8 of the ST states remote connections to trusted third party syslog servers are supported for exporting audit records. Communication with those external audit servers is protected via IPsec. Additionally, the TOE can sync to an external NTP server over a protected IPsec tunnel. This matches the requirement.

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.

For NDcPP22E:FTP_ITC.1, the ST identifies the authorized IT entities in the TOE as the following:

- Audit server –the TOE exports audit data to a syslog server (**IPsec**)
- NTP server – the TOE synchronizes its clock with an NTP server (**IPsec**)

Section "FMT_SMR.2.3" in the **Admin Guide** indicates that once Common Criteria Mode is enabled, available algorithms and ciphersuites are limited to those specified within the Security Target (ST).

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. The *Create IPsec Tunnel* dialog box provides a field to enter the remote server IP address and a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Method. The encryption algorithms and hash algorithms available can be selected only one time and applied across the Security Association (SA). ClearPass uses it HTTPS certificate for IPSec. This section also includes recovery instructions in the event that an IPsec VPN is unexpectedly dropped.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to update the ClearPass Server RADIUS and HTTPS certificates by installing certificates that are signed by a trusted certificate authority (CA). This is done via Administration -> Certificates -> Trust List in the Web UI.

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 and Test 2– The evaluator configured the TOE to send audit records to an external syslog server and configured an IPsec tunnel to protect the communication between the TOE and the syslog server. The evaluator confirmed that the communication was initiated by the TOE. The connection was successful and all audit records were transmitted to syslog encrypted. This test was repeated for the connection between the TOE and an external NTP server.

Test 3 – The evaluator has packet captures of each communications path to ensure it is encrypted.

Test 4 –The evaluator interrupted the communications path by physically disconnecting the network between the TOE and the remote server first for 30 seconds (mac layer disruption) and then the second time for 5 minutes

(application layer disruption). After the 30 second disruption, the connection resumed with the IPsec tunnel intact and all communications resumed in an encrypted form. After the 5 minute disruption, the connection resumed with a new IPsec connection and all communications continued to be encrypted. This was verified using a packet capture. This test was repeated with both the syslog server and the NTP server with the same results.

2.8.2 Inter-TSF Trusted Channel (AUTHSRVEP10:FTP_ITC.1(1))

2.8.2.1 AUTHSRVEP10:FTP_ITC.1.1(1)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.2 AUTHSRVEP10:FTP_ITC.1.2(1)

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.3 AUTHSRVEP10:FTP_ITC.1.3(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, for all communications with authorized IT entities identified in the requirement, each communications mechanism is identified in terms of the allowed protocols for that IT entity. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

Section 6.8 of the ST states the TOE uses either RADSEC or IPsec to communicate with associated NAS servers for RADIUS requests and responses. This matches the requirement.

Section 6.8 of the ST states remote connections to trusted third party syslog servers are supported for exporting audit records. Communication with those external audit servers is protected IPsec. Additionally, the TOE can sync to an external NTP server over a protected IPsec tunnel. This matches the requirement.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing and re-establishing the allowed protocols with each authorized IT entity.

For AUTHSRVEP10:FTP_ITC.1(1), the ST identifies the authorized entities in the TOE as the following:

- Network Access Devices (NAS) -- These devices forward authentication credentials to the authentication server (ie. the TOE). The TSF shall initiate the communication via the trusted channel for responses to RADIUS Access-Request messages received from the NAS. (EAP-TLS, IPsec, RADSEC)

Section "FMT_SMR.2.3" in the **Admin Guide** indicates that once Common Criteria Mode is enabled, available algorithms and ciphersuites are limited to those specified within the Security Target (ST).

Section "FTP_ITC.1" in the **Admin Guide** describes how to configure the ClearPass RADIUS service via the Web UI. The RADIUS service is configured via the Service tab in Configuration > Services, while the EAP-TLS authentication method is configured via the Authentication tab.

Section "Add Network Access Devices" in the **Admin Guide** describe how to add network access devices to the system before conducting RADIUS authentication events, how to configure a RADIUS shared secret key/password and how to enable RadSec.

The sections "FTP_ITC.1" and "Adding Network Access Devices" in the **Admin Guide** indicate when RADIUS communication is not functioning correctly, it is typically due to either an incorrect address specified, or the shared secret is not correctly entered between the devices. When RADIUS has been communicating correctly between two hosts and unexpectedly stops, the service should be re-validated on both systems. The user is instructed to ensure that the IP address(es) of all devices are still correctly specified and to re-enter shared secret passwords on both devices. Additionally, it should be verified that no network control device, such as a firewall or IPsec VPN tunnel, is preventing the network traffic from reaching both devices correctly.

