Assurance Activities Report

for

VMware Unified Access Gateway (UAG) 2209

Version 1.0

23 June 2023

Prepared by:

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

Prepared for:

VMware, Inc.
3401 Hillview Avenue
Palo Alto, CA 94304

The Developer of the TOE:

VMware, Inc.
3401 Hillview Avenue
Palo Alto, CA 94304

The TOE Evaluation was Sponsored by:

VMware, Inc.
3401 Hillview Avenue
Palo Alto, CA 94304

Evaluation Personnel:

Dawn Campbell
Kevin Zhang
Pascal Patin

Contents

1 Introduction

This document presents results from performing evaluation activities associated with the VMware Unified Access Gateway (UAG) 2209 evaluation. This report contains sections documenting the performance of evaluation activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in Evaluation Activities for Network Device cPP, Version 2.2, December 2019 [ND SD].

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

1.1 Applicable Technical Decisions

The NIAP Technical Decisions (TDs) that apply to [NDcPP] as well as rationale for those TDs that do not apply to this evaluation are identified in the ETR as specified by Labgram #105/Valgram #125.

1.2 Evidence

The following evidence was used to complete the evaluation activities.

[ST]	VMware Unified Access Gateway (UAG) 2209 Security Target, Version 1.0, 24 April 2023
[CCECG]	VMware Unified Access Gateway (UAG) 2209 Common Criteria (CC) Evaluated Configuration Guide Version 1.0, 24 April 2023
[ADMIN]	VMware Horizon 2209 Horizon Administration, 2022
[Deploy]	VMware Deploying and Configuring VMware Unified Access Gateway, 2022

1.3 Conformance Claims

Common Criteria Versions

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, dated: April 2017.
- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, Revision 5, dated: April 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Revision 5, dated: April 2017.

Common Evaluation Methodology Versions

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

Protection Profiles

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

2 Security Functional Requirement Evaluation Activities

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

2.1 Security Audit (FAU)

2.1.1 Audit Data Generation (FAU_GEN.1)

2.1.1.1 TSS Activities

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

[ST] Section 6.1 states that when key data is updated, the audit record also includes identifying information for the key (subject DN, issuer DN, thumbprint, and expiration).

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

The TOE is not distributed and therefore this is not applicable.

2.1.1.2 Guidance Activities

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

[CCECG] Section 5.3 Tables 6 and 7, provide sample auditable events consistent with the auditable events table in the NDcPP for the claimed SFRs as well as the claimed management functions in FMT_SMF.1. Section 5.1.3 Syslog Event Format describes each audit record format type along with a brief description of each field. The description of the fields contains the information required in FAU_GEN.1.2, and the additional information specified in the table of audit events.

The evaluator shall also make a determination of the administrative actions related to TSF data related to configuration changes. The evaluator shall examine the guidance documentation and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the cPP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative

guide are related to TSF data related to configuration changes. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

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

The administrative actions identified as auditable are:

- Resetting Passwords
- Logging in and logging out
- Configuration of log level settings
- Configuration of syslog export settings
- Setting length requirement for passwords
- Configuration of session inactivity time before session termination
- Configuration of authentication failure for FIA_AFL.1
- Generating/import of, changing, or deleting of Manage cryptographic keys
    - import and manage X.509v3 certificates to the TOE's trust store
    - generate certificate signing requests, which include key pairs
    - designate X.509v3 certificates as trust anchors
- Unlock a remote Administrator account
- Initiation of a software update
- Enable time sync
- Configuring the banner displayed prior to authentication

2.1.1.3 Test Activities

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

The evaluator collected audit records throughout testing.

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.

N/A – The TOE is not distributed.

2.1.2 User Identity Association (FAU_GEN.2)

2.1.2.1 TSS & Guidance Activities

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

2.1.2.2 Test Activities

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

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

N/A – The TOE is not distributed.

2.1.3 Protected Audit Trail Storage (FAU_STG.1)

2.1.3.1 TSS Activities

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

[ST] Section 6.1 states that with respect to the TSF, the admin.log, audit.log, and esmanager.log files contain records of security-relevant events. Each of the log files are rotated such that the exhaustion of storage for one file causes it to be archived with active auditing moving to the next file in the rotation. Once all files are filled in this manner, subsequent rotations will overwrite the oldest stored record. By default, each of the log files have a maximum file size of 10 MB and store five separate files in the rotation. No remote administrative commands exist to modify or delete log files; only the local root admin is authorized to interact with these files. A Security Administrator is therefore authorized to manually delete audit records only if they are authenticated to the TOE via the local console.

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

The TOE is not distributed and therefore this is not applicable.

2.1.3.2 Guidance Activities

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

[CCECG] Section 5.1.2 states that no remote administrative commands exist to modify or delete log files; only the local root admin is authorized to interact with these files. A Security Administrator is therefore authorized to manually delete audit records only if they are authenticated to the TOE via the local console. Since there is only one local admin, no configuration is required to restrict this.

2.1.3.3 Test Activities

The evaluator shall perform the following tests:

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

The evaluator was unable to access any services prior to authentication as a Security Administrator. Thus the evaluator was unable to modify and delete audit records.

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

The evaluator was able to log in as an authorized administrator and delete specific audit records.

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

N/A – The TOE is not distributed.

2.1.4 Protected Audit Event Storage (FAU_STG_EXT.1)

2.1.4.1 TSS Activities

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

[ST] Section 6.1 states that in the evaluated configuration, all log files are configured to transmit log data to a remote syslog server over TLS; this occurs in real-time, simultaneously with the audit records that are written locally. In the event of a syslog outage, there is no buffering mechanism that allows for synchronization between local and remote logs.

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.

[ST] Section 6.1 states that with respect to the TSF, the admin.log, audit.log, and esmanager.log files contain records of security-relevant events. Each of the log files are rotated such that the exhaustion of storage for one file causes it to be archived with active auditing moving to the next file in the rotation. Once all files are filled in this manner, subsequent rotations will overwrite the oldest stored record. Each of the log files have a maximum file size of 10 MB and store five separate files in the rotation. No remote administrative commands exist to modify or delete log files; only the local root admin is authorized to interact with these files. A Security Administrator is therefore authorized to manually delete audit records only if they are authenticated to the TOE via the local console.

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.

[ST] Section 6.1 states that the TOE is a standalone device that generates audit records that reside in its local storage.

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

[ST] The SFR has been completed with "overwrite previous audit records according to the following rule: log file rotation". The TSS Section 6.1 describes this as follows. Each of the log files are rotated such that the exhaustion of storage for one file causes it to be archived with active auditing moving to the next file in the rotation. Once all files are filled in this manner, subsequent rotations will overwrite the oldest stored record. By default, each of the log files have a maximum file size of 10 MB and store five separate files in the rotation.

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.

[ST] Section 6.1 states that all log files are configured to transmit log data to a remote syslog server over TLS in real-time.

For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how audit data transfer to the external audit server is implemented

among the different TOE components (e.g. every TOE components does its own transfer or the data is sent to another TOE component for central transfer of all audit events to the external audit server).

The TOE is not distributed and therefore this is not applicable.

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

The TOE is not distributed and therefore this is not applicable.

2.1.4.2 Guidance Activities

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

[CCECG] Section 5.2.1 describes how to configure an external syslog server for remote audit storage and refers to "Configure Syslog Server Settings" in Deploying and Configuring VMware Unified Access Gateway for additional information. Section 4.7 describes an additional setting to ensure revocation checking is performed for syslog server certificates.

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.

[CCECG] Section 5.1.2 states that in the evaluated configuration, all log files are configured to transmit log data to a remote syslog server over TLS; this occurs in real-time, simultaneously with the audit records that are written locally. In the event of a syslog outage, there is no buffering mechanism that allows for synchronization between local and remote logs.

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

The TOE does not offer any configuration options for FAU_STG_EXT.1.3 and therefore this is implicitly satisfied.

2.1.4.3 Test Activities

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

Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the

audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing. The evaluator shall verify that the TOE is capable of transferring audit data to an external audit server automatically without administrator intervention.

The evaluator established a secured session between TOE and audit server, confirming that data was not transmitted in the clear and was successfully received by the server.

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

1) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option 'drop new audit data' in FAU_STG_EXT.1.3).
2) The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option 'overwrite previous audit records' in FAU_STG_EXT.1.3)
3) The TOE behaves as specified (for the option 'other action' in FAU_STG_EXT.1.3).

The evaluator generated audit data and verified the data is stored locally. The evaluator was also able to verify that existing audit data was overwritten when file size limit was reached.

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

N/A – The TOE does not claim FAU_STG_EXT.2/LocSpace.

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.

N/A – The TOE does not claim FAU_STG_EXT.2/LocSpace.

2.2 Cryptographic Support (FCS)

2.2.1 Cryptographic Key Generation (FCS_CKM.1)

2.2.1.1 TSS Activities

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

[ST] Section 6.2 states the TOE generates asymmetric keys for TLS and X.509 certificates. The TOE generates ECC keys using P-256, P-384, and P-521 and FFC keys using ffdhe groups 2048, 3072, 4096, 6144, and 8192 in support of TLS key establishment as a client. As a server, the TOE uses P-384 for ECC keys. The TOE generates 3072-bit RSA keys as part of generating certificate signing requests per FIA_X509_EXT.3.

2.2.1.2 Guidance Activities

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites. Section 3.1 of the AGD indicates that the CC evaluated config requires use of the FIPS version of the OVF, which is automatically configured in FIPS mode.

2.2.1.3 Test 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.

Section 6.2 of [ST] ("Cryptographic Support"), Table 6 identifies the CAVP certifications verifying asymmetric key generation, as follows.

Algorithm	Tested Capabilities	Certificates
RSA schemes using cryptographic key sizes of 3072-bit that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3	Key Generation Mode: B.3.3 Properties: Modulo: 3072 Primality Tests: C.2 Public Exponent Mode: Fixed Fixed Public Exponent: 010001 Public Key Format: Standard	A1292 (OpenSSL) RSA KeyGen (FIPS186-4)

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.

Section 6.2 of [ST] ("Cryptographic Support"), Table 6 identifies the CAVP certifications verifying asymmetric key generation, as follows.

