Assurance Activities Report

for

VMware Horizon Connection Server 8 2209 (8.7)

Version 1.2

28 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 assurance activities associated with the evaluation of VMware Horizon Connection Server 8 2209 (8.7). This report contains sections documenting the performance of assurance activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in the Evaluation Activities for the Protection Profile for Application Software, Version 1.4, 2021-10-07 [App PP] and the Functional Package for Transport Layer Security (TLS), Version 1.1, 2019-02-12 [TLS Package].

1.1 Applicable Technical Decisions

The NIAP Technical Decisions that apply to the [App PP] and [TLS Package] are referenced in the Security Target. Rationale is included for those Technical Decisions that do not apply to this evaluation.

1.2 Evidence

[App PP]	Protection Profile for Application Software, Version 1.4, 2021-10-07
[TLS Package]	Functional Package for Transport Layer Security (TLS), Version 1.1, 2019-02-12
[ST]	VMware Horizon Connection Server 8 2209 (8.7) Security Target, Version 1.0, 6 April 2023
[CCECG]	VMware Horizon Connection Server 8 2209 (8.7) Common Criteria (CC) Evaluated Configuration Guide, Version 1.0, April 25, 2023
[Deploy]	VMware Horizon 2209 Horizon Overview and Deployment Planning, 2022
[INSTALL]	VMware Horizon 2209 Horizon Installation and Upgrade, 2022
[ADMIN]	VMware Horizon 2209 Horizon Administration, 2022
[POD]	VMware Horizon 2209 Cloud Pod Architecture in Horizon, 2022
[Security]	VMware Horizon 2209 Horizon Security, 2022
[Linux_Desk]	VMware Horizon 2209 Linux Desktops and Applications in Horizon, 2022
[Win_Desk]	VMware Horizon 2209 Windows Desktops and Applications in Horizon, 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, Version 3.1, Revision 5, dated: April 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Version 3.1, 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

- [AppPP] Protection Profile for Application Software, Version 1.4, 2021-10-07
- [TLS Package] Functional Package for Transport Layer Security (TLS), Version 1.1, 2019-02-12

1.4 CAVP/ACVP Certificates

The TOE uses cryptography to secure data in transit between itself and its operational environment. TSF cryptographic services are implemented by the Bouncy Castle cryptographic library included within the TOE boundary. The TOE uses VMware's BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. The cryptographic algorithms supplied by the TOE are CAVP validated. The following table identifies the cryptographic algorithms used by the TSF, the associated standards to which they conform, and the NIST certificates that demonstrate that the claimed conformance has been met.

Functions	Libraries	Standards	Certificates
FCS_CKM.1/AK Cryptographic Asymmetric Key Generation			
ECC key pair generation (NIST curves P-256, P-384, P-521)	BC-FJA	FIPS PUB 186-4	A2841 (BC-FJA)
FFC key pair generation (2048 bit)	BC-FJA	FIPS PUB 186-4	A2841 (BC-FJA)
Safe-prime FFC key pair generation (2048, 3072, 4096, 6144, 8192 bit)	BC-FJA	NIST SP 800-56A Revision 3; RFC 7919	A2841 (BC-FJA)
FCS_CKM.2 Cryptographic Key Establishment			
Elliptic curve-based key establishment	BC-FJA	NIST SP 800-56A	A2841 (BC-FJA)
FFC safe-primes key establishment	BC-FJA	NIST SP 800-56A Revision 3; RFC 7919	A2841 (BC-FJA)
FCS_COP.1/Hash Cryptographic Operation – Hashing			
SHA-256, SHA-384, and SHA-512 (digest sizes 256, 384, and 512 bits)	BC-FJA	FIPS PUB 180-4	A2841 (BC-FJA)
FCS_COP.1/KeyedHash Cryptographic Operation – Keyed Hash Message Authentication			
HMAC-SHA-256 and SHA-384	BC-FJA	FIPS PUB 198-1 FIPS PUB 180-4	A2841 (BC-FJA)
FCS_COP.1/Sig Cryptographic Operation – Signing			
RSA (2048-bit, 3072-bit)	BC-FJA	FIPS PUB 186-4, Section 5	A2841 (BC-FJA)
FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption			
AES-CBC (128 bits, 256 bits)	BC-FJA	CBC as defined in NIST SP 800-38A	A2841 (BC-FJA)
AES-GCM (128 bits, 256 bits)	BC-FJA	GCM as defined in NIST SP 800-38D	A2841 (BC-FJA)
AES-CTR (128 bits)	BC-FJA	CTR as defined in NIST SP 800-38A	A2841 (BC-FJA)
FCS_RBG_EXT.2 Random Bit Generation from Application			
Hash_DRBG (128 bits, 256 bits)	BC-FJA	NIST SP 800-90A NIST SP 800-57	A2841 (BC-FJA)

1.5 SAR Evaluation

The following Security Assurance Requirements (SARs) were evaluated during the evaluation of the TOE:

SAR	Verdict
ASE_CCL.1	Pass
ASE_ECD.1	Pass
ASE_INT.1	Pass
ASE_OBJ.2	Pass
ASE_REQ.2	Pass
ASE_TSS.1	Pass
ADV_FSP.1	Pass
AGD_OPE.1	Pass
AGD_PRE.1	Pass
ALC_CMC.1	Pass
ALC_CMS.1	Pass
ALC_TSU_EXT.1	Pass

2 Security Functional Requirement Assurance Activities

This section describes the assurance activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The assurance activities are derived from the [App PP] and [TLS Package] and modified by applicable NIAP Technical Decisions. Assurance activities for SFRs not claimed by the TOE have been omitted.

Evaluator notes, such as changes made due to NIAP Technical Decisions, are in bold text. Bold text is also used within assurance activities to identify when they are mapped to individual SFR elements rather than the component level.

2.1 Cryptographic Support (FCS)

2.1.1 FCS_CKM_EXT.1 Cryptographic Key Generation Services[1]

2.1.1.1 TSS Assurance Activity

The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the generate no asymmetric cryptographic keys selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.

The evaluator examined the application and its associated developer documentation and determined the TOE requires asymmetric key generation services, since it uses cryptographic protocols for communication with external IT entities. Section 5.2.1.1 of [ST] ("FCS_CKM_EXT.1 Cryptographic Key Generation Services") specifies the TOE implements asymmetric key generation. Section 6.2 of the [ST] describes the TOE's asymmetric key function. The activities were performed as stated in the selection-based requirements.

2.1.1.2 Guidance Assurance Activity

None.

2.1.1.3 Test Assurance Activity

None.

2.1.2 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation

2.1.2.1 TSS Assurance Activities

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

[ST] Section 6.2 Table 5 and text that follows identifies the supported asymmetric key sizes P-256, P-384, and P-521 ECC keys and ephemeral DH keys using ffdhe groups 2048, 3072, 4096, 6144, and 8192. These keys are generated in support of the DHE and ECDHE key establishment schemes that are used for TLS and TLS/HTTPS communications. The description also indicates that the TOE generates asymmetric DSA keys

[1] Modified by TD0717

that are an asymmetric wrapper to the symmetric 128-bit AES_CTR message keys. This implies FFC key generation as identified in Table 5.

If the application "invokes platform-provided functionality for asymmetric key generation," then the evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.

[ST] Section 5.2.1.2 selects "implement functionality" and does not select "invokes platform-provided functionality for asymmetric key generation". Therefore this is not applicable.

2.1.2.2 Guidance Assurance Activity

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 uses defined in this PP.

[CCECG] Section 3.5 describes how to configure the TLS parameters to use the selected key generation schemes for both Server and Client TLS functions. These settings are configured implicitly by defining the supported TLS cipher suite. Configuration for named groups is performed separately and is also described in Section 3.5. Sections 2.3 and 2.5.1.4.2 provide instructions for enabling FIPS Mode.

2.1.2.3 Test Assurance Activities

If the application "implements asymmetric key generation," then the following test activities shall be carried out.

Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application.

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 P521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

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

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

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

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

Private Key:

- len(q) bit output of RBG where $1 \leq x \leq q-1$
- len(q) + 64 bit output of RBG, followed by a mod q-1 operation where $1 \leq x \leq q-1$.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- $g \neq 0,1$
- q divides p-1
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

for each FFC parameter set and key pair.

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

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

The TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.3 FCS_CKM.1/SK Cryptographic Symmetric Key Generation

2.1.3.1 TSS Assurance Activity

The evaluator shall review the TSS to determine that it describes how the functionality described by FCS_RBG_EXT.1 is invoked.

If the application is relying on random bit generation from the host platform, the evaluator shall verify the TSS includes the name/manufacturer of the external RBG and describes the function call and parameters used when calling the external DRBG function. If different external RBGs are used for different platforms, the evaluator shall verify the TSS identifies each RBG for each platform. Also, the evaluator shall verify the TSS includes a short description of the vendor's assumption for the amount of entropy seeding the external DRBG. The evaluator uses the description of the RBG functionality in FCS_RBG_EXT or documentation available for the operational environment to determine that the key size being requested is identical to the key size and mode to be used for the encryption/decryption of the user data.

[ST] Section 6.2 states that the TOE generates 128-bit AES_CTR keys that are used as message keys. Message keys are single-use keys that are wrapped using the recipient's public key. The TOE implements DSA key generation to generate 2048-bit signing keys for data integrity. The TOE implements DRBG functionality for key generation, using the Hash_DRBG. Specifically, random numbers are obtained from

the Windows-PRNG platform API. The TSS states that it is assumed that the platform provides at least 256 bits of entropy. This is consistent with FCS_RBG_EXT and the proprietary EAR.

2.1.3.2 Guidance Assurance Activity

None.

2.1.3.3 Test Assurance Activity

None.

2.1.4 FCS_CKM.2 Cryptographic Key Establishment

2.1.4.1 TSS Assurance Activity

Modified by TD0717

The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in **FCS_CKM.1.1/AK**. 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 5.2.1.4 selects Elliptic curve-based key establishment schemes and FFC Schemes using "safe-prime" groups and RFC 7919. This corresponds with the ECC and FFC key generation schemes in FCS_CKM.1/AK. [ST] Section 6.2 says these keys are generated in support of the DHE and ECDHE key establishment schemes that are used for TLS and TLS/HTTPS communications. The TSS also states that the TOE has both a TLS client and a TLS server; and indicates that both support cipher suites using DHE and ECDHE (ciphers beginning with: TLS_DHE_* and TLS_ECDHE_*).