The "FCS_IPSEC_EXT.1" section in the **Admin Guide** describes how to configure IPSec which uses both RSA and ECDSA. The *Create IPsec Tunnel* dialog box provides a field to enter a remote IP address and a drop down menu from which either Preshared Keys or Certificates can be selected for the Authentication Method. The encryption algorithms and hash algorithms available can be selected only one time and applied across the Security Association (SA). ClearPass uses it HTTPS certificate for IPSec. This section also includes recovery instructions in the event that an IPsec VPN is unexpectedly dropped.

Section "FCS_EAP-TLS_EXT.1" in the **Admin Guide** confirms that when operating in Common Criteria mode, the TOE will only use the cipher suites specified for TLS, but qualifies that with a list of cipher suites that are not available for RADIUS sessions.

The "FIA_X509_EXT.1/Rev (Install Certificates)" section of the **Admin Guide** describes how to update the ClearPass Server RADIUS and HTTPS certificates by installing certificates that are signed by a trusted certificate authority (CA). This is done via Administration -> Certificates -> Trust List in the Web UI.

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

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

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.

Test 3: The evaluator shall ensure, for each communication channel with a NAS, the channel data uses the appropriate identified protocols.

Test 4: The evaluators shall, for each protocol associated with each NAS tested during test 1, physically interrupt an established connection. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

Further assurance activities are associated with the specific protocols based on the selection that is chosen in FTP_ITC.1.1.

Test 1 and Test 2 – The evaluator configured the TOE to receive authentication requests from an external NAS device and to protect the communications between the TOE's Radius server and the NAS device with IPsec. The connection was successful and all audit records were transmitted encrypted. This test was repeated with RadSec configured to protect the communication between the TOE and NAS devices. The evaluator confirmed that the IPSec connection was initiated by the TOE.

Test 3 – The evaluator has packet captures of each communications path to ensure it is encrypted with IPsec or RadSec.

Test 4 – IPSec: The ipsec channel testing was met in FTP_ITC.1.3.

RadSec: The evaluator interrupted the RadSec communications path by physically disconnecting the network between the TOE and the NAS device first for 30 seconds (mac layer disruption) After the 30 second disruption, a new RadSec connection was established. This was verified using a packet capture. Due to the how quick the radsec connection occurs, the mac layer disruption also hits the application layer disruption.

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

2.8.3.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.3 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 6.8 of the ST states that the TOE uses SSH and TLS/HTTPS to provide a trusted path for remote management interfaces to protect the communication from disclosure and modification. The TOE provides a trusted path for its remote administrative users accessing the TOE via the Ethernet ports provided on the TOE using either a command line interface using SSH or Web-based graphical user interface using TLS/HTTPS. Local console access via a serial port is also supported for command line access. However, this access is protected by physical protection of the serial interface along with the TOE itself. This discussion matches 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.

For NDcPP21:FTP_TRP.1/Admin, the ST identifies the authorized entities in the TOE as the following:

- Web Browser – for remote administration via the GUI – (**TLS/HTTPS)**
- SSH Client – for remote administration via the CLI – (**SSH)**

Section “FMT_SMR.2.3” in the **Admin Guide** indicates that once Common Criteria Mode is enabled, available algorithms and ciphersuites are limited to those specified within the Security Target (ST). ClearPass CLI access does not require further changes to access it in this mode. SSH clients that are not configured to support only FIPS140-2 approved cryptographic ciphers will need to have ciphers re-prioritized to use the ones allowed by ClearPass or connections will not establish.

Section “FCS_TLSS_EXT.2.1” in the **Admin Guide** lists the available TLS cipher suites consistent with those identified in the ST.

Section FCS_TLSS_EXT.2.2 in the **Admin Guide** provides instructions for how to configure Web UI sessions using certificate identification through TLS mutual authentication. This is configured with the aid of a setup wizard which is available to administrators in the Web UI.

The “FIA_X509_EXT.1/Rev (Install Certificates)” section of the **Admin Guide** describes how to update the ClearPass Server RADIUS and HTTPS certificates by installing certificates that are signed by a trusted certificate authority (CA). This is done via Administration -> Certificates -> Trust List in the Web UI.

The “FCS_SSHS_EXT.1.2” section in the **Admin Guide** provides instructions for configuring SSH public key authentication.

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

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

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

Further assurance activities are associated with the specific protocols.

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

Test 1 – All available TOE remote interfaces, including local, SSH and Web UI have been used throughout testing. These protocols were fully tested as part of FCS_TLSS_EXT.2 and FCS_SSHS_EXT.1.