Algorithm	Tested Capabilities	Certificates
ECC schemes using "NIST curves" P-256, P-384, and P-521, that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4	Curves: P-256, P-384, P-521	A1292 (OpenSSL) and A2841 (BC-FJA) ECDSA KeyGen (FIPS186-4) ECDSA KeyVer (FIPS186-4)

Modified per TD0580.

FFC Schemes using "safe-prime" groups

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

2.2.2 Cryptographic Key Establishment (FCS_CKM.2)

2.2.2.1 TSS Activities

Modified per TD0580.

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

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

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	Administration
ECDH	FCS_SSHC_EXT.1	Audit Server
ECDH	FCS_IPSEC_EXT.1	Authentication Server

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

Section 5.2.2.2 of [ST] ("Cryptographic Key Establishment") identifies the following key establishment methods supported by the TOE:

- Elliptic curve-based key establishment schemes
- FFC Schemes using "safe-prime" groups (groups listed in RFC 7919).

These key establishment methods correspond to the key generation schemes specified in FCS_CKM.1.1. Section 6.2 states that the TOE generates ECC and FFC keys in support of TLS key establishment.

All services where the TOE acts as a TLS/HTTPS server (FCS_TLSS_EXT.1, FCS_TLSS_EXT.2) use ECDH key establishment. All services where the TOE acts as a TLS or TLS/HTTPS client (FCS_TLSC_EXT.1) can use ECDH key establishment. Services that use BC-FJA (i.e. communications with Horizon Connection Server) can also use DH key establishment, depending on the negotiated cipher suite. Section 6.7 indicates that the TOE acts as a TLS client for communications with Horizon Connection Servers, therefore FCS_TLSC_EXT.1 is implied. Additionally, ffdhe groups 2048, 3072, 4096, 6144, and 8192 are identified as the FFC groups used in support of TLS key establishment as a client. Therefore, FFC Schemes using "safe-prime" groups is implied. The specific services for client and server connections are identified in Section 6.7. The evaluator created the following table corresponding to the descriptions and determined that the description in the ST was complete.

Scheme	SFR	Service	Specific Services
Elliptic curve-based	FCS_TLSS_EXT.1 FCS_TLSS_EXT.2	Services where the TOE acts as a TLS/HTTPS server (FCS_TLSS_EXT.1, FCS_TLSS_EXT.2)	Horizon Client user authentication (XML API channel) Horizon Client media transmission (Blast channel) Remote administration

Scheme	SFR	Service	Specific Services
	FCS_TLSC_EXT.1	Services where the TOE acts as a TLS or TLS/HTTPS client (FCS_TLSC_EXT.1) can use ECDH key establishment	Remote syslog server Horizon Connection Server session establishment Horizon Agent media transmission (Blast channel)
FFC Schemes using "safe-prime" groups (groups listed in RFC 7919)	FCS_TLSC_EXT.1	Services that use BC-FJA (i.e. communications with Horizon Connection Server)	Horizon Connection Server session establishment

2.2.2.2 Guidance Activities

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites.

2.2.2.3 Test 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.

Performed in accordance with NIAP Policy Letter #5.

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.

Section 6.2 of [ST] ("Cryptographic Support"), Table 6 ("Validated Algorithm Implementations") identifies the CAVP certifications verifying SP 800-56A Revision 3 key establishment schemes, as follows.

Algorithm	Tested Capabilities	Certificates
Elliptic curve-based key establishment scheme that meet the following: NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"	KAS-ECC-SSC Sp800-56Ar3: P-256, P-384, P-521	A1292 (OpenSSL) A2841 (BC-FJA)

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.

Not applicable. The TOE does not use RSA key establishment schemes.

Modified per TD0580 (section removed).

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.

Algorithm	Tested Capabilities	Certificates
FFC Schemes using "safe-prime" groups that meet the following: 'NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography" and [groups listed in RFC 7919]]"	Safe Prime Key Generation and KAS-FFC-SSC Sp800-56Ar3 ffdhe groups 2048, 3072, 4096, 6144, and 8192	A2841 (BC-FJA)

2.2.3 Cryptographic Key Destruction (FCS_CKM.4)

2.2.3.1 TSS Activities

The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for. [Where keys are stored encrypted or wrapped under another key then this may need to be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions]). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

[ST] Section 6.2 states that the TOE includes both plaintext static keys that reside in nonvolatile memory and plaintext ephemeral keys that reside in volatile memory. Keys stored in volatile memory include the TLS pre-master secret, session key, and session authentication key. When these are no longer needed, they are destroyed through destruction of reference to the key followed by a request for garbage collection. OpenSSL does this using API function calls and BC-FJA does this by invoking the JVM garbage collector on thread termination; there are no circumstances where the claimed behavior does not apply.

The only static key maintained by the TOE is the private key portion of the TOE's TLS server certificate. This key originates when it is generated by the TSF as part of generating a certificate signing request per FIA_X509_EXT.3. When the request is generated, the private key is temporarily stored in plaintext. Once the signed certificate response is received, the certificate and private key are loaded into the password-protected Java Keystore (see ST section 6.5). The private key is then destroyed through the TOE's invocation of OS mechanisms to destroy the file system object that represents the key; there are no circumstances where this claimed behavior does not apply.

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).

[ST] Section 6.2 indicates that the only static key maintained by the TOE is the private key portion of the TOE's TLS server certificate. This key is destroyed through the TOE's invocation of OS mechanisms to destroy the file system object that represents the key.

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

The ST author selects 'destruction of reference' (for volatile memory) and selects 'the invocation of an interface' (for non-volatile memory).

[ST] Section 6.2 states that for destruction of keys in volatile memory, OpenSSL does this using API function calls and BC-FJA does this by invoking the JVM garbage collector on thread termination; there are no circumstances where the claimed behavior does not apply. This section also indicates that the only static key maintained by the TOE is the private key portion of the TOE's TLS server certificate. This key originates when it is generated by the TSF as part of generating a certificate signing request per FIA_X509_EXT.3. When the request is generated, the private key is temporarily stored in plaintext. Once the signed certificate response is received, the certificate and private key are loaded into the password-protected Java Keystore. The private key is then destroyed through the TOE's invocation of OS mechanisms to destroy the file system object that represents the key; there are no circumstances where this claimed behavior does not apply.

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 [ST] Section 6.2 identifies a single key stored in non-plaintext form. The private key portion of the TOE's TLS server certificate. This encryption method used to store this key is a password-protected Java Keystore. The BC-FJA FIPS keystore uses AES-256, SHA-512, and PBKDF2 to ensure that the stored data is inaccessible without a valid password to unlock it. Section 6.5 describes how the password is also encrypted either using PBKDF2 password-based encryption with SHA-384, or using the Linux pam.d module as a SHA-512 hash depending on whether the administrator is local or remote.

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.

ST Section 6.2 states that there are no circumstances where the claimed behavior does not apply.

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

The ST does not specify the use of “a value that does not contain any CSP” and therefore this activity is not applicable.

2.2.3.2 Guidance Activities

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

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

[CCECG] Section 3.1 contains information about key destruction and indicates that there are no circumstances where the TOE’s implementation does not conform to requirement as described in the ST.

2.2.3.3 Test Activities

None.

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

2.2.4.1 TSS Activities

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

[ST] Section 6.2 states that the TOE implements the following cryptographic algorithms in support of TLS, each of which are implemented by both of the cryptographic libraries that the TOE uses: AES-CBC, AES-GCM (128-bit, 256-bit). TLS server functionality uses 256-bit AES-GCM only.

2.2.4.2 Guidance Activities

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites.

2.2.4.3 Test Activities

AES-CBC Known Answer Tests

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

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

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

KAT-2. To test the encrypt functionality of AES-C BC decryption. 93 KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys.

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows: # Input: PT, IV, Key for i = 1 to 1000: if i == 1: CT[1] = AES-CBC-Encrypt(Key, IV, PT) PT = IV else: CT[i] = AES-CBC-Encrypt(Key, PT) PT = CT[i-1]

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

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

AES-GCM Test

The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths: **128 bit and 256 bit keys a) Two plaintext lengths.** One of the plaintext lengths shall be a nonzero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported**. a) Three AAD lengths.** One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported**. b) Two IV lengths.** If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

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

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

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

AES-CTR Known Answer Tests

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Since the Counter Mode does not

specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AESGCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

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

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

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

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

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

AES-CTR Multi-Block Message Test

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

AES-CTR Monte-Carlo Test

The evaluator shall test the encrypt functionality using 100 plaintext/key pairs. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows: # Input: PT, Key for i = 1 to 1000: CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]

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

There is no need to test the decryption engine.

Section 6.2 of [ST] ("Cryptographic Support", Table 1: Validated Algorithm Implementations) identifies the CAVP certifications verifying AES encryption and decryption, as follows.

Algorithm	Tested Capabilities	Certificates
AES as specified in ISO 18033-3, CBC as specified in ISO 10116, and GCM as specified in ISO 19772 (128-bit, 256-bit)	Direction: Decrypt, Encrypt Key Length: 128, 256	A1292 (OpenSSL) A2841 (BC-FJA)

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

2.2.5.1 TSS Activities

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

[ST] Section 6.2 states that the TOE implements the SHA-256, SHA-384 cryptographic algorithms in support of TLS, each of which are implemented by both of the cryptographic libraries that the TOE uses. The TLS server functionality uses SHA-384 only. These hash functions are also associated with the TLS HMAC functions by implication: HMAC-SHA-256 and HMAC-SHA-384; and SHA-512 is associated with Hash_DRBG. The association between RSA as the digital signature function and the hash functions are via the list of cipher suites where each cipher suite identifies RSA and either SHA-256 or SHA-384.

Outside of trusted communications, the TOE also uses the following cryptographic algorithms:

- SHA-384: hashing of remote admin password data
- SHA-512: hashing of local admin password data
- HMAC-SHA-1 with SHA-1: OpenSSL software integrity verification
- HMAC-SHA-256 with SHA-256: BC-FJA software integrity verification

2.2.5.2 Guidance Activities

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites. Section 4.8 Configuring Edge Services contains additional instructions for restricting the hash size. No configuration is required for hashes used for other purposes, such as password hashing and software integrity verification.

2.2.5.3 Test Activities

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

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

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

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

Selected Long Messages Test - Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

Section 6.2 of [ST] (“Cryptographic Support”, Table 2: Validated Algorithm Implementations) identifies the CAVP certifications verifying cryptographic hashing, as follows.