2.1.4.2 Guidance Assurance Activity

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 3.5 describes how to configure the TLS parameters to use the selected key establishment schemes for both Server and Client TLS functions. These settings are configured implicitly by defining the supported TLS cipher suite. Configuration for named groups is performed separately and is also described in Section 3.5. Sections 2.3 and 2.5.1.4.2 provide instructions for enabling FIPS Mode.

2.1.4.3 Test Assurance Activities

Evaluation Activity 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.

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function

(KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

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

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

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_DIT_EXT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

The TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.5 FCS_COP.1/Hash Cryptographic Operation – Hashing

2.1.5.1 TSS Assurance Activity

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

[ST] Table 5 identifies the supported hash functions as SHA-256, SHA-384, and SHA-512 (digest sizes 256, 384, and 512 bits). Section 6.2 says that the TOE uses hash functions in support of TLS communications (SHA-256, SHA-384), HTTPS communications (SHA-256, SHA-384), digital signatures (SHA-384, SHA-512), and certificate identity verification (SHA-256). The hash function is also associated with the DRBG cryptographic function(i.e. Hash_DRBG).

2.1.5.2 Guidance Assurance Activity

None.

2.1.5.3 Test Assurance Activities

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF hashes only 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.

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

Test 1: 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.

Test 2: 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.

Test 3: 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. The length of the ith message is 512 + 99*i, where $1 \leq i \leq m$. 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.

Test 4: 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. The length of the ith message is 512 + 8*99*i, where $1 \leq i \leq m/8$. 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.

Test 5: Pseudorandomly Generated Messages Test This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

The TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.6 FCS_COP.1/KeyedHash Cryptographic Operation – Keyed-Hash Message Authentication

The evaluator shall perform the following activities based on the selections in the ST.

2.1.6.1 TSS Assurance Activity

None.

2.1.6.2 Guidance Assurance Activity

None.

2.1.6.3 Test Assurance Activity

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 IV using a known-good implementation.

The TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.7 FCS_COP.1/Sig Cryptographic Operation – Signing

2.1.7.1 TSS Assurance Activity

None.

2.1.7.2 Guidance Assurance Activity

None.

2.1.7.3 Test Assurance Activities

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

RSA Signature Algorithm Tests

Test 1: Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus

value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Test 2: Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.8 FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption

2.1.8.1 TSS Assurance Activity

None.

2.1.8.2 Guidance Assurance Activity

The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required modes and key sizes is present.

[CCECG] Section 3.5 describes how to configure the TLS parameters to use the selected ciphersuites for both Server and Client TLS functions. The encryption/decryption modes and key sizes are configured implicitly by defining the supported TLS cipher suite. Sections 2.3 and 2.5.1.4.2 provide instructions for enabling FIPS Mode. CCECG Section 3.3 describes steps to configure the message bus communications over JMS (uses AES-CTR-128 session keys to encrypt data).

2.1.8.3 Test Assurance Activities

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

AES-CBC Known Answer Tests

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

- KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.
- KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an allzeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality of AES-CBC, the

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

- KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.
- KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