Test 2 – All available TOE remote interfaces, including local SSH and Web UI have been used throughout testing. These protocols were fully tested as part of FCS_TLSS_EXT.2 and FCS_SSHS_EXT.1 and demonstrated that no data for either trusted path was sent in plaintext.

3. Protection Profile SAR Assurance Activities

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

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

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

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

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

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

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

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

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

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

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

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

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

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

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

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. Section "AGD_OPE.1" and other areas of the **Admin Guide** also describe the components and features of the product which have not been subject to evaluation.

Section "AGD_OPE.1" in the **Admin Guide** states that cryptographic limits ensure that the ClearPass appliance is configured to use only approved ciphers and algorithms. Without these configurations, there are additional capabilities that are capable of being used that were not evaluated as part of the Common Criteria process. To ensure that only approved cryptographic functionality is enabled, ClearPass must be configured to use both FIPS140-2 and Common Criteria Mode when operating to limit functionality to evaluated capabilities.

Section "AGD_OPE.1" in the **Admin Guide** indicates that updates can be obtained from the Aruba support website and provides the instructions for manually installing the update via the Web UI by navigating to Administration > Agents and Software Updates > Software Updates to install patches. To manually install patches, the patch must first be loaded to the ClearPass server by clicking the button 'Import Updates' under 'Firmware & Patch Updates'. The interface box will upload the patch to the appropriate directory for installation. Internet connected systems may download the patches through 'Firmware & Patch Updates' section in the Web UI by clicking the 'Download' button to download the patch, then 'Install' to install the patch. Most patches will require a reboot once installed.

Section "FPT_TUD_EXT.1.3" in the **Admin Guide** states that ClearPass makes use of a digital signature whenever updates/upgrades are applied to the system, regardless of the package size or intent. When a new package is to be installed on ClearPass it will initially be loaded to the server. Package signatures are verified after the package is loaded, but prior to the installation process. The signature is verified using a locally stored copy of the public key. If the cryptographic signatures are identical then the update process is allowed to proceed. If the signatures do not match, the package update will fail with an error message indicating that the package has failed validation prior to installation.

3.2.2 Preparative Procedures (AGD_PRE.1)

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

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

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

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

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

The preparative procedures must

a) include instructions to provide a protected administrative capability; and

b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

The evaluation team had the following documents to use when configuring the TOE:

- Common Criteria Configuration Guidance Aruba ClearPass Policy Manager 6.11, Version 5.0, March 2023 (**Admin Guide**)

In some instances, the document referenced general Aruba ClearPass manuals which the evaluation could find on the Aruba web site. The completeness of the documentation is addressed by their use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

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

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

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

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

3.3.2 TOE CM Coverage (ALC_CMS.1)

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

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

See 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 evaluation team exercised the independent tests specified in the *'collaborative Protection Profile for Network Devices', Version 2.2e, 23 March 2020/'Network Device Collaborative Protection Profile (NDcPP)/Application Software Protection Profile (App PP) Extended Package (EP) for Authentication Servers', Version 1.0, 07 August 2015 (NDcPP22e/AUTHSRVEP10)*) against the evaluated configuration of the TOE. The following diagram indicates the test environment.