Algorithm	Tested Capabilities	Certificates
SHS as defined in ISO/IEC 10118-3:2004	SHA-1 (digest size 160 bits)	A1292 (OpenSSL)
SHS as defined in ISO/IEC 10118-3:2004	SHA-256 (digest size 256 bits) SHA-384 (digest size 384 bits) SHA-512 (digest size 512 bits)	A1292 (OpenSSL) A2841 (BC-FJA)

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

2.2.6.1 TSS Activities

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

[ST] Section 6.2 identifies the HMAC values as follows:

- For TLS (TLS server functionality uses HMAC-SHA-384 only):
  - HMAC-SHA-256 (512 bit key and block size, 256 bit output length)
  - HMAC-SHA-384 (512 bit key and block size, 384 bit output length)
- For software integrity verification (self-testing)
  - HMAC-SHA-1: OpenSSL software integrity verification (512 bit key and block size, 160 bit output length)
  - HMAC-SHA-256: BC-FJA software integrity verification (512 bit key and block size, 256 bit output length)

2.2.6.2 Guidance Activities

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites. No configuration is required for the KeyedHash HMAC function used for other purposes, i.e., for software integrity verification.

2.2.6.3 Test Activities

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

Section 6.2 of [ST] (“Cryptographic Support”, Table 3: Validated Algorithm Implementations) identifies the CAVP certifications verifying keyed hashing, as follows.

Algorithm	Tested Capabilities	Certificates
HMAC that meets ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”	HMAC-SHA1 MAC: 160 Key Length: 512, digest size 160 bits	A1292 (OpenSSL)
HMAC that meets ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”	HMAC-SHA2-256 MAC: 256	A1292 (OpenSSL) A2841 (BC-FJA)

	Key Length: 512, digest size 256 bits HMAC-SHA2-384 MAC: 384 Key Length: 512, digest size 384 bits	

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

2.2.7.1 TSS Activities

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

[ST] Section 6.2 states that RSA signature generation and verification (3072 bit) is used for TLS.

2.2.7.2 Guidance Activities

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

[CCECG] Section 4.3.6 Step 6: Set the TLS Signature Schemes provides instructions for configuring the configuring the TOE to use the selected cryptographic algorithm and key size for signature services.

2.2.7.3 Test Activities

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

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

RSA Signature Algorithm Tests

Signature Generation Test

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

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

Signature Verification Test

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

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

Section 6.2 of [ST] ("Cryptographic Support", Table 4: Validated Algorithm Implementations) identifies the CAVP certifications verifying digital signature services, as follows.

Algorithm	Tested Capabilities	Certificates
RSA schemes using cryptographic key sizes of 3072 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5	**RSA Signature Generation (FIPS186-4)** Signature Type: PKCS 1.5 Modulo: 3072 Hash Algorithm: SHA2-256 Hash Algorithm: SHA2-384 **RSA Signature Verification (FIPS186-4)** Signature Type: PKCS 1.5 Modulo: 3072 Hash Algorithm: SHA2-256 Hash Algorithm: SHA2-384	A1292 (OpenSSL) A2841 (BC-FJA)

2.2.8 HTTPS Protocol (FCS_HTTPS_EXT.1)

2.2.8.1 TSS Activities

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

[ST] Section 6.2 states that the TOE implements HTTPS for inbound connectivity from remote administrators and Horizon Clients, and for outbound connectivity to Horizon Connection Servers. All HTTPS implementation is compliant with RFC 2818 and uses the TLS functionality as described elsewhere in this AAR (i.e. TLS 1.2). In all cases, an invalid certificate will cause the connection to be aborted.

2.2.8.2 Guidance Activities

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites.

2.2.8.3 Test Activities

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

Tests are performed in conjunction with the TLS evaluation activities.

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

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

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

2.2.9.1 TSS Activities

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

[ST] Section 6.2 and Table 6 indicate that the TOE implements the ISO/IEC 18031:2011 Deterministic Random Bit Generator (DRBG) based on the AES 256 block cipher in counter mode (CTR_DRBG (AES-256)) for OpenSSL and a SHA-512 Hash_DRBG for Bouncy Castle BC-FJA. The TOE instantiates the DRBG with maximum security strength, obtaining a minimum 256 bits of entropy drawn from /dev/random to seed the DRBG. The TOE obtains platform entropy data through the processor RDRAND instruction set, described in the proprietary Entropy Design document. The VMware ESXi hypervisor in the Operational Environment provides a passthrough interface to underlying platform hardware. The TOE uses its DRBGs to generate all keys.

2.2.9.2 Guidance Activities

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

[CCECG] Sections 3.3 and 3.4.2.1 contain instructions for configuring the RNG function.

2.2.9.3 Test Activities

Performed in accordance with NIAP Policy Letter #5.

Section 6.2 of [ST] (“Cryptographic Support”, Table 5: Validated Algorithm Implementations) identifies the CAVP certifications verifying random bit generation, as follows.

Algorithm	Tested Capabilities	Certificates
CTR-DRBG in accordance with ISO/IEC 18031:2011	Counter DRBG Mode AES-256	A1292 (OpenSSL)

Hash_DRBG in accordance with ISO/IEC 18031:2011	Hash DRBG Mode: SHA2-512	A2841 (BC-FJA)

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

2.2.10.1 TSS Activities

FCS_TLSC_EXT.1.1
The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the ciphersuites supported are specified. The evaluator shall check the TSS to ensure that the ciphersuites specified include those listed for this component.

[ST] Section 6.2 identifies the supported cipher suites when acting as TLS client:

- For all interfaces (OpenSSL and BC-FJA):
  - 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
- For BC-FJA interfaces only:
  - TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246
  - TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288
  - TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288

FCS_TLSC_EXT.1.2
The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported.

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

If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE's conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC 5952 for IPv6, RFC 3986 for IPv4) is enforced.

[ST] Section 6.2 states that the TOE validates any presented server certificate in the manner specified by RFC 6125 section 6. Specifically, the client uses the FQDN (host name) in the CN or SAN as the reference identifier for the TLS server certificate. IP addresses and wildcards are not supported. In the evaluated configuration, the Security Administrator can configure whether untrusted certificates are accepted.

FCS_TLSC_EXT.1.4
The evaluator shall verify that TSS describes the Supported Elliptic Curves/Supported Groups Extension and whether the required behaviour is performed by default or may be configured.

[ST] Section 6.2 states that in the evaluated configuration, the TOE is configured to support the following parameters for TLS. The TLS server uses secp384r1 as the curve used for key establishment. The TLS client can use any of secp256r1, secp384r1, or secp521r1 when a TLS_ECDHE cipher suite is negotiated. For interfaces that use BC-FJA, the client also supports the RFC7919 'safe-prime' key generation methods ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, and ffdhe8192 when a TLS_DHE cipher suite is negotiated.

2.2.10.2 Guidance Activities

FCS_TLSC_EXT.1.1
The evaluator shall check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server according to the description in the TSS and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites.

FCS_TLSC_EXT.1.2
The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE includes support for IP addresses, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

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

[CCECG] Section 3.5 describes the identifier for the UAG as a DNS name in the SAN attribute; IP addresses are not supported. Section 4.5 describes the supported identifiers for client certificates as automatically validated using the value in the CN field; SAN is not used as the basis for reference identifier checking and no configuration is needed for this behavior.

Ip addresses are not supported and the TOE is not distributed, so these activities are not applicable.

FCS_TLSC_EXT.1.4
If the TSS indicates that the Supported Elliptic Curves/Supported Groups Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves/Supported Groups Extension.

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server, including the supported Elliptic Curves and Groups Extensions.

2.2.10.3 Test Activities

Removed by TD0670: ~~For all tests in this chapter the TLS server used for testing of the TOE shall be configured not to require mutual authentication~~.

FCS_TLSC_EXT.1.1

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

The evaluator established a TLS connection with the TOE using all supported ciphersuites.

FCS_TLSC_EXT.1.1

Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field, and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

The evaluator configured the server to present a certificate that did not have the Server Authentication purpose in the extendedKeyUsage field and verified the connection failed.

FCS_TLSC_EXT.1.1

Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

The evaluator attempted a connection using a EC certificate while using the expected ECDSA RSA ciphersuite and verified the connection failed.

FCS_TLSC_EXT.1.1

Test 4: The evaluator shall perform the following 'negative tests':

a) The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.

b) Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

c) [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension the evaluator shall configure the server to perform an ECDHE or DHE key exchange in the TLS connection using a non-supported curve/group (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

The evaluator attempted to connect using unsupported and NULL ciphersuites and verified the connection failed. The evaluator also presented a unsupported curve and verified the connection failed.

FCS_TLSC_EXT.1.1

Test 5: The evaluator performs the following modifications to the traffic:

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version and verify that the client rejects the connection.

b) [conditional]: If using DHE or ECDH, modify the signature block in the Server's Key Exchange handshake message, and verify that the handshake does not finished successfully, and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

The evaluator used a TLS tool to modify the TLS version on the server and attempted a connection, verifying the connection failed. The evaluator used the tool to modify the signature block in the Server's Key Exchange handshake message and verified the connection failed.

FCS_TLSC_EXT.1.1

Test 6: The evaluator performs the following 'scrambled message tests':

a) Modify a byte in the Server Finished handshake message and verify that the handshake does not finish successfully and no application data flows.

b) Send a garbled message from the server after the server has issued the ChangeCipherSpec message and verify that the handshake does not finish successfully and no application data flows.

c) Modify at least one byte in the server's nonce in the Server Hello handshake message and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

The evaluator used a TLS tool to modify a byte in the Server Finished handshake message and verified the connection failed. The evaluator also used the tool to send a garbled message from the server after the ChangeCipherSpec message and verified the connection failed. Lastly, the evaluator used the tool to modify a byte in the server's nonce in the Server Hello handshake message and verified the connection failed.

FCS_TLSC_EXT.1.2

Note that the following tests are marked conditional and are applicable under the following conditions:

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

or

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

or

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

Note that for some tests additional conditions apply.

FCS_TLSC_EXT.1.2

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

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.
- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

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

FCS_TLSC_EXT.1.2

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

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

The evaluator configured the server to present a certificate whose CN does not match the reference identifier and does not contain a SAN extension. The connection failed.