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

AES-CBC Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation. AES-CBC Monte Carlo Tests The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3- tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
    if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
    else:
      CT[i] = AES-CBC-Encrypt(Key, PT)
      PT = CT[i-1]
```

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

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

AES-GCM Monte Carlo Tests

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
- Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
- Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a nonzero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
- 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 Tests

Test 1: Known Answer Tests (KATs)

There are four Known Answer Tests (KATs) described below. 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.

To test the encrypt functionality, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from 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, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input.

To test the encrypt functionality, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. Five of the key values shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.

To test the encrypt functionality, the evaluator shall supply the two sets of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values an an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second 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, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values 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 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.

To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 128-bit key value of all zeros and using a 256 bit key value of all zeros, respectively, and an IV of all zeros. 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]. To test the decrypt functionality, 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.

Test 2: Multi-Block Message Test

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

Test 3: Monte-Carlo Test

For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine.

The evaluator shall test the encrypt functionality using 200 plaintext/key pairs. 100 of these shall use 128 bit keys, and 100 of these shall use 256 bit keys. The plaintext values shall be 128-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode # 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 TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.9 FCS_HTTPS_EXT.1/Client HTTPS Protocol

2.1.9.1 TSS Assurance Activity

FCS_HTTPS_EXT.1.1/Client

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 the TOE's implementation of HTTPS conforms to RFC 2818 and its implementation of TLS conforms to RFC 5246.

The TOE uses TLS 1.2, both standalone and as part of HTTPS, for client and server communications. If a presented certificate is invalid, the TSF will automatically reject it when in its evaluated configuration, regardless if the connection is TLS or HTTPS or if the presented certificate is from a client or server peer. When the TOE is acting as a client, mutual authentication is supported for Connection Server cloud pod communications as identified in Table 6.

FCS_HTTPS_EXT.1.2/Client and FCS_HTTPS_EXT.1.3/Client
None.

2.1.9.2 Guidance Assurance Activity

FCS_HTTPS_EXT.1.1/Client, FCS_HTTPS_EXT.1.2/Client, and FCS_HTTPS_EXT.1.3/Client
None.

2.1.9.3 Test Assurance Activities

FCS_HTTPS_EXT.1.1/Client
The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

Covered by FCS_TLSC_EXT.1.1 Test 1 that shows a successful connection from the TOE to a client secured with TLS.

FCS_HTTPS_EXT.1.2/Client
Other tests are performed in conjunction with the TLS package.

Covered by FCS_TLSC_EXT.1.1 Test 1 that shows a successful connection from the TOE to a client secured with TLS.

FCS_HTTPS_EXT.1.3/Client
Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:
Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the selected action in the SFR. If "notify the user" is selected in the SFR, then the evaluator shall also determine that the user is notified of the certificate validation failure. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that again, using a certificate without a valid certification path results in the selected action in the SFR, and if "notify the user" was selected in the SFR, the user is notified of the validation failure.

Evidence for validation failure because of an incomplete certificate chain can be found in FIA_X509_EXT.1.1 Test 1.

2.1.10 FCS_HTTPS_EXT.1/Server HTTPS Protocol[2]

2.1.10.1 TSS Assurance Activity

FCS_HTTPS_EXT.1.1/Server
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 the TOE's implementation of HTTPS conforms to RFC 2818 and its implementation of TLS conforms to RFC 5246.

The TOE uses TLS 1.2, both standalone and as part of HTTPS, for client and server communications. If a presented certificate is invalid, the TSF will automatically reject it when in its evaluated configuration, regardless if the connection is TLS or HTTPS or if the presented certificate is from a client or server peer. Mutual authentication is supported for connections with remote administrators and for Connection Server cloud pod communications as identified in Table 6.

FCS_HTTPS_EXT.1.2/Server, FCS_HTTPS_EXT.1.3/Server
None.

2.1.10.2 Guidance Assurance Activity

FCS_HTTPS_EXT.1.1/Server, FCS_HTTPS_EXT.1.2/Server, and FCS_HTTPS_EXT.1.3/Server

None.

2.1.10.3 Test Assurance Activities

FCS_HTTPS_EXT.1.1/Server
The evaluator shall attempt to establish an HTTPS connection to the TOE using a client, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

Covered by FCS_TLSS_EXT.1.1 Test 1 that shows a successful connection to the TOE secured with TLS.

FCS_HTTPS_EXT.1.2/Server
Other tests are performed in conjunction with the TLS package.

Covered by FCS_TLSS_EXT.1.1 Test 1 that shows a successful connection to the TOE secured with TLS.

Added by TD0709
FCS_HTTPS_EXT.1.3/Server
Other tests are performed in conjunction with the TLS Functional Package, FCS_HTTPS_EXT.2 (dependent on selections in FTP_DIT_EXT.1), and FIA_X509_EXT.1.

[2] Modified by TD0709

Evidence for validation failure because of unsupported signature algorithm can be found in FCS_TLSS_EXT.2 Test 3.

2.1.11 FCS_HTTPS_EXT.2 HTTPS Protocol with Mutual Authentication

2.1.11.1 TSS Assurance Activity

None.

2.1.11.2 Guidance Assurance Activity

None.

2.1.11.3 Test Assurance Activities

Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the selected action in the SFR. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that again, using a certificate without a valid certification path results in the selected action in the SFR.

Evidence for validation failure because of an incomplete certificate chain can be found in FIA_X509_EXT.1.1 Test 1.

2.1.12 FCS_RBG_EXT.1 Random Bit Generation Services

2.1.12.1 TSS Assurance Activities

If "use no DRBG functionality" is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

In FCS_RBG_EXT.1, the ST author has selected ***implement DRBG functionality***. Therefore this activity is not applicable.

If "implement DRBG functionality" is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.

In FCS_RBG_EXT.1, the ST author has selected ***implement DRBG functionality***. The evaluator confirmed the ST includes FCS_RBG_EXT.2.

If "invoke platform-provided DRBG functionality" is selected, the evaluator performs the following activities. The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers. The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used correctly for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

In FCS_RBG_EXT.1, the ST author has not selected ***invoke platform-provided DRBG functionality***. Therefore this activity is not applicable.

2.1.12.2 Guidance Assurance Activity

None.

2.1.12.3 Test Assurance Activity

If "invoke platform-provided DRBG functionality" is selected, the following tests shall be performed:

The evaluator shall decompile the application binary using a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

The following are the per-platform list of acceptable APIs:

Android: The evaluator shall verify that the application uses at least one of javax.crypto.KeyGenerator class or the java.security.SecureRandom class or/dev/random or /dev/urandom.

Microsoft Windows: The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator API for Windows Universal Applications. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

Apple iOS: The evaluator shall verify that the application invokes either SecRandomCopyBytes, CCRandomGenerateBytes or CCRandomCopyBytes, or uses /dev/random directly to acquire random.

Linux: The evaluator shall verify that the application collects random from /dev/random or /dev/urandom.

Oracle Solaris: The evaluator shall verify that the application collects random from /dev/random.

Apple macOS: The evaluator shall verify that the application invokes either CCRandomGenerateBytes or CCRandomCopyBytes, or collects random from /dev/random.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

The TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.13 FCS_RBG_EXT.2 Random Bit Generation from Application

2.1.13.1 TSS Assurance Activity

FCS_RBG_EXT.2.1

None.

FCS_RBG_EXT.2.2

Documentation shall be produced – and the evaluator shall perform the activities – in accordance with Appendix C – Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

The vendor provided documentation describing the entropy sources used by the TOE. The evaluator performed the activities, in accordance with Appendix C – Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

2.1.13.2 Guidance Assurance Activity

FCS_RBG_EXT.2.1 and FCS_RBG_EXT.2.2
None.

2.1.13.3 Test Assurance Activities

The evaluator shall perform the following tests, depending on the standard to which the RBG conforms.

FCS_RBG_EXT.2.1

Implementations Conforming to FIPS 140-2 Annex C.

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

FCS_RBG_EXT.2.1

Test 1: The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.

FCS_RBG_EXT.2.1

Test 2: The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST-Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section E.3. The evaluators ensure that the 10,000th value produced matches the expected value.

FCS_RBG_EXT.2.1

Implementations Conforming to NIST Special Publication 800-90A

- **Test 1:** The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.

  If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The

next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. “generate one block of random bits” means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

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

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

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

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

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

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

FCS_RBG_EXT.2.2

In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates.

The TOE includes ACVP-certified VMware BC-FJA (Bouncy Castle FIPS Java API) version 1.0.2.3 with JDK 11. See Section 1.4 for the applicable ACVP certificates.

2.1.14 FCS_STO_EXT.1 Storage of Credentials

2.1.14.1 TSS Assurance Activity

The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

[ST] Section 6.2: The TOE relies on platform-provided storage mechanisms for storing X.509 certificates and external system credentials

All certificates are protected by the platform via the Windows Certificate Store.

Any credential where the TOE authenticates to an external system (e.g. vCenter) is stored locally in an LDAP directory. These credentials are separate from credentials used by administrators to access the TOE, which are maintained by the environmental Active Directory. All credential data marked as sensitive in the LDAP directory is protected by the OS platform using DPAPI. The master key for encryption of the directory

is stored in the Windows Registry and is also protected using DPAPI. The following credentials are protected in this manner:

- Per-domain per-administrator accounts for one-way and no-trust domain relationships
- Per-domain service accounts for no-trust domains
- Per-domain service accounts for VM domain join
- Per-resource access credentials
- vCenter CRUD service account
- Event database CRUD service account
- Network resource access credentials for Hybrid Logon user sessions

2.1.14.2 Guidance Assurance Activity

None.

2.1.14.3 Test Assurance Activity

Modified by TD0717

For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/SKC or conditioned according to FCS_CKM.1.1/AK and **FCS_CKM_EXT.1/PBKDF**. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

Android: The evaluator shall verify that the application uses the Android KeyStore or the Android KeyChain to store certificates.

Microsoft Windows: The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). For Windows Universal Applications, the evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

Apple iOS: The evaluator shall verify that all credentials are stored within a Keychain.

Linux: The evaluator shall verify that all keys are stored using Linux keyrings.

Oracle Solaris: The evaluator shall verify that all keys are stored using Solaris Key Management Framework (KMF).

Apple macOS: The evaluator shall verify that all credentials are stored within Keychain.

The evaluator verified all certificates were stored in the Windows Certificate Store and other credentials are stored using DPAPI.

2.1.15 FCS_TLS_EXT.1 TLS Protocol [TLS Package]

2.1.15.1 General Assurance Activity

The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

The selections in FCS_TLS_EXT.1.1 are ***TLS as a client*** and ***TLS as a server***. The ST includes FCS_TLSS_EXT.1 and FCS_TLSC_EXT.1 from the selection-based requirements in [TLS Package], consistent with the selections made in FCS_TLS_EXT.1.1.

2.1.16 FCS_TLSC_EXT.1 TLS Client Protocol [TLS Package]

2.1.16.1 TSS Assurance 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 cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

[ST] Section 6.2 identifies the following supported cipher suites conforming to RFC 5246:

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

The specified ciphers is identical to the ones specified in SFR component.

FCS_TLSC_EXT.1.2