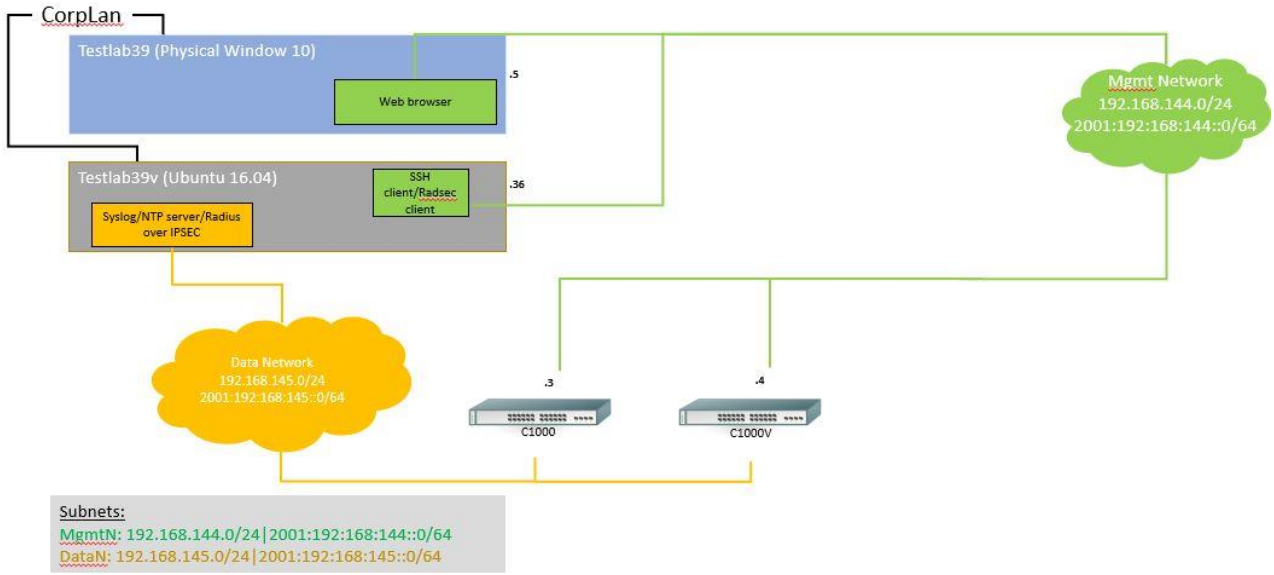

Figure 1 Test Setup

TOE Platforms:

- C1000
- C1000V

Supporting Products:

- Windows 10, 64-bit
  - Standard Windows utilities (e.g., notepad, snip tool)
  - Putty release 0.74 (used to connect to the Linux devices attached to the TOE devices to facilitate session logging)
  - HxD (Hexeditor) version 2.4.0.0 (used to corrupt valid software image)
  - Wireshark version 4.0.3 (used to examine network packets)
  - Microsoft Hyper-V (part of Windows)
- Ubuntu version 16.04.7, 64-bit
  - Standard Linux commands (e.g., cat, grep, awk)
  - OpenSSL version 1.0.2g-fips (used to generate certificates)
  - Strongswan version U5.3.5 (used as ipsec peer)
  - Stunnel version 5.30 (used for tls client testing)
  - Radsecproxy version 1.8.1 (used for RadSec testing)
  - Eapol_test version 2.6 (for eap-tls and radius testing)
  - tcpdump (comes with Ubuntu – used to generate packet capture files for network traffic)
  - rsyslog version 8.16.0 (used to accept syslog logs)
  - ntpd version ntpd 4.2.8p4
  - robot-detect tool
  - Evaluator developed test scripts

Supporting Software:

- SSH Client – Putty version 6.2
- SSH Client – SecureCRT version 5.1.2
- Big Packet Putty version 6.2
- Wireshark version 1.10.0 (3 instances identified above)
- Nmap version 6.25

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 components7 that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

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

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

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

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

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

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

The evaluator searched the National Vulnerability Database (https://web.nvd.nist.gov/vuln/search), Vulnerability Notes Database (http://www.kb.cert.org/vuls/), Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities), Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories), Exploit / Vulnerability Search Engine (http://www.exploitsearch.net), SecurITeam Exploit Search (http://www.securiteam.com), Tenable Network Security (http://nessus.org/plugins/index.php?view=search), Offensive Security Exploit Database (https://www.exploit-db.com/) on 3/21/2023 with the following search terms: "Aruba", "HPE Aruba", "Clearpass", "C1000", "C2000", "C2010", "C2020", "C3000", "C3010", "C1000V", "Atom C2758", "Xeon E3-1240", "Xeon E-2274G", "Xeon E5-2620", "Xeon Gold 5118", "ESXi 7.0", "Xeon E-2254ML", "StrongSwan", "Apache", "radsecproxy", "OpenSSH", "OpenSSL".

3.5.2 Additional Flaw Hypotheses (AVA_VLA.1)

Assurance Activities: The following additional tests shall be performed:1.) [Conditional]: If the TOE is a TLS server and supports ciphersuites that use RSA transport (e.g. supporting TLS_RSA_WITH_* ciphers) the following test shall be performed. Where RSA Key Establishment schemes are claimed and especially when PKCS#1 v1.5* padding is used, the evaluators shall test for implementation flaws allowing Bleichenbacher and Klima et al. style attacks, including Bock et al's ROBOT attacks of 2017 in the flaw analysis. Even though Bleichenbacher's original paper is two decades old, Bock et al. found these attacks to still be effective in weakening the security of RSA key establishment in current products. Bleichenbacher and Klima et al. style attacks are complex and may be difficult to detect, but a number of software testing tools have been created to assist in that process. The iTC strongly recommends that at least one of the tools mentioned in Bock et al's ROBOT attacks of 2017 webpage or paper, as effective to detect padding oracle attacks, be used to test TOE communications channels using RSA based Key Establishment (related sources: http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf, https://eprint.iacr.org/2003/052, https://robotattack.org/). Network Device Equivalency Considerations

Since the TOE acts as a TLS server and supports ciphersuites that use RSA transport, the evaluator utilized the robot-check detection tool to check for implementation flaws allowing Bleichenbacher and Klima et al. style attacks. Both devices were found to be not vulnerable.