FCS_TLSC_EXT.1.2

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

The evaluator configured the server to present a certificate whose CN matched the reference identifier but its SAN does not, and verified the connection failed.

FCS_TLSC_EXT.1.2

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

The evaluator configured the server to present a certificate whose CN matched the reference identifier and has no SAN, and verified the connection still succeeded.

FCS_TLSC_EXT.1.2

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

The evaluator configured the server to present a certificate whose CN did not match the reference identifier but had a SAN that did, and verified the connection succeeded.

FCS_TLSC_EXT.1.2

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

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

N/A – The TOE does not support wildcards.

Modified per TD0634.

FCS_TLSC_EXT.1.2

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

Test 6: [conditional] If IP address identifiers supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.0.1...

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

N/A – The TOE does not support IP addresses.

FCS_TLSC_EXT.1.2

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

of one attribute type at a time while other attribute types contain values that will match a portion of the reference identifier):

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

N/A – RFC 5280 was not selected.

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

FCS_TLSC_EXT.1.3

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

The evaluator loaded all necessary certificates to the TOE to complete the validation chain and successfully established a trusted channel with an external entity.

FCS_TLSC_EXT.1.3

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

The evaluator configured the TOE to cover the presented scenarios and confirmed the connection failed for each.

FCS_TLSC_EXT.1.3

Test 3 [conditional]: The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry

in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

N/A – The TOE does not support administrative overrides for certificate validation failures.

FCS_TLSC_EXT.1.4

Test 1 [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension, the evaluator shall configure the server to perform ECDHE or DHE (as applicable) key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

The evaluator established successful connections with supported curves.

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

2.2.11.1 TSS Evaluation Activity

FCS_TLSS_EXT.1.1

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

[ST] Section 6.2 identifies one supported cipher suite for the TOE's TLS server: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA38 as defined in RFC 52894.

FCS_TLSS_EXT.1.2

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

[ST] Section 6.2 states that the TOE implements TLS 1.2 as both a client and a server, rejecting all other TLS/SSL versions. Specifically, the use of the "FIPS" build as the evaluated configuration of the TOE forces the TOE to use TLS 1.2; configuration options are present for TLS 1.0, 1.1, and 1.3, but they are permanently greyed out in this build. Pre-TLS SSL versions are automatically disabled by default and cannot be enabled under any circumstance.

Modified per TD0635.

FCS_TLSS_EXT.1.3

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.

[ST] Section 6.2 states that for the TLS server, the TOE supports secp384r1 as a supported curve used for key establishment.

FCS_TLSS_EXT.1.4

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

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.

[ST] Section 6.2 states that the TOE does not support session resumption based on either session IDs or session tickets for any TLS interfaces.

2.2.11.2 Guidance Activities

FCS_TLSS_EXT.1.1

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites.

FCS_TLSS_EXT.1.2 and FCS_TLSS_EXT.1.3

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

[CCECG] Section 4.3 provides instructions for configuring the TLS parameters for both client and server and states that the supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS cipher suites. The evaluator verified that all necessary configuring to meet the requirement is present.

2.2.11.3 Test Activities

FCS_TLSS_EXT.1.1

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

The evaluator established connections to the TOE using the supported ciphersuite.

FCS_TLSS_EXT.1.1

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

Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

The evaluator attempted connections using unsupported and NULL ciphersuites, and confirmed the connection failed.

FCS_TLSS_EXT.1.1

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'.

The evaluator used a TLS tool to modify a byte in the Client Finished handshake message and verified the connection failed. The evaluator also examined a successful connection and confirmed the Finished message was encrypted.

FCS_TLSS_EXT.1.2

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

The evaluator attempted a connection using unsupported TLS protocol versions SSL2.0, SSL3.0, TLS1.0, and TLS1.1 and verified all connections failed.

FCS_TLSS_EXT.1.3

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

a) The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single supported elliptic curve specified in the Elliptic Curves Extension. The Evaluator shall verify (though a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange message and successfully establishes the connection.
b) The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single unsupported elliptic curve (e.g. secp192r1 (0x13)) specified in RFC4492, chap. 5.1.1. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

The evaluator made a successful connection using supported elliptic curves. Configuring the server to use an unsupported curve resulted in a failed connection.

FCS_TLSS_EXT.1.3

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).

N/A – The TOE does not support DHE ciphersuite for TLSS connections.

FCS_TLSS_EXT.1.3

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.

N/A – The TOE does not support RSA key establishment.

FCS_TLSS_EXT.1.4

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

FCS_TLSS_EXT.1.4

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.

Added per TD0569.

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

The evaluator used a TLS tool to help capture a session ID from a previous session and use that for a new connection. The evaluator confirmed the connection failed.

FCS_TLSS_EXT.1.4

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.

Added per TD0569.

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

context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

N/A – The TOE does not support session resumption.

Modified by TD0556 and TD0555.

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

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

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

Added per TD0569.

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

N/A – The TOE does not support session resumption.

2.2.12 TLS Server Support for Mutual Authentication (FCS_TLSS_EXT.2)

2.2.12.1 TSS Activities

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

[ST] Section 6.3 states that all certificates that are presented to the TOE are validated, including TLS client certificates when inbound Horizon Client connections are established (i.e., as identified in section 6.7, mutual authentication for Horizon Client user authentication (XML API channel) and mutual authentication for Horizon Client media transmission (Blast channel)).

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

[ST] Section 6.2 states that when acting as a TLS server, the TOE supports mutual authentication for inbound connectivity from remote Horizon Clients. The reference identifier is based on the CN of the certificate and the entire public key portion of the certificate. Specifically, a hash is computed from the public key portion of the certificate and associated with the UPN or email address that is contained in the CN field of the client certificate. Subsequent connection attempts with that CN must use the same certificate. If the presented client certificate is invalid for any reason, the connection is rejected. No administrator override exists to re-adjudicate the rejection of an invalid certificate.

FCS_TLSS_EXT.2.3

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.

[ST] Section 6.2 describes the reference identifier is based on the CN of the certificate and the entire public key portion of the certificate. Specifically, a hash is computed from the public key portion of the certificate and associated with the UPN or email address that is contained in the CN field of the client certificate. Subsequent connection attempts with that CN must use the same certificate. That is, the identifier is the CN which is parsed as the 'baseline' value the first time a certificate with that CN is presented to it. This is stored along with a hash of the public key. The next time the TOE sees a certificate with that same CN, it will compute the hash using the same fields and make sure it matches. In this way each certificate is associated with its "expected" CN.

2.2.12.2 Guidance Activities

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

[CCECG] Section 4.5 includes instructions for configuring the client-side certificates for TLS mutual authentication.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

[CCECG] Section 4.7 describes how to configure TLS client certificate authentication by enabling ClientCertEkuCheck.

FCS_TLSS_EXT.2.3
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.

[CCECG] Section 4.5 includes instructions for configuring the client-side certificates. This section states that the reference identifier of the certificate is automatically validated using the value in the CN field; SAN is not used as the basis for reference identifier checking and no configuration is needed for this behavior.

2.2.12.3 Test Activities

For all tests in this chapter the TLS client used for testing of the TOE shall support mutual authentication.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2
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.

The evaluator started a connection and offered a zero-length client certificate. The connection ultimately failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2
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.

N/A – There is no fallback authentication.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2
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.

The evaluator used a TLSS tool to send a client certificate using an unsupported signature algorithm and confirmed the connection failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

The evaluator created a client certificate signed by an imposter CA and used it to connect to the TOE. The connection failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

The evaluator demonstrated a good, mutually authenticated TLS connection to the TOE. Next, the evaluator attempted to connect to the TOE using a client certificate that lacked Client Authentication purpose in the extendedKeyUsage field. The connection failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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 demonstrated a good, mutually authenticated TLS connection to the TOE. Next, the evaluator used a TLS tool to modify a byte in the client certificate's signature block during Certificate Verify and verified the connection broke.

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

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

The evaluator uploaded all necessary CA certificates to complete the validation chain and verified the connection succeeded.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

The evaluator removed the intermediate CA from the validation chain, thus breaking it. Connections that require this validation chain failed to complete.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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.

N/A – The TOE does not support administrative overrides for selected certificate validation failures.

FCS_TLSS_EXT.2.3

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 presented a client certificate whose identifier did not match the expected identifier and verified the connection breaks.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication Failure Management (FIA_AFL.1)

2.3.1.1 TSS Activities

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

[ST] Section 6.3 provides a description of the remote web GUI interface. It is the only interface provided for remote administration. The remote web GUI locks out administrators after several failed authentication attempts; the threshold for this is a configurable value between 1 and 100, with a default value of 3. The TSF increments the failure counter in non-volatile memory so that the accumulated number of failures is persisted across reboots. In the event the remote administrator is locked out, access is restored by a local admin issuing the 'supervisorctl restart admin' command or by waiting for the lockout period to expire. The lockout period is a configurable value between 1 and 9999 minutes, with a default value of 5.

Section 6.3 describes the remote web GUI and local console as using two different sets of credentials ("admin" for remote administration, "root" for local console), but both are local password-based authentication mechanisms.

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).

[ST] Section 6.3 describes a lockout feature applicable to the remote GUI interface only and states that in the event the remote administrator is locked out, access is restored by a local admin issuing the 'supervisorctl restart admin' command or by waiting for the lockout period to expire. As such, administrator access is always available.

2.3.1.2 Guidance Activities

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

[CCECG] Section 3.4.2 provides instructions for configuring the authentication attempts and time period for the remote administrator. Section 4.2 describes how to unlock a remote Administrator account that is locked.

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.

[CCECG] Section 3.4.2.2.1 describes how the root account failed lockout parameter must be set to zero to ensure that the local admin always has access.

2.3.1.3 Test Activities

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

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

The evaluator configured the number of unsuccessful connections allowed before locking the account, as well as the length of lockout. The evaluator then triggered the lockout by failing authentication attempts until the username was locked out.

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

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

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

The evaluator followed guidance to administratively unlock the locked user account. The evaluator also waited the configured time and confirmed the username was unlocked.

2.3.2 Password Management (FIA_PMG_EXT.1)

2.3.2.1 TSS Evaluation Activity

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.