The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported (e.g. Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the product.

[ST] Section 6.2 states that as part of certificate validation in the establishment of TLS connectivity, the TOE will validate the reference identifier of a presented server certificate. This is done through comparison of the DN in the connection URL against the CN and SAN in the certificate. IP addresses are not supported. Wildcards are only supported for the left-most label immediately preceding the public suffix. Certificate pinning is not supported.

FCS_TLSC_EXT.1.3
If the selection for authorizing override of invalid certificates is made, then the evaluator shall ensure that the TSS includes a description of how and when user or administrator authorization is obtained. The evaluator shall also ensure that the TSS describes any mechanism for storing such authorizations, such that future presentation of such otherwise-invalid certificates permits establishment of a trusted channel without user or administrator action.

[ST] The ST author does not make the selection for authorizing override of invalid certificates and therefore this activity is not applicable. Section 6.2 states that if a presented certificate is invalid, the TSF will automatically reject it, regardless if the connection is TLS or HTTPS or if the presented certificate is from a client or server peer.

2.1.16.2 Guidance Assurance Activities

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

[CCECG] Section 2.5.3 instructs to enable FIPS mode that configures the TOE to use FIPS compliant Cryptography. Section 3.5 describes how to configure the TLS parameters to use the selected ciphersuites for both Server and Client TLS functions. This section also states that TLS 1.2 is enabled by default; no separate configuration is needed for this.

FCS_TLSC_EXT.1.2
The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

[CCECG] Section 3.1 provides instructions for setting the reference identifier for a Connection Server in external POD. Section 3.9 provides instructions for setting the reference identifiers for the database. Section 3.10 provides the instructions for setting the reference identifiers for the vCenter.

FCS_TLSC_EXT.1.3
None defined.

2.1.16.3 Test Assurance Activities

The evaluator shall also perform the following tests:

FCS_TLSC_EXT.1.1
Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

The evaluator confirmed only supported ciphersuites were used in network connections.

FCS_TLSC_EXT.1.1
Test 2: The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.
The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

The evaluator presented a certificate that lacked Server Authentication extendedKeyUsage extension 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 cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

The evaluator presented a ECDSA certificate while still using an RSA cipher, verifying the connection failed.

FCS_TLSC_EXT.1.1

Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.

The evaluator configured the server to use NULL cipher and verified the connection failed.

FCS_TLSC_EXT.1.1

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

FCS_TLSC_EXT.1.1

Test 5.1: Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

The evaluator used a TLS tool to present an undefined TLS version to the TOE, resulting in a failed connection.

FCS_TLSC_EXT.1.1

Test 5.2: Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.

The evaluator used a TLS tool to present the most recent unsupported TLS version to the TOE, resulting in a failed connection.

FCS_TLSC_EXT.1.1

Test 5.3: [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.

The evaluator used a TLS tool to modify a byte in the server's nonce in the Server Hello handshake, resulting in a failed connection.

FCS_TLSC_EXT.1.1

Test 5.4: Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.

The evaluator used a TLS tool to modify the Server Hello's ciphersuite to an unsupported one, resulting in a failed connection.

FCS_TLSC_EXT.1.1

Test 5.5: [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake 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 signature block in a server's Key Handshake, resulting in a failed connection.

FCS_TLSC_EXT.1.1

Test 5.6: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.

The evaluator used a TLS tool to modify a byte in the Server Finished handshake message, resulting in a failed connection.

FCS_TLSC_EXT.1.1

Test 5.7: Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.

The evaluator used a TLS tool to send a message of random bytes after the ChangeCipherSpec message was sent, resulting in a failed connection.

Modified per TD0499.

The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7.

FCS_TLSC_EXT.1.2

Test 1: 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.

Note that 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 presented a server certificate whose CN did not match the reference identifier and no SAN, resulting in a failed connection.

FCS_TLSC_EXT.1.2

Test 2: 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.

The evaluator presented a server certificate whose CN matched the reference identifier, but the SAN did not. This caused the connection to fail.

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. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

The evaluator presented a certificate with a good CN and no SAN, resulting in a successful connection.

FCS_TLSC_EXT.1.2

Test 4: 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 presented a certificate whose CN did not match the reference identifier but the SAN did, resulting in a successful connection.

FCS_TLSC_EXT.1.2

Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.

FCS_TLSC_EXT.1.2

Test 5.1: [conditional]: If wildcards are supported, 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.

The evaluator presented a certificate with a wildcard that was not in the left-most label. The connection failed.

FCS_TLSC_EXT.1.2

Test 5.2: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (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. 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.come) and verify that the connection fails.

The evaluator presented a certificate with a single left-most label wildcard, resulting in a successful connection. The evaluator then presented a certificate without a left-most label. This connection failed. The evaluator then presented a certificate with two left-most labels. This too failed.

FCS_TLSC_EXT.1.2

Test 5.3: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The

evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.

The evaluator presented a certificate with a left-most label wildcard immediately preceding the public suffix, then configured the reference identifier with a single left-most label. This connection failed. The evaluator then configured the reference identifier with two left-most labels. This also failed.

FCS_TLSC_EXT.1.2

Test 5.4: [conditional]: If wildcards are not supported, 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 fails.

N/A – The TOE supports wildcards.

FCS_TLSC_EXT.1.2

Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

N/A – URI and service names are not supported.

FCS_TLSC_EXT.1.2

Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

N/A – Pinned certs are not supported.

The evaluator shall demonstrate that using an invalid certificate (unless excepted) results in the function failing as follows, unless excepted:

Modified per TD0513 (Test 1 replaced as follows).

FCS_TLSC_EXT.1.3

Test 1a: The evaluator shall demonstrate that a server using a certificate with a valid certification path successfully connects.

The evaluator attempted to connect using a valid certification path. This succeeded.

FCS_TLSC_EXT.1.3

Test 1b: The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.

The evaluator removed the intermediate CA from the validation path and attempted a connection. This connection failed.

FCS_TLSC_EXT.1.3

Test 1c [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.

N/A – The Trust Store is not managed on the TOE.

FCS_TLSC_EXT.1.3

Test 2: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.

The evaluator attempted a connection using a revoked certificate, resulting in a failed connection.

FCS_TLSC_EXT.1.3

Test 3: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.

The evaluator attempted a connection using an expired certificate, resulting in a failed connection.

FCS_TLSC_EXT.1.3

Test 4: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.

The evaluator attempted a connection using a certificate with invalid identifiers, resulting in a failed connection.

2.1.17 FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication [TLS Package]

2.1.17.1 TSS Assurance Activities

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication. The evaluator shall also ensure that the TSS describes any factors beyond configuration that are necessary in order for the client to engage in mutual authentication using X.509v3 certificates.

[ST] Section 6.4, “Identification and Authentication” describes the client-side certificates required as follows: when the TOE’s use of certificates is to present its own certificate to a remote endpoint, the certificate is chosen based on what the administrator has loaded into the Windows Certificate Store and given an alias that identifies its association with the TOE.

2.1.17.2 Guidance Assurance Activities

The evaluator shall ensure that the AGD guidance includes any instructions necessary to configure the TOE to perform mutual authentication. The evaluator also shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

[CCECG] Sections 2.5.3 and 3.1 provide instructions for configuring the certificates for the TOE and for Connection Server in external POD. These instructions include a setting to assign either “Server authentication” or “Client authentication” ensuring mutual authentication. Section 3.6 provides instructions for configuring remote administrator authentication to the Connection Server.

2.1.17.3 Test Assurance Activities

The evaluator shall also perform the following tests:

Test 1: The evaluator shall establish a connection to a server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake

The evaluator attempted a connection to a server not configured for mutual authentication and confirmed the TOE did not send a Client Certificate message during handshake.

Test 2: The evaluator shall establish a connection to a server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.

The evaluator attempted a connection to the server configured for mutual authentication and confirmed the TOE responded with a non-empty Client Certificate and Certificate Verify message.

2.1.18 FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension [TLS Package]

2.1.18.1 TSS Assurance Activities

The evaluator shall verify that the TSS describes the Supported Groups Extension

[ST] Section 6.2 states that for elliptic curve Diffie-Hellman, the TSF presents secp256r1, secp384r1, and secp521r1 in the Supported Groups extension as the parameter used for key establishment. For finite field Diffie-Hellman, the TSF presents the groups ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, and ffdhe8192 in the Supported Groups extension as the parameter used for key establishment.

2.1.18.2 Guidance Assurance Activities

None defined.

2.1.18.3 Test Assurance Activities

The evaluator shall also perform the following test:

Test 1: The evaluator shall configure a server to perform 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 connected to the server using each supported curve and group and verified that in each case the connection was successful.

2.1.19 FCS_TLSS_EXT.1 TLS Server Protocol [TLS Package]

2.1.19.1 TSS Assurance Activities

FCS_TLSS_EXT.1.1

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

The TOE supports the same set of cipher suites for both its TLS Client and its TLS Server connections. The list of cipher suites identified in the ST Section 6.2 is identical to the list in FCS_TLSS_EXT.1.1. See also Section 2.1.17.1 above for the list of ciphers.

FCS_TLSS_EXT.1.2
The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions consistent relative to selections in FCS_TLSS_EXT.1.2.

[ST] The TOE supports only TLS 1.2. Older SSL and TLS versions are not accepted.

FCS_TLSS_EXT.1.3
The evaluator shall verify that the TSS describes the key agreement parameters of the server's Key Exchange message.

[ST] Section 6.2 states, all supported cipher suites use finite field or elliptic curve ephemeral Diffie-Hellman as the method of key establishment. For finite field Diffie-Hellman, the TSF presents the groups ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, and ffdhe8192 in the Supported Groups extension as the parameter used for key establishment. For elliptic curve Diffie-Hellman, the TSF presents secp256r1, secp384r1, and secp521r1 in the Supported Groups extension as the parameter used for key establishment.

2.1.19.2 Guidance Assurance Activities

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

[CCECG] Section 2.5.3 instructs to enable FIPS mode that configures the TOE to use FIPS compliant Cryptography. Section 3.5 describes how to configure the TLS parameters to use the selected ciphersuites for both Server and Client TLS functions.

FCS_TLSS_EXT.1.2
The evaluator shall verify that the AGD guidance includes any configuration necessary to meet this requirement.

[CCECG] Section 3.5 states that TLS 1.2 is enabled by default; no separate configuration is needed.

FCS_TLSS_EXT.1.3
The evaluator shall verify that any configuration guidance necessary to meet the requirement must be contained in the AGD guidance.

The evaluators verified that all necessary configuration instructions to meet the requirement is contained in [CCECG] Section 3.5.

2.1.19.3 Test Assurance Activities

The evaluator shall also perform the following tests:

FCS_TLSS_EXT.1.1
Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to

satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

The evaluator confirmed only supported ciphersuites are used during TLS connections.

FCS_TLSS_EXT.1.1

Test 2: The evaluator shall send a Client Hello to the server with a list of cipher suites that does not contain any of the cipher suites 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 cipher suite and verify that the server denies the connection.

The evaluator sent a Client Hello using a NULL ciphersuite and confirmed the connection failed.