[ST] Section 6.3 identifies the allowed special characters: !, @, #, $, %, ^, &, *, (, and). The minimum password length is configurable between 8 and 64 characters, and the maximum password length is 64 characters.

2.3.2.2 Guidance Activities

The evaluator shall examine the guidance documentation to determine that it:

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

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

[CCECG] Section 3.4.2.2 describes the password management policy settings and includes the characters that can be used as well as minimum password lengths supported. It also provides guidance on composition of strong passwords.

2.3.2.3 Test Activities

The evaluator shall perform the following tests.

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

The evaluator generated a number of good users and passwords and confirmed they were accepted by the TOE.

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 create a number of bad users with poor passwords that did not pass complexity requirements, and verified those were rejected.

2.3.3 Protected Authentication Feedback (FIA_UAU.7)

2.3.3.1 TSS Evaluation Activity

None.

2.3.3.2 Guidance Activities

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

[CCECG] Section 3.6 indicates that the password characters are automatically obscured (a bullet ('·') will be displayed in place of each character).

2.3.3.3 Test Activities

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

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

The evaluator logged in locally and verified authentication information was obscured.

2.3.4 Password-based Authentication Mechanism (FIA_UAU_EXT.2)

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

2.3.5 User Identification and Authentication (FIA_UIA_EXT.1)

2.3.5.1 TSS Evaluation Activity

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".

[ST] Section 6.3 states that the remote web GUI and local console use two different sets of credentials ("admin" for remote administration, "root" for local console), but both are local password-based authentication mechanisms. In both cases, a successful logon is measured by computing the hash of the supplied password and comparing it to the expected result. Remote user authentication occurs at the application layer and is unrelated to the establishment of the underlying trusted path.

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.

[ST] Section 6.3 states that for both local and remote management interfaces, there is no TSF-mediated function that will be allowed prior to authentication aside from display of the warning banner to the administrator.

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

The TOE is not distributed and therefore this is not applicable.

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

The TOE is not distributed and therefore this is not applicable.

2.3.5.2 Guidance Activities

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

[CCECG] Section 3.6 provides instructions for logging on using the Web UI. Section 4.1 provides instructions for logging on to the console.

2.3.5.3 Test Activities

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

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

The evaluator attempted to log in using good credentials and successfully connected. The evaluator attempted to log in using bad credentials and was denied access.

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

The evaluator examined the TOE for any services available to a user prior to remotely authenticating and found none.

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.

The evaluator examined the TOE for any services available to a user prior to locally authenticating and found the services were consistent.

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.

N/A – The TOE is not distributed.

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

2.3.6.1 TSS Activities

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

[ST] Section 6.3 states that all certificates that are presented to the TOE are validated. This includes TLS server certificates for all cases where the TOE acts as a TLS client, TLS client certificates when inbound Horizon Client connections are established, and the TOE's own TLS server certificate when a CA response to a certificate signing request is provided back to the TOE.

Section 6.3 also notes that the TOE does not use X.509 certificates for trusted updates and executable code integrity verification so this portion of the requirement is trivially satisfied by the TSF.

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.

[ST] Section 6.3 states that the TOE supports the use of CRL for revocation status checking of TLS certificates. In all cases, if the revocation status of a certificate cannot be determined, the TSF treats it is invalid. Both the leaf certificate and any intermediate CA certificates are checked for TLS.

2.3.6.2 Guidance Activities

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

[CCECG] Section 4.7 describes how to configure the TOE to validate certificates for expiration, identity, revocation, and extended key usage values and includes steps for both server and client certificates including syslog client certificates. Section 4.5 describes how the TOE validates client certificates against the CAs placed in the trust store and uses a CRL. The reference identifier of the certificate is automatically validated using the value in the CN field; SAN is not used as the basis for reference identifier checking and no configuration is needed for this behavior.

Section 2.3 states that the TOE does not use X.509 certificates for trusted updates and executable code integrity verification and these extendedKeyUsage fields are not supported by the TOE.

2.3.6.3 Test Activities

The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

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

The evaluator uploaded all necessary CA files to complete the validation chain and successfully connected.

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

The evaluator removed the intermediate CA, thus breaking the validation chain. Connections using that validation chain failed.

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

The evaluator attempted to connect using an expired certificate and verified the connection failed.

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

The evaluator revoked the server cert and attempted to connect. The connection failed due to the cert being revoked. The evaluator repeated the test using a revoked intermediate certificate, and this failed as well.

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

The evaluator uploaded a CRL signed by a CA that did not have CRLsign key usage and attempted to connect using this CRL. The connection failed.

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

The evaluator used a TLS tool to modify a byte in the first eight bytes of the certificate and started a connection. The connection failed.

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

The evaluator used a TLS tool to modify a byte in the signatureValue field of the certificate and started a connection. The connection failed.

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

The evaluator used a TLS tool to modify a byte in the certificate's public key and started a connection. The connection failed.

Test added in accordance with TD0527.

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 evaluation shall conduct the following tests.

N/A – The TOE does not support EC certificates.

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.

N/A – The TOE does not support EC certificates.

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.

N/A – The TOE does not support EC certificates.

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.

N/A – The TOE does not support EC certificates.

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.

N/A – The TOE does not support EC certificates.

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of

the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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

The evaluator installed a CA that lacked any basicConstraints extension and attempted a connection. The connection failed.

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 installed a CA whose basicConstraints extension was set to FALSE and attempted a connection. The connection failed.

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).

X.509 certificates are not used for trusted updates and executable code integrity verification, therefore there are no other distinct uses of certificates.

2.3.7 X.509 Certificate Authentication (FIA_X509_EXT.2)

2.3.7.1 TSS Activities

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

[ST] Section 6.3 states that all certificates that are presented to the TOE are validated and this includes TLS server certificates for all cases where the TOE acts as a TLS client, TLS client certificates when inbound Horizon Client connections are established, and the TOE's own TLS server certificate when a CA response to a certificate signing request is provided back to the TOE. The TOE chooses the certificate to present to external TLS clients based on the server certificate that is loaded into it as part of administrative configuration. In all other cases the TOE validates the certificate that is presented to it and validates the chain based on what is included in the certificate.

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.

[ST] Section 6.3 states that if the revocation status of a certificate cannot be determined, the TSF treats it is invalid. The TOE does not support administrator actions and this is consistent with the selection in the requirement.

2.3.7.2 Guidance Activities

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

[CCECG] contains required configuration for certificates. Specifically, Section 4.5 describes how to configure client certificates, including obtaining and uploading the trusted Root CA and intermediate CA certificates.

UAG authenticates Horizon Client users via mutual TLS authentication and passes access request to Connection Server via SAML assertion. Section 4.6 describes how to configure the SAML Settings required for this, including uploading the Private Key and Certificate Chain.

Section 4.8 covers configuration of Edge Services certificates including adding any root and intermediate CA certificates needed to trust the certificates presented by Horizon Connection Server and Horizon Agents; and optional steps to modify the Host Entries field for name resolution.

Section 4.9 describes 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.

2.3.7.3 Test Activities

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

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

The evaluator shut down the CRL distribution server and started a connection with the TOE. The connection failed because the TOE was unable to communicate with the distribution server, and thus unable to verify any connections.

2.3.8 X.509 Certificate Requests (FIA_X509_EXT.3)

2.3.8.1 TSS Activities

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

This activity is not applicable since the ST author does not select "device-specific information".

2.3.8.2 Guidance Activities

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

[CCECG] Section 3.5 contains instructions on requesting certificates from a CA, including generation of a Certificate Request and establishing the "Common Name", "Organization", "Organizational Unit", or "Country" fields. The instructions state that the identifier for the UAG must be a DNS name in the SAN attribute; IP addresses are not supported.

2.3.8.3 Test Activities

The evaluator shall perform the following tests:

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

The evaluator generated a CSR following user guidance, captured the generated request and ensured that it conforms to the format specified and provides the public key and other required information.

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

The evaluator attempted to upload a signed cert from the previous CSR, but without a full valid certification path. The attempted failed. The evaluator uploaded the necessary CAs to complete the certification path and retried uploaded, which succeeded.

2.4 Security Management (FMT)

2.4.1 General requirements for distributed TOEs

2.4.1.1 TSS 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.

The TOE is not distributed and therefore this is not applicable.

2.4.1.2 Guidance Activities

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

The TOE is not distributed and therefore this is not applicable.

2.4.1.3 Tests Activities

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

The TOE is not distributed and therefore this is not applicable.

2.4.2 Management of Security Functions Behavior (FMT_MOF.1/Functions)

2.4.2.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1.

The TOE is not distributed and therefore this is not applicable.

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).

[ST] Section 6.4 states that the Security Administrator is able to determine and modify the behavior of the TOE's connectivity to an external syslog server and Table 7 identifies the function as available from the web GUI.

2.4.2.2 Guidance Activities

For distributed TOEs see chapter 2.4.1.2.
For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the 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.

[CCECG] Section 5.2.1 Configuring Syslog Server Settings describes how to configure an external syslog server for remote audit storage and refers to "Configure Syslog Server Settings" in Deploying and Configuring VMware Unified Access Gateway for additional information.

2.4.2.3 Test 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.

Without prior authentication as Security Administrator, evaluator was unable to make configuration changes regarding transmission of audit data to an external IT entity.

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

With prior authentication as Security Administrator, the evaluator was able to make configuration changes.

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 SFR s FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

N/A – This option was not selected.

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.

N/A – This option was not selected.

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.

N/A – This option was not selected.

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.

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

[N/A – This option was not selected.

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

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.

Without authentication as Security Administrator, the evaluator was unable to perform any tasks to determine the behavior of selected functions.

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.

With authentication as Security Administrator, the evaluator was able to perform tasks such as configure syslog to an appointed syslog server.

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

2.4.3.1 TSS Activities

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

The TOE is not distributed.

2.4.3.2 Guidance Activities

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

[CCECG] Section 5.4 provides the steps to perform manual updates .and states that updates are applied during the boot process so UAG's security functionality is unavailable as this occurs.

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

The TOE is not distributed and therefore this is not applicable.

2.4.3.3 Test Activities

The evaluator shall try to perform the update using a legitimate update image without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). The attempt to update the TOE shall fail.

Without administrator privileges, the evaluator was unable to perform any form of updates.

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.

This test case was covered by the tests for FPT_TUD_EXT.1.

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

2.4.4.1 TSS Activities

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

[ST] Section 6.4 identifies the administrative functions and whether the function is accessed from the local and/or the remote administrative interface(s). Both interfaces and therefore all of the administrative functions require administrator log-in prior to access. The guidance documentation does not identify any admin functions as available prior to login.

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.

[ST] Section 6.4 states that the management of the TOE's trust store is performed using the web GUI which is restricted to the remote account user (the remote Security Administrator): "admin" as defined in Section 6.3. Section 6.4 also states that there are no lesser-privileged management roles on the TOE; all administrative access to the TOE is by Security Administrators. Therefore, the trust stored is protected using role-based access control restrictions.

2.4.4.2 Guidance Activities

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

The evaluator reviewed the guidance documentation and determined that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified.

The administrative functions are each described in the following sections of the [CCECG]:

- Configuration of syslog export settings (Section 5.2.1 Configuring Syslog Server Settings)
- Setting length requirement for passwords (Section 3.3 Deploying UAG Using OVF Template Wizard, 3.4.2.2 Administrator Password Management Policies (for a PowerShell configuration))
- Configuration of session inactivity time before session termination (Section 3.3 Deploying UAG Using OVF Template Wizard, 3.4.2.2 Administrator Password Management Policies (for a PowerShell configuration))
- Configuration of authentication failure for FIA_AFL.1 (Section 3.3 Deploying UAG Using OVF Template Wizard, 3.4.2.2 Administrator Password Management Policies (for a PowerShell configuration))
- Generating/import of, changing, or deleting of Manage cryptographic keys
    - import and manage X.509v3 certificates to the TOE's trust store (Section 3.5 Obtain and Upload the TLS Server Certificate, Section 4.5 Configuring X.509 Client Authentication, Section 4.8 Configuring Edge Servers)
    - generate certificate signing requests, which include key pairs (Section 3.5 Obtain and Upload the TLS Server Certificate,)
    - designate X.509v3 certificates as trust anchors (Section 3.5 Obtain and Upload the TLS Server Certificate, Section 4.5 Configuring X.509 Client Authentication, Section 48 Configuring Edge Servers)
- Unlock a remote Administrator account (Section 4.2 Unlocking Remote Administrator Account)

- Initiation of a software update (Section 5.4 Performing UAG Package Updates)
- Configuring the banner displayed prior to authentication (Section 4.4, and Section 3.3)

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 as a trust anchor.

Sections 3.5, 4.5, 4.6, and 4.8 of [CCECG] provide information for the administrator to configure and maintain the trust store in a secure way. The information includes secure loading of CA certificates into the trust store and binding signed certificates to UAG admin and public interfaces. Following the provided procedures implicitly sets CA certificates as a trust anchors.

2.4.4.3 Test Activities

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

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

2.4.5.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1.

The TOE is not distributed and therefore this is not applicable.

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

[ST] Section 6.4 states that the Security Administrator has the ability to manage cryptographic keys by using the remote web GUI to manipulate the certificates that reside in the TOE's trust store and by using the local console to generate certificate signing requests, which include key pairs.

2.4.5.2 Guidance 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.

[CCECG] Section 3.5 describes using the local console to generate certificate signing requests, which include key pairs and then how to import them.

The section titled *Configure Syslog Server Settings* in [Deploy] describes how to upload a CA certificate through the GUI for authenticating secure Syslog servers.

2.4.5.3 Test Activities

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

Without prior authentication as Security Administrator, the evaluator was unable to modify or manage any cryptographic keys.

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

With prior authentication as Security Administrator, the evaluator was able to modify cryptographic keys.

2.4.6 Specification of Management Functions (FMT_SMF.1)

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

2.4.6.1 TSS Activities (containing also requirements on Guidance Documentation and Tests)

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

[ST] Section 6.4 identifies the administrative functions provided by the TOE and whether the function is accessed from the local and/or the remote administrative interface(s). The functions are identified as:

Function	Local	Remote
Ability to administer the TOE locally	X	
Ability to administer the TOE remotely		X
Ability to configure the access banner	X	X
Ability to configure the session inactivity time before session termination		X

Ability to update the TOE and verify the updates using hash comparison prior to installing those updates	X	X
Ability to configure authentication failure parameters for FIA_AFL.1		X
Ability to modify the behavior of the transmission of audit data to an external IT entity		X
Ability to manage the cryptographic keys	X	X
Ability to re-enable an Administrator account	X	
Ability to manage the TOE's trust store and designate X.509v3 certificates as trust anchors		X
Ability to import X.509v3 certificates to the TOE's trust store		X

The evaluator examined the TSS, Guidance Documentation and the TOE as observed during all other testing and confirmed that the management functions specified in FMT_SMF.1 are provided by the TOE.

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.

[ST] Section 6.4 identifies the local management interface: a local-only console, which is accessed via VMware vCenter. Section 6.4 further describes the local interface as a direct console interface to the TOE's OS platform with an OS root user account (Security Administrator). [CCECG] Section 3.3 describes all console access is locally through vSphere. Section 3.4.2 describes the root login account used for local console access.

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

The TOE is not distributed and therefore this is not applicable.

2.4.6.2 Guidance Activities

See section 2.4.6.1 in this AAR.

2.4.6.3 Test Activities

The evaluator tests management functions as part of testing the SFRs identified in section 2.4.6. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

The evaluator was able to exercise all management functions.

2.4.7 Restrictions on Security Roles (FMT_SMR.2)

2.4.7.1 TSS Activities

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

[ST] Section 6.4 states that the local and remote interfaces have different roles with different credentials; the local account is an OS root user and the remote account is specifically for the UAG management application itself. Both accounts function as Security Administrators for the management functions that are available on their respective interfaces. There are no lesser-privileged management roles on the TOE; all administrative access to the TOE is by Security Administrators.

2.4.7.2 Guidance Activities

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

The guidance documentation contains instructions for administering the TOE both locally and remotely. Initial configuration procedures in Section 3 are performed locally using the console and the steps starting in Section 4 are performed using the remote Web GUI. Section 2.3 indicates that a web browser is used for remote administration. Therefore no particular configuration is expected to be needed on the client for remote administration.

2.4.7.3 Test Activities

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

In the course of performing the testing activities for the evaluation, the evaluator used both supported management interfaces: local console and remote Web GUI.

2.5 Protection of the TSF (FPT)

2.5.1 Protection of Administrator Passwords (FPT_APW_EXT.1)

2.5.1.1 TSS Activities

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

[ST] Section 6.5 states that the TSF stores all password data in an obfuscated format. Specifically, remote administrator password data is stored in using PBKDF2 password-based encryption with SHA-384, and local administrator password data is stored using the Linux pam.d module as a SHA-512 hash. No administrative interface exists to read password data in plaintext form.

2.5.1.2 Guidance Activities

None defined.

2.5.1.3 Test Activities

None defined.

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

2.5.2.1 TSS Activities

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

[ST] Section 6.2 indicates that the private keys are stored in plaintext and Section 6.5 states that the TSF stores all persistent non-public key data (i.e. certificate private keys) in the password-protected Java Keystore. No administrative interface exists to read the credential data in plaintext form.

2.5.2.2 Guidance Activities

None defined.

2.5.2.3 Test Activities

None defined.

2.5.3 Reliable Time Stamps (FPT_STM_EXT.1)

2.5.3.1 TSS Activities

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.

[ST] Section 6.5 states that the TSF uses time data for audit log timestamps, timed administrator lockouts, idle session timeouts, and validation of X.509 certificates. As a virtual network device, the TOE obtains its time data from the virtualization system on which it is deployed, which can be assumed to have accurate time data per the A.VS_CORRECT_CONFIGURATION assumption.

Added per TD0632.
If “obtain time from the underlying virtualization system” is selected, the evaluator shall examine the TSS to ensure that it identifies the VS interface the TOE uses to obtain time. If there is a delay between updates to the time on the VS and updating the time on the TOE, the TSS shall identify the maximum possible delay.

[ST] Section 6.5 states that As a virtual network device, the TOE obtains its time data from the virtualization system on which it is deployed using the VMware Tools periodic time synchronization interface. This interface is invoked once per minute to determine if there is a mismatch in system time between the host and the VM and either updates the VM’s clock immediately (if behind the host clock) or slows down the VM’s clock to match the host (if ahead of the host clock). A sync operation happens

once every minute regardless of drift; there is no minimum threshold of drift required to trigger a correction and the maximum possible drift is one minute.

2.5.3.2 Guidance 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.

[CCECG] Section 4.5 Enable Time Sync provides instructions for configuring the TOE to obtain time from the underlying VS. The TOE does not use an NTP server.

Added per TD0632.

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

[CCECG] Section 4.5 Enable Time Sync provides instructions for configuring the TOE to obtain time from the underlying VS and informs the administrator of the maximum possible delay.

2.5.3.3 Test Activities

The evaluator shall perform the following tests:

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

The evaluator was able to manually set the time and observed that the time was set correctly.

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.

N/A – The TOE is unable to support NTP servers.

Added per TD0632.

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

The evaluator verified the time changes set by underlying virtual systems.

If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized

or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

The audit component of the TOE does not consist of several parts with independent time information, and therefore this is not applicable.

2.5.4 TSF Testing (FPT_TST_EXT.1)

2.5.4.1 TSS Activities

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

[ST] Section 6.5 states that to verify its own integrity, the TOE performs a file system integrity test on the boot partition using fsck and cryptographic module self-tests during initial start-up for both OpenSSL and Bouncy Castle BC-FJA. The cryptographic module self-tests: Software integrity tests and Cryptographic known answer tests are listed.

Section 6.5 provides the following argument, “These tests are sufficient to ensure that the non-modifiable portions of the TSF executable code have appropriately not been modified, and that the cryptographic functionality has not been tampered with or degraded, either through a compromise of the cryptographic functionality itself or through a degradation of any hardware components on which this functionality relies. There is no situation in which an administrator may unwittingly be using a modified TOE or may be using the TOE to transmit sensitive data using a cryptographic implementation that is not guaranteed to provide appropriate data-at-rest protections.’

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

The TOE is not distributed and therefore this is not applicable.