FCS_TLSS_EXT.1.1

Test 3: If RSA key exchange is used in one of the selected ciphersuites, the evaluator shall use a client to send a properly constructed Key Exchange message with a modified EncryptedPreMasterSecret field during the TLS handshake. The evaluator shall verify that the handshake is not completed successfully and no application data flows.

N/A – The TOE does not use RSA key exchange.

FCS_TLSS_EXT.1.1

Test 4: The evaluator shall perform the following modifications to the traffic:

Modified per TD0469 (Test 4.1 removed).

FCS_TLSS_EXT.1.1

Test 4.2: Modify a byte in the data of the client's Finished handshake message, and verify that the server rejects the connection and does not send any application data.

The evaluator used a TLS tool to modify a byte in the client's Finished Handshake message, resulting in a failed connection.

FCS_TLSS_EXT.1.1

Modified per TD0588.

Test 4.3: Demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption):

Test 4.3i [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 evaluator shall send a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.**
b. **The evaluator shall verify the server does not send a NewSessionTicket handshake message (at any point in the handshake).**
c. **The evaluator shall verify 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 evaluator shall complete the TLS handshake and capture the SessionID from the ServerHello.

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

Test 4.3ii [conditional]: If the TOE supports session resumption using session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2), the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

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

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

Test 4.3iii [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):

c. The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with a ServerHello with an empty SessionTicket extension, NewSessionTicket, ChangeCipherSpec and Finished messages (as seen in figure 2 of RFC 5077).

d. 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.

The evaluator connected to the TOE and observed it can resume sessions via Session ID. The evaluator then used a TLS tool to start a connection, send a Fatal Alert before the handshake finished, then used the

first connection's ID to resume a session. The evaluator confirmed the reused ID was rejected and a new one was issued during the full handshake.

FCS_TLSS_EXT.1.1

Test 4.4: Send a message consisting of random bytes from the client after the client has issued the ChangeCipherSpec message and verify that the server denies the connection.

The evaluator used a tool to send a message of random bytes after the client issued a ChangeCipherSpec message, resulting in a failed connection.

FCS_TLSS_EXT.1.2

Test 1: The evaluator shall send a Client Hello requesting a connection with version SSL 2.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 3.0 and TLS 1.0, and TLS 1.1 if it is selected.

The evaluator attempted to connect to the TOE using SSL 2.0, SSL 3.0, TLS v1.0, and TLS v1.1, and confirmed each connection failed.

FCS_TLSS_EXT.1.3

The evaluator shall conduct the following tests. The testing can be carried out manually with a packet analyzer or with an automated framework that similarly FCS_TLSS_EXT.2.1 FCS_TLSS_EXT.2.2 captures such empirical evidence. Note that this testing can be accomplished in conjunction with other testing activities. For each of the following tests, determining that the size matches the expected size is sufficient.

FCS_TLSS_EXT.1.3

Test 1: [conditional] If RSA-based key establishment is selected, the evaluator shall attempt a connection using RSA-based key establishment with a supported size. The evaluator shall verify that the size used matches that which is configured. The evaluator shall repeat this test for each supported size of RSA-based key establishment.

N/A – The TOE does not claim RSA-based key establishment.

FCS_TLSS_EXT.1.3

Test 2: [conditional] If finite-field (i.e. non-EC) Diffie-Hellman ciphers are selected, the evaluator shall attempt a connection using a Diffie-Hellman key exchange with a supported parameter size or supported group. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported parameter size or group.

The evaluator verified that each claimed Diffie-Hellman sizes was supported.

FCS_TLSS_EXT.1.3

Test 3: [conditional] If ECDHE ciphers are selected, the evaluator shall attempt a connection using an ECDHE ciphersuite with a supported curve. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported elliptic curve.

The evaluator verified that all claimed curves were supported.

2.1.20 FCS_TLSS_EXT.2 TLS Server Support for Mutual Authentication [TLS Package]

2.1.20.1 TSS Assurance Activity

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 6.4 “Identification and Authentication” describes the use of client-side certificates for TLS Server mutual authentication as follows. If mutual authentication is configured for the remote administrative interface (i.e. TOE is server), the TOE validates the presented TLS client certificate in this case. The TOE relies on the OS platform to validate presented TLS client certificates from remote cloud pods. When the TOE’s use of the certificate validation function is to validate the authenticity of remote endpoints, the TSF chooses what certificates to use based on what is presented to it as part of establishing the TLS session. For example, for mutual authentication with the TOE’s TLS Server, this would be the remote Connection Server cloud pod client certificates.

FCS_TLSS_EXT.2.3
If the product implements mutual authentication, the evaluator shall verify that the TSS describes how the DN and SAN in the certificate is compared to the expected identifier.

[ST] Section 6.2 states that as part of certificate validation in the establishment of TLS connectivity, the TOE will validate the reference identifier of a presented server certificate. This is done through comparison of the DN in the connection URL against the CN and SAN in the certificate. IP addresses are not supported.

2.1.20.2 Guidance Assurance Activity

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2
The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication. The evaluator shall ensure that the AGD guidance includes instructions for configuring the server to require mutual authentication of clients using these certificates.

[CCECG] Sections 2.5.3 and 3.1 provide instructions for configuring the certificates for the TOE and for Connection Server in external POD. These instructions include a setting to assign either “Server authentication” or “Client authentication” ensuring mutual authentication. Section 3.6 provides instructions for configuring remote administrator authentication to the Connection Server.

FCS_TLSS_EXT.2.3
If the DN is not compared automatically to the domain name, IP address, username, or email address, the evaluator shall ensure that the AGD guidance includes configuration of the expected identifier or the directory server for the connection.

Section 6.2 of the [ST] states that comparison of the reference identifier of a presented server certificate is done through comparison of the DN in the connection URL against the CN and SAN in the certificate. Configuration procedures are provided in [CCECG] Sections 2.5.3, 3.1, and 3.6.

2.1.20.3 Test Assurance Activity

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

The evaluator shall use TLS as a function to verify that the validation rules in FIA_X509_EXT.1.1 are adhered to and shall perform the following tests. The evaluator shall apply the AGD guidance to configure the server to require TLS mutual authentication of clients for the following tests, unless overridden by instructions in the test activity:

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 1: The evaluator shall configure the server to send a certificate request to the client. The client shall send a certificate_list structure which has a length of zero. The evaluator shall verify that the handshake is not finished successfully and no application data flows

The evaluator sent a zero-length certificate and verified the connection failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 2: The evaluator shall configure the server to send a certificate request to the client. The client shall send no client certificate message, and instead send a client key exchange message in an attempt to continue the handshake. The evaluator shall verify that the handshake is not finished successfully and no application data flows.

The evaluator used a TLS tool to not send a client certificate message and attempt to continue the handshake, resulting in a failed connection.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 3: 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 handshake is not finished successfully and no application data flows.

The evaluator used a TLS tool to send a certificate request with an unsupported signature algorithm, resulting in a failed connection.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 4: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.

The evaluator attempted a connection with the full certification path and verified it succeeded. The evaluator then broke the certification path and attempted a connection again. This connection failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 5: 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 in fact correspond to the CA certificate trusted by the TOE (meaning that the client certificate is invalid because its certification path does not in fact terminate in the claimed CA certificate). The evaluator shall verify that the attempted connection is denied.

The evaluator utilized a certificate signed by an imposter CA and attempted to connect to the TOE. This connection failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 6: 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 started a connection with certificate possessing Client Authentication purpose and successfully connected. The evaluator presented a certificate that lacked Client Authentication purpose and verified the connection failed.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 7: The evaluator shall perform the following modifications to the traffic: a) Configure the server to require mutual authentication and then modify a byte in the client's certificate. The evaluator shall verify that the server rejects 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. The evaluator shall verify that the server rejects the connection

The evaluator used a TLS tool to modify a byte in the client certificate, resulting in a failed connection. The evaluator used a TLS tool to modify a byte in the certificate's signature block, also resulting in a failed connection.

FCS_TLSS_EXT.2.3

Test 1: The evaluator shall send a client certificate with an identifier that does not match any of the expected identifiers and verify that the server denies the connection. The matching itself might be performed outside the TOE (e.g. when passing the certificate on to a directory server for comparison).

The evaluator used a client certificate whose identifier did not match any expected identifiers, resulting in a failed connection.

2.1.21 FCS_TLSS_EXT.4 TLS Server Support for Renegotiation [TLS Package]

2.1.21.1 TSS Assurance Activity

None defined.

2.1.21.2 Guidance Assurance Activity

None defined.

2.1.21.3 Test Assurance Activity

The following tests require connection with a client that supports secure renegotiation and the "renegotiation_info" extension.

Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that the "renegotiation_info" field is included in the ServerHello message.

The evaluator examined a PCAP and confirmed "renegotiation_info" field is included.

Test 2: The evaluator shall modify the length portion of the field in the ClientHello message in the initial handshake to be non-zero and verify that the server sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

The evaluator used a TLS tool to modify the length portion of the Client Hello to be non-zero, resulting in a failed connection.

Test 3: The evaluator shall modify the "client_verify_data" or "server_verify_data" value in the ClientHello message received during secure renegotiation and verify that the server terminates the connection.

The evaluator used a TLS tool to modify the ClientRenegotiation info and verified the connection failed.

2.2 User Data Protection (FDP)

2.2.1 FDP_DAR_EXT.1 Encryption of Sensitive Application Data

2.2.1.1 TSS Assurance Activity

The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application. The evaluator shall then ensure that the following activities cover all of the sensitive data identified in the TSS.

If **not store any sensitive data** is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory. The evaluator shall also ensure that this is consistent with the filesystem test below.

[ST] Section 6.3 identifies the sensitive data as credential data at rest in non-volatile storage. Sensitive data also includes log file data, which is protected at rest by VMware VM Encryption..

Section 6.2 identifies the credentials as:

- All certificates,
- Any credential where the TOE authenticates to an external system (e.g. vCenter). These include:
    - Per-domain per-administrator accounts for one-way and no-trust domain relationships
    - Per-domain service accounts for no-trust domains
    - Per-domain service accounts for VM domain join
    - Per-resource access credentials
    - vCenter CRUD service account
    - Event database CRUD service account
    - Network resource access credentials for Hybrid Logon user sessions.

2.2.1.2 Guidance Assurance Activity

None.

2.2.1.3 Test Assurance Activity

Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1.

The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted.

If “leverage platform-provided functionality” is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis.

Android: The evaluator shall inspect the TSS and verify that it describes how files containing sensitive data are stored with the MODE_PRIVATE flag set.

Microsoft Windows: The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.

Apple iOS: The evaluator shall inspect the TSS and ensure that it describes how the application uses the Complete Protection, Protected Unless Open, or Protected Until First User Authentication Data Protection Class for each data file stored locally.