2.5.4.2 Guidance Activities

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

[CCECG] Section 2.5 describes the possible errors; actions the administrator should take in response; the description corresponds to 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.

This is not applicable since the TOE is not distributed.

2.5.4.3 Test Activities

It is expected that at least the following tests are performed:

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

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

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

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

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

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

The evaluator verified by various self-tests passed during initial startup.

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

N/A – The TOE is not distributed.

2.5.5 Trusted Update (FPT_TUD_EXT.1)

2.5.5.1 TSS Activities

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

[ST] Section 6.5 states supports software/firmware updates. Only one version of the TOE software/firmware is loaded at any one time. (Therefore no delayed activation is possible). The overall TOE version can be checked through the remote administrative GUI, and the versions of individual packages can be checked through the ‘tdnf list intalled’ command on the local console.

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.

[ST] Section 6.5 indicates that the TOE has a manual update mechanism. Package updates to apply security fixes for incremental updates are obtained by the administrator from packages.vmware.com along with the corresponding updates-fips.json file. These packages are hashed using SHA-256; their hash values are checked manually by the administrator. If found to be valid, the administrator places the packages and the updates-fips.json file on a file server that the TOE is configured to check for available updates. The Security Administrator then uses the web GUI to set the TOE to update on next boot and then reboots the TOE. Successful and failed updates are captured in the package-updates.log file.

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.

The options 'support automatic checking for updates' and 'support automatic updates' are not chosen from the selection in FPT_TUD_EXT.1.2 and therefore this activity is not applicable.

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

The TOE is not distributed and therefore this is not applicable.

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

[ST] A published hash is used to protect the trusted update mechanism. The evaluator verified that Section 6.5 describes the Security Administrator active authorization step and that it is not an automated process. After acquiring the update package from the VMware customer support site, the admin validates the SHA-256 hash of the package against the known value published by VMware. The TSS indicates that to enable the update, the GUI is used and therefore the user must be authenticated since the GUI is restricted to the Security Administrator.

2.5.5.2 Guidance Activities

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

Section 3.6 of the CCECG describes how the GUI displays the TOE 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.

[CCECG] Section 5.4 describes how verification of the authenticity of the update is performed using a SHA-256 checksum. It provides procedures for continuing with the process when the hash has been validated and implies not to proceed when validation has not occurred.

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.

[CCECG] Section 5.4 indicates that the published hash is downloaded with the update file (in the JSON file). The description includes how to obtain this.

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

The TOE is not distributed and therefore this is not applicable.

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

The TOE is not distributed and therefore this is not applicable.

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

The TOE does not use a certificate-based mechanism. Therefore this activity is not applicable.

2.5.5.3 Test Activities

The evaluator shall perform the following tests:

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

After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

The evaluator verified the current version of the product; confirmed ability to install an update of UAG components; and verified the version correctly corresponds to that of the update and to the current version of the product.

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.

N/A – This was not claimed.

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 manually verified the published hash of downloaded update files and verified that they matched, resulting in a successful update.

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).

Only manual update is supported.

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

N/A – The TOE is not distributed.

2.6 TOE Access (FTA)

2.6.1 TSF-initiated Termination (FTA_SSL.3)

2.6.1.1 TSS Activities

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

[ST] Section 6.6 states The local console and admin web GUI both have configurable idle session timeout periods – from 30-3600 seconds for the local console (default 300 seconds) and from 1-1440 minutes for

the Admin UI (default 10 minutes). The idle session timeout value is independently configurable on each interface. When the idle session timeout period has elapsed, the session is terminated.

2.6.1.2 Guidance Activities

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

Instructions for configuring the inactivity time period re included in [CCECG] Section 3.4.2.2 which identifies the session timeout parameters for both administrator login accounts: root and admin and indicate that they must be set to a non-zero number.

2.6.1.3 Test Activities

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

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

The evaluator used the guidance documentation to configure several different values for the inactivity time period and verified that the user was logged out after exceeding that time limit.

2.6.2 User-initiated Termination (FTA_SSL.4)

2.6.2.1 TSS Activities

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

[ST] Section 6.6 states that the TOE includes mechanisms for each interface for the Security Administrator on each to voluntarily terminate their own session (logout button on GUI, 'exit' command on CLI).

2.6.2.2 Guidance Activities

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

[CCECG] Section 4.1 describes how to terminate a local or remote interactive session.

2.6.2.3 Test Activities

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

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

The evaluator was able to log in and log off in a local session.

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

The evaluator was able to log in and log off in a remote session.

2.6.3 TSF-initiated Session Locking (FTA_SSL_EXT.1)

2.6.3.1 TSS Activities

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

[ST] Section 6.6 states that the local console has configurable idle session timeout periods from 30-3600 seconds (default 300 seconds), when the configured time-period has been reached the session is terminated.

2.6.3.2 Guidance Activities

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

[CCECG] Section 3.4.2.2.1 states that the evaluated configuration requires a local administrator session to be terminated after some period of inactivity and provides the default value and parameters available for the command.

2.6.3.3 Test Activities

The evaluator shall perform the following test.

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

The evaluator followed the guidance documentation to configure several different values for the inactivity time period and verified each local login was automatically logged out after exceeding the inactivity time period.

2.6.4 Default TOE Access Banners (FTA_TAB.1)

2.6.4.1 TSS Evaluation Activity

The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning

message might be different for different administrative methods of access and might be configured during initial configuration (e.g. via configuration file).

[ST] Section 6.6 states that both local and remote interfaces display a configurable pre-authentication warning banner that can be used to advise Security Administrators of appropriate usage of the system. Each interface has its own independently configurable banner message. The evaluator determined that all admin access is covered in the description, since the description covers both local and remote interfaces and there is only one remote interface (i.e. the web GUI) and one local interface (the console). Both of which have their own banner mechanisms.

2.6.4.2 Guidance Activities

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

Section 4.4 “Configuring the Warning Banner” of the [CCECG] describes how to configure the configure the warning banner for both the Web UI and for the console.

2.6.4.3 Test Activities

The evaluator shall also perform the following test:

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

The evaluator was able to set notice and warning messages and verified they appeared for local and remote access.

2.7 Trusted Path/Channels (FTP)

2.7.1 Inter-TSF Trusted Channel (FTP_ITC.1)

2.7.1.1 TSS Activities

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

[ST] Section 6.7 contains the descriptions of all of trusted channels with authorized IT entities as follows:

- Remote syslog server
  - TLS (port determined by server)
  - No client authentication
  - TOE is client
  - Implemented by OpenSSL
- Horizon Client user authentication (XML API channel)
  - TLS/HTTPS (TCP 443)

        - Client authentication
        - TOE is server
        - Implemented by Bouncy Castle BC-FJA
    - Horizon Client media transmission (Blast channel)
        - TLS/HTTPS (TCP 443)
        - Client authentication
        - TOE is server
        - Implemented by Bouncy Castle BC-FJA
    - Connection Server session establishment
        - TLS/HTTPS (TCP 443)
        - No client authentication
        - TOE is client
        - Implemented by Bouncy Castle BC-FJA
    - Horizon Agent media transmission (Blast channel)
        - TLS/HTTPS (TCP 22443)
        - No client authentication
        - TOE is client
        - Implemented by OpenSSL

The evaluator confirmed all information is present and 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.

2.7.1.2 Guidance Activities

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

Section 4.3 describes Configuring TLS Parameters for both TLS Client and TLS Server connections. Section 4.5 Configuring X.509 Client Authentication describes obtaining certificates (for mutual authentication-TOE is Server) and configuring the authentication settings for the Horizon Client to connect to the UAG. Section 4.7 describes advanced X.509 Settings necessary for proper handling of X.509 certificates. Section 4.8 describes Configuring Edge Services which includes Connection Server, Blast, SAML, Client, and specific X.509 Certificate settings. Section 5.2 describes how to configure an external syslog server for remote audit storage.

Section 4.9 provides recovery instructions should a connection be unintentionally broken.

2.7.1.3 Test Activities

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

The evaluator shall perform the following tests:

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

The evaluator verified connections with each IT entity was tested and 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.

The evaluator verified connections were initiated from the TOE.

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

The evaluator verified connections not sent in plaintext.

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

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

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

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

The evaluator physically disrupted connections with the TOE and verified restored connections were protected.

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.

N/A – The TOE is not distributed.

2.7.2 Trusted Path (FTP_TRP.1/Admin)

2.7.2.1 TSS Activities

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

[ST] Section 6.7 identifies remote administration communications are protected using TLS/HTTPS. The TOE is a server and client authentication is not supported. The protocols described are those specified in the requirement and both HTTPS and TLS server requirements are included in the ST.

2.7.2.2 Guidance Activities

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

[CCECG] Section 3.6 describes how the remote administrative Web UI sessions are established using a web browser. This is the only method supported.

2.7.2.3 Test Activities

The evaluated shall perform the following tests.

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

The evaluator verified that all communications successfully connected.

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

The evaluator verified that all communications were 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.

N/A – The TOE is not distributed.

3 Security Assurance Requirements

3.1 Class ADV: Development

3.1.1 ADV_FSP.1 Basic Functional Specification

The EAs for this assurance component focus on understanding the interfaces (e.g., application programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 3.

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

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

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

3.1.1.1 ADV_FSP.1 Evaluation Activity

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

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

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

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

- Web UI
- Direct console interface
- TLS logical interface

- Horizon Client user authentication (XML API channel)
- Horizon Client media transmission (Blast channel)
- Horizon Connection Server session establishment
- Horizon Agent media transmission (Blast channel)
- Syslog interface.

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

3.1.1.2 ADV_FSP.1 Evaluation Activity

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

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

3.1.1.3 ADV_FSP.1 Evaluation Activity

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

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

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

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

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

3.2 Class AGD: Guidance Documents

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

3.2.1 AGD_OPE.1 Operational User Guidance

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

In addition, the evaluator performs the EAs specified below.

3.2.1.1 AGD_OPE.1 Evaluation Activity

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

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

[ADMIN] and [Deploy] are published at VMware's Documentation web site (https://docs.vmware.com).

3.2.1.2 AGD_OPE.1 Evaluation Activity

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

Section 2 of [CCECG] identifies the platform claimed for the TOE as identified in [ST]; the specific conditions that are expected to be met by the operational environment and/or administrators; and the components in the operational environment.