Linux: The Linux platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

Oracle Solaris: The Solaris platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

Apple macOS: The macOS platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption clear to the end user.

The evaluator confirmed that user guidance has the required instructions to activate platform encryption prior to using the TOE. CCECG Section 2.5.1.4.1 indicates that to prepare the TOE for installation of the TOE, the user should enable VM encryption and provides a link to vSphere guidance. Test procedures were executed in accordance with a TOE that leverages platform-provided functionality and as approved by NIAP.

2.2.2 FDP_DEC_EXT.1 Access to Platform Resources

2.2.2.1 TSS Assurance Activity

FDP_DEC_EXT.1.1 and FDP_DEC_EXT.1.2

None.

2.2.2.2 Guidance Assurance Activities

FDP_DEC_EXT.1.1

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.

The evaluator inspected the user documentation and determined that it uses network connectivity. This is consistent with the selection made. [CCECG] Section 4.1 System Resources Used states that the TOE requires the use of the underlying operating system's network connectivity to process incoming connection requests, to set up and configure virtual machines running Horizon Agents, and to connect to external federated instances. It also requires the use of the underlying platform's log functionality to record its own behavior for diagnostic purposes.

FDP_DEC_EXT.1.2

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information repository which it accesses, identify the justification as to why access is required.

The evaluator inspected the user documentation and determined that it uses log functionality. This is consistent with the selection made. [CCECG] Section 4.1 System Resources Used states that the TOE requires the use of the underlying operating system's network connectivity to process incoming connection requests, to set up and configure virtual machines running Horizon Agents, and to connect to external federated instances. It also requires the use of the underlying platform's log functionality to record its own behavior for diagnostic purposes.

2.2.2.3 Test Assurance Activities

FDP_DEC_EXT.1.1

Android: The evaluator shall verify that each uses-permission entry in the AndroidManifest.xml file for access to a hardware resource is reflected in the selection.

Microsoft Windows: For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as ID_CAP_ISV_CAMERA, ID_CAP_LOCATION, ID_CAP_NETWORKING, ID_CAP_MICROPHONE, ID_CAP_PROXIMITY and so on. A complete list of Windows App permissions can be found at:

- http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of the required hardware resources.

Apple iOS: The evaluator shall verify that either the application or the documentation provides a list of the hardware resources it accesses.

Linux: The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Oracle Solaris: The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

Apple macOS: The evaluator shall verify that either the application software or its documentation provides a list of the hardware resources it accesses.

The evaluator verified [CCECG] Section 4.1 *System Resources Used* lists the hardware resources the TOE accesses.

FDP_DEC_EXT.1.2

Android: The evaluator shall verify that each uses-permission entry in the AndroidManifest.xml file for access to a sensitive information repository is reflected in the selection.

Microsoft Windows: For Windows Universal Applications the evaluator shall check the WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS,ID_CAP_APPOINTMENTS,ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:

- http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

Microsoft Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.

Apple iOS: The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Linux: The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Oracle Solaris: The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

Apple macOS: The evaluator shall verify that either the application software or its documentation provides a list of sensitive information repositories it accesses.

The evaluator verified [CCECG] Section 4.1 *System Resources Used* lists the sensitive information repositories the TOE accesses.

2.2.3 FDP_NET_EXT.1 Network Communications

2.2.3.1 TSS Assurance Activity

None.

2.2.3.2 Guidance Assurance Activity

None.

2.2.3.3 Test Assurance Activities

The evaluator shall perform the following tests:

Test 1: The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.

The evaluator started a normal connection and observed PCAP data, confirming all network traffic was documented in TSS or user-initiated.

Test 2: The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).

The evaluator ran a port scan and verified no unusual ports were open.

Android: If "no network communication" is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a uses-permission or usespermission-sdk-23 tag containing android:name="android.permission.INTERNET". In this case, it is not necessary to perform the above Tests 1 and 2, as the platform will not allow the application to perform any network communication.

N/A – The TOE does not support Android.

2.3 Identification and Authentication (FIA)

2.3.1 FIA_X509_EXT.1 X.509 Certificate Validation

2.3.1.1 TSS Assurance Activity

FIA_X509_EXT.1.1
The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

[ST] Section 6.4 describes how the X.509 certificates are validated by the TSF when establishing trusted communication in accordance with RFC 5280. The TOE uses X.509 certificates to validate the TLS server certificates of the environmental components that it communicates with, as well as TLS client certificates when the TOE is acting as a server for mutually-authenticated TLS connections. The TOE relies on the OS platform to validate presented TLS server certificates and to validate presented TLS client certificates from remote cloud pods. If mutual authentication is configured for the remote administrative interface, the TOE validates the presented TLS client certificate in this case.

FIA_X509_EXT.1.2
None.

2.3.1.2 Guidance Assurance Activity

FIA_X509_EXT.1.1 and FIA_X509_EXT.1.2
None.

2.3.1.3 Test Assurance Activities

FIA_X509_EXT.1.1

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. If the application supports chains of length four or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

FIA_X509_EXT.1.1

Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:

- by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
- by omitting the basicConstraints field in one of the issuing certificates,
- by setting the basicConstraints field in an issuing certificate to have CA=False,
- by omitting the CA signing bit of the key usage field in an issuing certificate, and
- by setting the path length field of a valid CA field to a value strictly less than the certificate path.

The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.

The evaluator presented a certificate modified for several conditions and verified each variant resulted in a failed connection.

FIA_X509_EXT.1.1

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

The evaluator attempted to connect to the TOE using an expired certificate, resulting in a failed connection.

FIA_X509_EXT.1.1

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-“conditional on whether CRL, OCSP, OCSP Stapling or OCSP Multi-stapling is selected; if multiple methods are selected, then the following tests shall be performed for each method:

- The evaluator shall test revocation of the node certificate.
- The evaluator shall also test revocation of an intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA), if intermediate CA certificates are supported. If OCSP stapling per RFC 6066 is the only supported revocation method, this test is omitted.
- 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.

The evaluator attempted to connect using a revoked certificate, resulting in a failed connection.

FIA_X509_EXT.1.1

Test 4: If any OCSP option 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 generated certificates that did not have the OCSP signing purpose and CRL purpose and signed revocation data with them. Using these certificates, the evaluator confirmed the connections failed.

FIA_X509_EXT.1.1

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.

FIA_X509_EXT.1.1

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

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

FIA_X509_EXT.1.1

Test 7: The evaluator shall modify any byte in the public key of 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 public key of the certificate and started a connection. The connection failed.

FIA_X509_EXT.1.1

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, 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.

FIA_X509_EXT.1.1

Test 9: (Conditional on support for EC certificates as indicated in FCS_COP.1/Sig). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate 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.

FIA_X509_EXT.1.2

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. If the application supports chains of length four

or greater, the evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

FIA_X509_EXT.1.2

Test 1: The evaluator shall ensure that the certificate of at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate without the basicConstraints extension to the TOE's trust store.

The evaluator started a connection in which one of the CAs in the validation chain does not contain the basicConstraints extension. The connection failed.

FIA_X509_EXT.1.2

Test 2: The evaluator shall ensure that the certificate of at least one of the CAs in the chain has the CA flag in the basicConstraints extension not set (or set to FALSE). The evaluator shall confirm that validation of the certificate path fails (i) as part of the validation of the peer certificate belonging to this chain; and/or (ii) when attempting to add the CA certificate with the CA flag not set (or set to FALSE) in the basicConstraints extension to the TOE's trust store.

The evaluator started a connection in which one of the CAs in the validation chain has its basicConstraints set to FALSE. The connection failed.

2.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication

2.3.2.1 TSS Assurance Activity

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 6.4 states that when the TOE's use of the certificate validation function is to validate the authenticity of remote endpoints, the TSF chooses what certificates to use based on what is presented to it as part of establishing the TLS session. When the TOE's use of certificates is to present its own certificate to a remote endpoint, the certificate is chosen based on what the administrator has loaded into the Windows Certificate Store and given an alias that identifies its association with the TOE.

The evaluator shall examine the TSS to confirm that it describes the behavior 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 operational guidance contains instructions on how this configuration action is performed.

[ST] Section 6.4 states that in the event that the revocation status of a certificate cannot be verified (i.e. the OCSP responder cannot be reached or the CRL is stale and a new one is unavailable), the TOE will reject the certificate.

2.3.2.2 Guidance Assurance Activity

None.

2.3.2.3 Test Assurance Activities

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

Test 1: 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 operational guidance to determine that all supported administrator-configurable options behave in their documented manner.

The evaluator shut off the revocation server then attempted to start a connection. The connection failed because the TOE was unable to validate certificates using the revocation server.

Test 2: The evaluator shall demonstrate that an invalid certificate that requires certificate validation checking to be performed in at least some part by communicating with a nonTOE IT entity cannot be accepted.

The evaluator attempted to connect to the TOE using an invalid certificate and was rejected.

2.4 Security Management (FMT)

2.4.1 FMT_CFG_EXT.1 Secure by Default Configuration

2.4.1.1 TSS Assurance Activity

FMT_CFG_EXT.1.1
The evaluator shall check the TSS to determine if the application requires any type of credentials and if the application installs with default credentials.

[ST] Section 6.5 states that the TOE is run locally as an application on the host platform. The user installing the TOE must be a domain user with local administrator privileges for the installation to be successful. When installing the TOE, the user will specify either their own account as the administrator for the remote web interface or they may specify any domain user or group. Administrative users are authenticated by the OS platform's interaction with an environmental Active Directory server that is used to validate their domain account; the TSF does not provide default credentials.

FMT_CFG_EXT.1.2
None.

2.4.1.2 Guidance Assurance Activity

FMT_CFG_EXT.1.1 and FMT_CFG_EXT.1.2
None.

2.4.1.3 Test Assurance Activities

If the application uses any default credentials the evaluator shall run the following tests.

FMT_CFG_EXT.1.1

Test 1: The evaluator shall install and run the application without generating or loading new credentials and verify that only the minimal application functionality required to set new credentials is available.

N/A – The TOE does not use default credentials.

FMT_CFG_EXT.1.1

Test 2: The evaluator shall attempt to clear all credentials and verify that only the minimal application functionality required to set new credentials is available.

N/A – The TOE does not use default credentials.

FMT_CFG_EXT.1.1

Test 3: The evaluator shall run the application, establish new credentials and verify that the original default credentials no longer provide access to the application.

N/A – The TOE does not use default credentials.

FMT_CFG_EXT.1.2

The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

Android: The evaluator shall run the command find -L . -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Microsoft Windows: The evaluator shall run the SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. For Windows Universal Applications the evaluator shall consider the requirement met because of the AppContainer sandbox.

Apple iOS: The evaluator shall determine whether the application leverages the appropriate Data Protection Class for each data file stored locally.