3.2.1.3 AGD_OPE.1 Evaluation Activity

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

Section 3.1 of [CCECG] identifies VMware's OpenSSL and Bouncy Castle BC-FJA cryptographic libraries provided by the TOE. These are the only cryptographic libraries included in the TOE and no configuration is necessary to enable them since use of the FIPS version automatically ensures that they are used.

3.2.1.4 AGD_OPE.1 Evaluation Activity

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

[CCECG] Section 1.2 makes it explicitly clear that only the functionality claimed in [ST] was evaluated. Any other functionality that does not relate to the security functional claims are considered to be non-interfering with respect to security. Any product security functionality that is within the scope of the evaluation must be configured in the manner specified in this guidance.

3.2.1.5 AGD_OPE.1 Evaluation Activity

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

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

b) **Updated according to TD0536**

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

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

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

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

Part a) is addressed by section 3.2.1.3 above.

Part b) is addressed in section 2.5.5.2 above.

Part c) is addressed by section 3.2.1.4 above.

3.2.2 AGD_PRE.1 Preparative Procedures

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

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

In addition, the evaluator performs the EAs specified below.

3.2.2.1 AGD_PRE.1 Evaluation Activity

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

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

Section 2.1 of [CCECG] identifies the assumptions that state the specific conditions that are expected to be met by the operational environment and/or administrators. Sections 2.2 and 2.3 describe the scope

of evaluation and operational environment that the system must be in to ensure a secure deployment. These sections are written in an informal style and provide sufficient detail and explanation that they can be understood and used by the target audience.

3.2.2.2 AGD_PRE.1 Evaluation Activity

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

Section 2.3 of [CCECG] identifies a single TOE platform consistent with the security target. The instructions and guidance contained in the [CCECG] are applicable to this TOE platform.

3.2.2.3 AGD_PRE.1 Evaluation Activity

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

The evaluators reviewed [CCECG] and [Deploy] and determined that they describe how set up the TOE's logical interfaces for initial use and to configure the external interfaces specified as the TOE's Operational Environment by [ST]. The instructions and guidance contained in the [CCECG] is applicable to the TOE platform and following them results in a successful installation.

3.2.2.4 AGD_PRE.1 Evaluation Activity

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

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

3.2.2.5 AGD_PRE.1 Evaluation Activity

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

The preparative procedures must

a) include instructions to provide a protected administrative capability; and
b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

[CCECG] Section 3.3 and 3.4 include instructions to provide a protected administrative capability. These sections indicate that the passwords for the console root admin and Web UI admin will be set interactively during the deployment process. These sections describe the deployment process and include instructions for setting the passwords.

3.3 Class ALC: Life-Cycle Support

3.3.1 ALC_CMC.1 Labelling of the TOE

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

These CEM work units have been completed and documented in the ETR.

3.3.2 ALC_CMS.1 TOE CM Coverage

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

These CEM work units have been completed and documented in the ETR.

3.4 Class ASE: Security Targeted Evaluation

General ASE

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

3.4.1 ASE_TSS.1 TOE Summary Specification for Distributed TOEs

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

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

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

3.5 Class ATE: Tests

3.5.1 ATE_IND.1 Independent Testing – Conformance

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.

The evaluator should consult Appendix A [in the SD] when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

Testing of the TOE was performed at the Leidos Accredited Testing and Evaluation Lab located in Columbia, Maryland from May 2, 2022 to May 8, 2023.

The evaluation team established a test configuration including the following TOE software instance:

- VMware UAG 2209.2 for vSphere (FIPS) running on VMware ESXi 7.0 with Intel Xeon 6230R (Cascade Lake) processor.

The test configuration included the following components:

VMware Hypervisor

Purpose: TOE host
IP / MAC: 172.16.23.232 / 78:AC:44:41:B7:68

ESXi Version: 7.0

Hyper-V

Purpose: Hosting server
IP / MAC: 172.16.50.10 / DC:F4:01:E8:60
Version: Windows Server Datacenter 10.0.1836

ATE Phone Host server

Purpose: Virtualization server
IP / MAC: 172.16.23.12 / 8C:AE:4C:E1:70:84
Version: Windows 10 Professional

VMware Connection Server

Purpose: Client device to TOE
IP / MAC: 172.16.23.122 / 00:50:56:88:36:BC
Version: 2209.1

VMware Connection Server 2

Purpose: Secondary client device to TOE
IP / MAC: 172.16.23.224 / 00:50:56:A7:A7:17
Version: 2209.1

Domain Controller

Purpose: Domain Controller for local network
IP / MAC: 172.16.23.224 / 00:50:56:88:26:bc
Version: Windows Server 2019

VMware Windows Client

Purpose: Client device to TOE
IP / MAC: 172.16.23.57 / 8C:AE:4C:E1:70:84
Version: 2209.1

VMware Android Client

Purpose: Client device to TOE
IP / MAC: 172.16.100.206 / BE:C6:70:E3:22:8B
Phone model: Galaxy S10 5G
OS: Android 11

Windows 10 Agent

Purpose: Networked device for TOE
IP / MAC: 172.16.23.131 / 00:50:56:88:69:92
Version: 2209.1
ESXi Version: 7.0.2

Windows Server Agent

Purpose: Networked device for TOE
IP / MAC: 172.16.23.133 / 00:0C:29:52:6B:B4
Version: 2209.1
ESXi Version: 7.0.2

RHEL Agent

Purpose: Networked device for TOE
IP / MAC: 172.16.23.222 / 00:0C:29:A):F4:04
Version: 2209.1
ESXi Version: 7.0.2

ATE-GW (Physical)

Purpose: Main router/gateway
IP/ MAC: 172.16.0.1 / ac:1f:6b:95:0c:1d

OS: PfSense 2.4.4-RELEASE-p2

ATE-DC (Physical)

Purpose: Main Domain Controller (DC) for Test environment/DNS server

IP /MAC: 172.16.0.2 / 00:22:19:58:EB:8D

OS: Windows Server 2016 version 1607

Protocols used: RDP, DNS

ATE-ESXi-1 (Physical)

Purpose: Virtualization server

IP/ MAC: 172.16.1.62 / 10:7b:44:92:77:bf

OS: VMware ESXi, 6.5.0, 5969303

Terminal Server (Physical)

Purpose: Provide tester access to the Test Environment from corporate network.

IP/MAC: 172.16.1.50 / D4:BE:D9:B4:FE:66

OS: Windows server 2016 version 1607

Protocols used: RDP

TLSS.leidos.ate (VM)

Purpose: Hosts TLS Test Tools

IP/MAC: 172.16.0.25 / 00:50:56:b1:66:0b

OS: Ubuntu 18.04.5

Protocols Used: TLS

Relevant Software:

Proprietary Python TLS test tools

OpenSSL 1.1.1

Wireshark 2.6.10

Revocation server

Purpose: CRL distribution

IP/MAC: 172.16.1.70 / 00:50:56:B1:A0:FC

OS: Ubuntu 18.04.5

Relevant Software:

Apache 2.4.29

UAG-Update

Purpose: Update repository

IP/MAC: 172.16.23.140 / 00:50:56:A7:97:9A

OS: Photon 3

Relevant Software:

Apache 2.4.55

UAG-Syslog

Purpose: Syslog machine

IP/MAC: 172.16.23.140 / 00:50:56:A7:FB:B2

OS: Photon 3

Relevant Software:

Syslog-ng 3.37.1

Each relevant Test Activity identified in [SD_ND] was given its own test case in the Test Report. Each test case consists of the test steps specified in the Supporting Documents, along with the actual test steps performed by the evaluators and any corresponding evidence.

3.5.1.1 Tests Evaluation Activity

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section A.9.3 in the SD.

The TOE is not a distributed TOE, so these additional activities are not applicable.

3.6 Class AVA: Vulnerability Assessment

3.6.1 AVA_VAN.1 Vulnerability Survey

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

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. Table 2: Mapping of AVA_VAN.1 CEM Work Units to Evaluation Activities, in the SD, indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

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

3.6.1.1 AVA_VAN.1 Evaluation Activity (Documentation)

In addition to the activities specified by the CEM in accordance with Table 2 in the SD, 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.

Modified by TD0547:

The developer shall provide documentation identifying the list of software and hardware components[1] that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE), for example a web server, protocol or cryptographic libraries (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version

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

number of the components and will be used by the evaluators in formulating vulnerability 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 3.4.1.2 and 3.5.1.2.

The vendor provided information on the software components of the TOE. Specifically, [ST] identifies the name of the software TOE, the OS (VMware Photon 3.0), and the two cryptographic implementations included in the TOE: VMware's OpenSSL FIPS Object Module 2.0.20-vmw and Vmware's BC-FJA (Bouncy Castle FIPS Java API) 1.0.2.3.

The vendor separately provided the version of the OpenSSL library used by the TOE as version 1.0.2zg. This information was used as input to the vulnerability analysis.

There is no hardware included in the TOE, it runs as a virtual network device on an environmental hypervisor and hardware platform, consistent with "Case 1" virtual network devices defined in section 1.2 of the NDcPP. However, since the PP states that at a minimum the processors used by the TOE should be included in the search, the evaluator included a search for the processor the TOE in the tested configuration as identified in the ST: Intel Xeon 6230R (Cascade Lake).

The TOE is not distributed and therefore the last parts of the activity are not applicable.

3.6.1.2 AVA_VAN.1 Evaluation Activity

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

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

- NIST National Vulnerabilities Database (used to access CVE and US-CERT databases identified below): http://web.nvd.nist.gov/view/vuln/search
    - Common Vulnerabilities and Exposures: http://cve.mitre.org/cve/ https://www.cvedetails.com/vulnerability-search.php
    - US-CERT: http://www.kb.cert.org/vuls/html/search
- VMware's Security Advisories page: https://www.vmware.com/security/advisories.html

Searches were performed on 10 May 2023 and updated 22 June 2023 using the following search terms:

- VMware Unified Access Gateway
- VMware UAG

- Unified Access Gateway
- VMware Photon 3.0
- Vmware's OpenSSL FIPS Object Module 2.0.20-vmw
- OpenSSL 1.0.2zg
- BC-FJA
- Intel Xeon 6230R.

No vulnerabilities were identified for the TOE.

The evaluation team determined that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential. Results are detailed in the AVA.