Linux: The evaluator shall run the command find -L . -perm /002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Oracle Solaris: The evaluator shall run the command find . \(-perm -002 \) inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

Apple macOS: The evaluator shall run the command find . -perm +002 inside the application's data directories to ensure that all files are not world-writable. The command should not print any files.

The evaluator ran icacls.exe and verified that file permissions do not allow standard users to modify the application or data files.

2.4.2 FMT_MEC_EXT.1 Supported Configuration Mechanism

2.4.2.1 TSS Assurance Activity

The evaluator shall review the TSS to identify the application's configuration options (e.g. settings) and determine whether these are stored and set using the mechanisms supported by the platform or implemented by the application in accordance with the PP-Module for File Encryption. At a minimum the TSS shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an SFR.

Conditional: If "implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption" is selected, the evaluator shall ensure that the TSS identifies those options, as well as indicates where the encrypted representation of these options is stored.

[ST] Section 6.5 states that most security-relevant configuration data is stored in an environmental LDAP directory in %ProgramData%. This includes policy settings for user sessions on Horizon Agents (session timeout value, login banner, forced logoff warning message), administrator role and privilege assignments, user entitlements to applications and desktops, and user entitlements to desktops. The Windows Registry also stores configuration information related to logging (path to log files, maximum days kept).

The TOE is managed through its web GUI. Some management functions relate to the configuration of the TOE itself, but most relate to the configuration of environmental components. Specifically, the primary purpose of the TOE is to facilitate connectivity between Horizon Clients and Horizon Agents. Configuration then largely revolves around the access that individual Horizon Client users are granted to resources that are managed by Horizon Agents. The following management functions are configurable via the GUI:

- Log bundle collection – Setting log levels, initiating/cancelling requests to collect logs, downloading log bundles to the local file system, and deleting logs.
- Administer policy, including idle session policy – configuring the types of resources that clients are globally allowed/not allowed to interact with and how long an active client session can remain idle before termination.
- Allocate roles to administrative users – configuring the administrative levels of privilege used to interact with the TOE's management functionality.
- Administer entitlements to resources– Configuring the Horizon Client users that are authorized to launch a particular resource on a Horizon Agent system.
- Helpdesk functions
  - Troubleshoot desktop or application sessions
  - View status of desktop or application sessions
  - Administration of helpdesk access to desktop resources
  - Perform remote assistance to TOE users on connected desktops
  - Perform maintenance operations on TOE user accounts
  - Disconnect and log off desktop or application sessions
  - Restart virtual desktop infrastructure VM
  - Send notification to published desktops or virtual desktop

2.4.2.2 Guidance Assurance Activity

None.

2.4.2.3 Test Assurance Activities

Modified per TD0624.

If "invoke the mechanisms recommended by the platform vendor for storing and setting configuration options" is chosen, the method of testing varies per platform as follows:

Android: The evaluator shall run the application and make security-related changes to its configuration. The evaluator shall check that at least **one file** at location /data/data/package/shared_prefs/ **(for SharedPreferences) and/or /data/data/package/files/datastore (for DataStore)** where the package is the Java package of the application. **For SharedPreferences the evaluator shall examine the XML file**

to make sure it reflects the changes made to the configuration to verify that the application used SharedPreferences and/or PreferenceActivity to store the configuration data. For DataStore the evaluator shall use a protocol buffer analyzer to examine the file to make sure it reflects the changes made to the configuration to verify that the application used DataStore to store the configuration data.

Microsoft Windows: The evaluator shall determine and verify that Windows Universal Applications use either the Windows.Storage namespace, Windows.UI.ApplicationSettings namespace, or the IsolatedStorageSettings namespace for storing application specific settings. For .NET applications, the evaluator shall determine and verify that the application uses one of the locations listed in https://docs.microsoft.com/en-us/dotnet/framework/configure-apps/ for storing application specific settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify that ProcMon logs show corresponding changes to the Windows Registry or C:\ProgramData\ directory.

Apple iOS: The evaluator shall verify that the app uses the user defaults system or key-value store for storing all settings.

Linux: The evaluator shall run the application while monitoring it with the utility strace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that strace logs corresponding changes to configuration files that reside in /etc (for systemspecific configuration), in the user's home directory (for user-specific configuration), or /var/lib/ (for configurations controlled by UI and not intended to be directly modified by an administrator).

Oracle Solaris: The evaluator shall run the application while monitoring it with the utility dtrace. The evaluator shall make security-related changes to its configuration. The evaluator shall verify that dtrace logs corresponding changes to configuration files that reside in /etc (for systemspecific configuration) or in the user's home directory(for user-specific configuration).

Apple macOS: The evaluator shall verify that the application stores and retrieves settings using the NSUserDefaults class.

If "implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption" is selected, for all configuration options listed in the TSS as being stored and protected using encryption, the evaluator shall examine the contents of the configuration option storage (identified in the TSS) to determine that the options have been encrypted.

The evaluator ran ProcMon while making changes to the TOE configuration and verified changes were saved in the Windows Registry or appropriate directories.

2.4.3 FMT_SMF.1 Specification of Management Functions

2.4.3.1 TSS Assurance Activity

None.

2.4.3.2 Guidance Assurance Activity

The evaluator shall verify that every management function mandated by the PP is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

The evaluator verified that every management function mandated by the PP is described in the operation guidance. [CCECG] Section 4.2 lists each of the functions and provides a description and link to online

guidance documentation with the steps to perform the management duties associated with the management function.

2.4.3.3 Test Assurance Activity

The evaluator shall test the application's ability to provide the management functions by configuring the application and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

The evaluator confirmed the security administrator was able to perform management functions such as log bundle collection, administer policy, allocate roles to administrative users, administer entitlements to resources, and help desk functions.

2.5 Privacy (FPR)

2.5.1 FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information

2.5.1.1 TSS Assurance Activity

The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted.

[ST] Section 6.6 states that the TOE's primary function is to connect authenticated Horizon Client users to the Horizon Agent resources that their organizational role entitles them to access. As such, the TOE only receives and handles username and IP address data from a particular user. The TOE does not have a resource to receive or transmit PII for either a user or administrator

2.5.1.2 Guidance Assurance Activity

None.

2.5.1.3 Test Assurance Activities

If require user approval before executing is selected, the evaluator shall run the application and exercise the functionality responsibly for transmitting PII and verify that user approval is required before transmission of the PII.

N/A – The TOE does not transmit PII.

2.6 Protection of the TSF (FPT)

2.6.1 FPT_AEX_EXT.1 Anti-Exploitation Capabilities

2.6.1.1 TSS Assurance Activity

FPT_AEX_EXT.1.1

The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

[ST] Section 6.7 states that the TOE implements address space layout randomization (ASLR) through the combination of the use of the /dynamicbase compiler flag for compiled code and the intrinsic memory management of the Java Runtime Environment (JRE) for Java code.

FPT_AEX_EXT.1.2, FPT_AEX_EXT.1.3, FPT_AEX_EXT.1.4, and FPT_AEX_EXT.1.5
None.

2.6.1.2 Guidance Assurance Activity

FPT_AEX_EXT.1.1, FPT_AEX_EXT.1.2, FPT_AEX_EXT.1.3, FPT_AEX_EXT.1.4, and FPT_AEX_EXT.1.5
None.

2.6.1.3 Test Assurance Activities

FPT_AEX_EXT.1.1
The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform. For those platforms requiring the same application running on two different systems, the evaluator may alternatively use the same device. After collecting the first instance of mappings, the evaluator must uninstall the application, reboot the device, and reinstall the application to collect the second instance of mappings.

Android: The evaluator shall run the same application on two different Android systems. Both devices do not need to be evaluated, as the second device is acting only as a tool. Connect via ADB and inspect /proc/PID/maps. Ensure the two different instances share no memory mappings made by the application at the same location.

Microsoft Windows: The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

Apple iOS: The evaluator shall perform a static analysis to search for any mmap calls (or API calls that call mmap), and ensure that no arguments are provided that request a mapping at a fixed address.

Linux: The evaluator shall run the same application on two different Linux systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Oracle Solaris: The evaluator shall run the same application on two different Solaris systems. The evaluator shall then compare their memory maps using pmap -x PID to ensure the two different instances share no mapping locations.

Apple macOS: The evaluator shall run the same application on two different Mac systems. The evaluator shall then compare their memory maps using vmmap PID to ensure the two different instances share no mapping locations.

The evaluator ran VMMap on two instances and verified they do not share mapping locations.

FPT_AEX_EXT.1.2
The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

Android: The evaluator shall perform static analysis on the application to verify that

- mmap is never invoked with both the PROT_WRITE and PROT_EXEC permissions, and
- mprotect is never invoked.

Microsoft Windows: The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the /NXCOMPAT flag was used during compilation to verify that DEP protections are enabled for the application.

Apple iOS: The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

Linux: The evaluator shall perform static analysis on the application to verify that both

- mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and
- mprotect is never invoked with the PROT_EXEC permission.

Oracle Solaris: The evaluator shall perform static analysis on the application to verify that both

- mmap is never be invoked with both the PROT_WRITE and PROT_EXEC permissions, and
- mprotect is never invoked with the PROT_EXEC permission.

Apple macOS: The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the PROT_EXEC permission.

The evaluator ran BinScope and verified NXCheck passed.

FPT_AEX_EXT.1.3

The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

Android: Applications running on Android cannot disable Android security features, therefore this requirement is met and no evaluation activity is required.

Microsoft Windows: If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-us/windows/security/threat-protection/windows-defenderexploit-guard/customize-exploit-protection.

If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

Apple iOS: Applications running on iOS cannot disable security features, therefore this requirement is met and no evaluation activity is required.

Linux: The evaluator shall ensure that the application can successfully run on a system with either SELinux or AppArmor enabled and in enforce mode.

Oracle Solaris: The evaluator shall ensure that the application can run with Solaris Trusted Extensions enabled and enforcing. Platforms:Apple macOS... The evaluator shall ensure that the application can successfully run on macOS without disabling any security features.

The evaluator examined Windows Exploit Guard and confirmed the TOE functioned normally with all exploit protections active.

FPT_AEX_EXT.1.4

The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:

Android: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored under /data/data/package/ where package is the Java package of the application.

Microsoft Windows: For Windows Universal Applications the evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Apple iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

Linux: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Oracle Solaris: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

Apple macOS: The evaluator shall run the program, mimicking normal usage, and note where all user-modifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files.

The evaluator ran the TOE as normal and confirmed no executable files were stored in the same directories as user-modifiable files.

FPT_AEX_EXT.1.5

The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

Microsoft Windows: Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS and verify that the /GS flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of /GS.

For PE, the evaluator will disassemble each and ensure the following sequence appears:

```
mov rcx, QWORD PTR [rsp+(...)]
xor rcx, (...)
call (...)
```

.

For ELF executables, the evaluator will ensure that each contains references to the symbol **__stack_chk_fail**.

Tools such as Canary Detector may help automate these activities.

The evaluator ran BinScope and verified GSCheck passed.

2.6.2 FPT_API_EXT.1 Use of Supported Services and APIs

2.6.2.1 TSS Assurance Activity

The evaluator shall verify that the TSS lists the platform APIs used in the application.

[ST] Appendix A.1 lists the APIs used by each platform version of the TOE.

2.6.2.2 Guidance Assurance Activity

None.

2.6.2.3 Test Assurance Activity

The evaluator shall then compare the list with the supported APIs (available through e.g. developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

The evaluator verified all listed APIs are supported.

2.6.3 FPT_IDV_EXT.1 Software Identification and Versions

2.6.3.1 TSS Assurance Activity

If "other version information" is selected the evaluator shall verify that the TSS contains an explanation of the versioning methodology

[ST] Section 6.7 describes the TOE versioning as using both YYMM date-based versioning to correspond to the approximate release of a particular version and major/minor release versioning. For example, 2209 refers to a version released on or around September of 2022 and is also synonymous with version 8.7; SWID is not used.

2.6.3.2 Guidance Assurance Activity

None.

2.6.3.3 Test Assurance Activities

The evaluator shall install the application, then check for the existence of version information. If SWID tags is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

The evaluator confirmed the existence of version information.

2.6.4 FPT_LIB_EXT.1 Use of Third Party Libraries

2.6.4.1 TSS Assurance Activity

None.

2.6.4.2 Guidance Assurance Activity

None.

2.6.4.3 Test Assurance Activities

The evaluator shall install the application and survey its installation directory for dynamic libraries. The evaluator shall verify that libraries found to be packaged with or employed by the application are limited to those in the assignment.

The evaluator examined the installation directory and found no unexpected dynamic libraries.

2.6.5 FPT_TUD_EXT.1 Integrity for Installation and Update

2.6.5.1 TSS Assurance Activities

FPT_TUD_EXT.1.1, FPT_TUD_EXT.1.2, and FPT_TUD_EXT.1.3
None.

FPT_TUD_EXT.1.4
The evaluator shall verify that the TSS identifies how updates to the application are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.

[ST] Section 6.7 states that the TOE is packaged as an .msi file and signed by VMware using 2048-bit RSA. The TOE is updated by acquiring the update package from outside the product (e.g. through the VMware support site).

FPT_TUD_EXT.1.5
The evaluator shall verify that the TSS identifies how the application is distributed. If "with the platform" is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If "as an additional package" is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

[ST] Section 6.7 states that the TOE is a standalone application that is not natively bundled as part of a host OS. "as an additional package" is selected, the evaluator performed the tests in FPT_TUD_EXT.2.

2.6.5.2 Guidance Assurance Activities

FPT_TUD_EXT.1.1
The evaluator shall check to ensure the guidance includes a description of how updates are performed.

[CCECG] Section 2.6 refers to the “Upgrading Horizon Connection Server” section of the Horizon Installation and Upgrade guide which includes a description of how updates are performed.

FPT_TUD_EXT.1.2
The evaluator shall verify guidance includes a description of how to query the current version of the application.

[CCECG] Section 2.5.4 includes a description of how to query the current version of the application.

FPT_TUD_EXT.1.3, FPT_TUD_EXT.1.4, and FPT_TUD_EXT.1.5
None.

2.6.5.3 Test Assurance Activities

FPT_TUD_EXT.1.1
The evaluator shall check for an update using procedures described in either the application documentation or the platform documentation and verify that the application does not issue an error. If it is updated or if it reports that no update is available this requirement is considered to be met.

The evaluator successfully updated the TOE following application documentation.

FPT_TUD_EXT.1.2
The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

The evaluator was able to query the current version according to operational guidance.

FPT_TUD_EXT.1.3
The evaluator shall verify that the application's executable files are not changed by the application.

Apple iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

N/A – The TOE does not support Apple iOS.

FPT_TUD_EXT.1.3
For all other platforms, the evaluator shall perform the following test:

Test 1: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

The evaluator gathered hash data of all executable files, ran the TOE as normal, then saved a second set of hash data. The two sets matched.

FPT_TUD_EXT.1.4 and FPT_TUD_EXT.1.5
None.

2.6.6 FPT_TUD_EXT.2 Integrity for Installation and Update

2.6.6.1 TSS Assurance Activity

FPT_TUD_EXT.2.1 and FPT_TUD_EXT.2.2
None.

FPT_TUD_EXT.2.3
The evaluator shall verify that the TSS identifies how the application installation package is signed by an authorized source. The definition of an authorized source must be contained in the TSS.

[ST] Section 6.7 states that the TOE is packaged as an .msi file and signed by VMware using 2048-bit RSA.

2.6.6.2 Guidance Assurance Activity

FPT_TUD_EXT.2.1, FPT_TUD_EXT.2.2, and FPT_TUD_EXT.2.3

None.

2.6.6.3 Test Assurance Activities

Modified per TD0628.

FPT_TUD_EXT.2.1

If a container image is claimed the evaluator shall verify that application updates are distributed as container images.

If the format of the platform-supported package manager is claimed, the evaluator shall verify that application updates are distributed in the **correct** format. This varies per platform:

Android: The evaluator shall ensure that the application is packaged in the Android application package (APK) format.

Microsoft Windows: The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the Windows Universal Application package (.APPX) format. See https://msdn.microsoft.com/en-us/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.

Apple iOS: The evaluator shall ensure that the application is packaged in the IPA format.

Linux: The evaluator shall ensure that the application is packaged in the format of the package management infrastructure of the chosen distribution. For example, applications running on Red Hat and Red Hat derivatives shall be packaged in RPM format. Applications running on Debian and Debian derivatives shall be packaged in DEB format.

Oracle Solaris: The evaluator shall ensure that the application is packaged in the PKG format.

Apple macOS: The evaluator shall ensure that application is packaged in the DMG format, the PKG format, or the MPKG format.

The evaluator inspected the installer file and confirmed it is in .exe format.

FPT_TUD_EXT.2.2

Android: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

Microsoft Windows: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

Apple iOS: The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox).

Linux: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

Oracle Solaris: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

Apple macOS: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

The evaluator recorded the path of all files within a system, installed the TOE, ran it normally, uninstalled it, then recorded the path of all files again. The evaluator confirmed no additional files were generated by the TOE.

FPT_TUD_EXT.2.3

None.

2.7 Trusted Path/Channels (FTP)

2.7.1 FTP_DIT_EXT.1 Protection of Data in Transit

2.7.1.1 TSS Assurance Activity

For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.

[ST] Section 5.2.7.1 does not identify platform-provided protection of data in transit and therefore this is not applicable.

2.7.1.2 Guidance Assurance Activity

None.

2.7.1.3 Test Assurance Activities

The evaluator shall perform the following tests.

Test 1: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS, TLS, DTLS, SSH, or IPsec in accordance with the selection in the ST.

The evaluator inspected a PCAP of a normal connection and verified all traffic was in TLS.

Test 2: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.

The evaluator inspected a PCAP of a normal connection and verified no sensitive data was transmitted in the clear.

Test 3: The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

The evaluator inspected a PCAP of a normal connection and verified no plaintext credentials were transmitted.

Android: If "not transmit any data" is selected, the evaluator shall ensure that the application's AndroidManifest.xml file does not contain a uses-permission or usespermission-sdk-23 tag containing android:name="android.permission.INTERNET". In this case, it is not necessary to perform the above Tests 1, 2, or 3, as the platform will not allow the application to perform any network communication.

Apple iOS: If "encrypt all transmitted data" is selected, the evaluator shall ensure that the application's Info.plist file does not contain the NSAllowsArbitraryLoads or NSExceptionAllowsInsecureHTTPLoads keys, as these keys disable iOS's Application Transport Security feature.

N/A – The TOE is does not support Android.

3 Security Assurance Requirements

3.1 Class ASE: Security Target

As per ASE activities define in [CEM].

3.2 Class ADV: Development

3.2.1 ADV_FSP.1 Basic Functional Specification

There are no specific evaluation activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 2 Security Functional Requirement Assurance Activities, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other evaluation activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

The Assurance Activities identified above provided sufficient information to determine the appropriate content for the TSS section and to perform the assurance activities. Since these are directly associated with the SFRs, and are implicitly already done, no additional documentation or analysis is necessary.

3.3 Class AGD: Guidance Documents

3.3.1 AGD_OPE.1 Operational User Guidance

3.3.1.1 TSS Assurance Activity

None defined.

3.3.1.2 Guidance Assurance Activity

Some of the contents of the operational guidance will be verified by the evaluation activities in Section 2 Security Functional Requirement Assurance Activities and evaluation of the TOE according to the [CEM]. The following additional information is also required.

If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The documentation must describe the process for verifying updates to the TOE by verifying a digital signature – this may be done by the TOE or the underlying platform.

The evaluator shall verify that this process includes the following steps:

- 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).
- Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the digital signature. The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

[CCECG] Section 3 subsections provide instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. Section 3 provides a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

[CCECG] Section 2.6 states that Software updates are acquired from the VMware site in the same manner as the initial installation package. Section 2.5.2 describes how to obtain software. The detailed instructions for installing an update are provided in the "Upgrading Horizon Connection Server" section of the Horizon Installation and Upgrade guide. The process uses an installer which handles the specifics of making the for making the update accessible to the TOE. Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful (i.e. it includes checking the installed version). This includes generation of the digital signature. [CCECG] Section 2.5.4 describes how the installer is signed by VMware and checked at installation time; a failed integrity check will prevent installation of the application.

Section 1.3 describes features and functions not included in the TOE evaluation and states that the product was evaluated against applicable requirements in the Protection Profile for Application Software and Functional Package for Transport Layer Security (TLS).

3.3.1.3 Test Assurance Activity

None defined.

3.3.2 AGD_PRE.1 Preparative Procedures

3.3.2.1 TSS Assurance Activity

None defined.

3.3.2.2 Guidance Assurance Activity

As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

[AGD Notes] The TOE's platform is a Windows Server 2019, virtualized on VMware ESXi 7.0. The provided guidance adequately covers this platform.

3.3.2.3 Test Assurance Activity

None defined.

3.4 Class ALC: Life-Cycle Support

3.4.1 ALC_CMC.1 Labeling of the TOE

3.4.1.1 TSS Assurance Activity

None defined.

3.4.1.2 Guidance Assurance Activity

The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Section 1.1 of [ST] ("Security Target, TOE and CC Identification") includes the TOE identification. The TOE is identified as VMware Horizon Connection Server 8 2209 (Horizon 8.7). [ST] also describes the TOE versioning as using YYMM date-based versioning to correspond to the approximate release of a particular version and major/minor release versioning, e.g. 2209 refers to the TOE version released on or around September of 2022 and is also synonymous with version 8.7. The evaluator checked the AGD guidance and TOE samples received for testing and observed that the version number is consistent with that in the ST; and that the information provided in the ST is sufficient to distinguish the product on the vendor's website.

3.4.1.3 Test Assurance Activity

None defined.

3.4.2 ALC_CMS.1 TOE CM Coverage

3.4.2.1 TSS Assurance Activity

The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements.

By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the evaluation activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an in-depth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

As described in Section 3.4.1.2 above, the evaluator confirmed the TOE is labelled with its unique software version identifier.

3.4.2.2 Guidance Assurance Activity

The evaluator shall ensure that the developer has identified (in guidance documentation for application developers concerning the targeted platform) one or more development environments appropriate for use in developing applications for the developer's platform. For each of these development environments, the developer shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler flags). The evaluator shall ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator shall ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation

provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

Section 6.7 of [ST] ("Protection of the TSF") describes how the TOE uses security features provided by the Windows platform. This includes address space layout randomization (ASLR) through the combination of the use of the /dynamicbase compiler flag for compiled code and the intrinsic memory management of the Java Runtime Environment (JRE) for Java code and relies fully on its underlying host platforms to perform memory mapping. The TOE is compiled with stack overflow protection through the use of the /GS compiler flag, data execution protection, and stack-based buffer overflow protection. The TOE is packaged as an .msi file, the compilation has already been done by default. As described in Section **Error! Reference source not found.**.2 above, the evaluator confirmed the TOE is labelled with its unique software version identifier.

3.4.2.3 Test Assurance Activity

None defined.

3.4.3 ALC_TSU_EXT.1 Timely Security Updates

3.4.3.1 TSS Assurance Activity

The evaluator shall verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator shall verify that this description addresses the entire application. The evaluator shall also verify that, in addition to the TOE developer's process, any third-party processes are also addressed in the description. The evaluator shall also verify that each mechanism for deployment of security updates is described.

The evaluator shall verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the TOE patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator shall verify that this time is expressed in a number or range of days.

The evaluator shall verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the TOE. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

Section 6.1 of [ST] ("Timely Security Updates") describes the timely security update process used by the developer to create and deploy TOE security updates. The description encompasses the entirety of the TOE.

VMware uses an internal classification system to categorize product security flaws by severity level. The levels are critical, important, moderate, and low. The standard release cycle for VMware products is quarterly, so all Moderate and Low findings are typically resolved within a maximum of 90 days, while more significant findings are generally resolved in less time (i.e. <90 days).

VMware provides an email address (security@vmware.com) that is used for the reporting of potential security findings. VMware encourages the use of Pretty Good Privacy (PGP) to encrypt any communications sent to this email address and provides a copy of their PGP public key at https://kb.vmware.com/s/article/1055.

VMware staff identifies potential vulnerabilities through third-party researchers reporting potential flaws via email, reports from field personnel, reports from customers, and monitoring of public vulnerability sites. When a report is received, VMware attempts to reproduce the finding and determine its severity. If a finding is discovered for which there is no current fix, VMware will publish a Knowledge Base article about the finding as well as any potential workarounds that may be used until an updated version of the product can be delivered.

Both quarterly releases and mid-cycle patches can be obtained from https://customerconnect.vmware.com. All security updates to the TOE are delivered as part of the next planned maintenance release of the product and important security updates will be released as a patch if appropriate to do so. Critical fixes or corrective action is begun immediately and will be made available in the shortest commercially reasonable time.

3.4.3.2 Guidance Assurance Activity

None defined.

3.4.3.3 Test Assurance Activity

None defined.

3.5 Class ATE: Tests

3.5.1 ATE_IND.1 Independent Testing – Conformance

3.5.1.1 TSS Assurance Activity

None defined.

3.5.1.2 Guidance Assurance Activity

None defined.

3.5.1.3 Test Assurance Activity

The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's evaluation activities.

While it is not necessary to have one test case per test listed in an evaluation activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no effect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be

provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform.

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (e.g SSH). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

The TOE was tested at Leidos's Columbia, MD location. The procedures and results of this testing are available in the DTR document.

3.6 Class AVA: Vulnerability Assessment

3.6.1 AVA_VAN.1 Vulnerability Survey

3.6.1.1 TSS Assurance Activity

None defined.

3.6.1.2 Guidance Assurance Activity

None defined.

3.6.1.3 Test Assurance Activity

Modified per TD0554 (including deletion of last sentence in first paragraph).

The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses.

The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

For Windows, Linux, macOS and Solaris: The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious.

The evaluation team performed a search of the following online sources:

- National Vulnerability Database (https://nvd.nist.gov/),
- OpenSSL.org (https://www.openssl.org/news/vulnerabilities.html), and

- VMware's Security Advisories page: https://www.vmware.com/security/advisories.html6

The evaluation team performed searches on 4/26/2023 and again on 6/15/2023. Some search terms were added and searched on 6/27/2023. The following search terms were used:

1. VMware Horizon
2. Horizon Connection Server
3. VMware's BC-FJA (Bouncy Castle FIPS Java API) 1.0.2.3
4. Centralized content server
5. Enterprise resource delivery
6. Enterprise content delivery
7. OpenSSL 1.0.2zg (third party library)
8. Third Party Libraries identified in Section A.2 of the Security Target

No vulnerabilities were identified for the TOE.

The evaluator ran a virus scan with up to date virus definitions against the Windows TOE executables and verified that no files were flagged as malicious.

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