Assurance Activities Report

for

VMware Horizon Client 8 2209 (Horizon 8.7)

Version 1.0

13 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:

National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

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 Client 8 2209 (Horizon 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] with the following optional and selection-based SFRs: FCS_CKM.1/AK, FCS_CKM.1/SK, FCS_CKM.2, FCS_COP.1/SKC, FCS_COP.1/Hash, FCS_COP.1/Sig, FCS_COP.1/KeyedHash, FCS_HTTPS_EXT.1/Client, FCS_RBG_EXT.2, FIA_X509_EXT.1, FIA_X509_EXT.2; and the Functional Package for Transport Layer Security (TLS), Version 1.1, 2019-02-12 [TLS Package] with the following optional and selection-based SFRs: FCS_TLSC_EXT.1, FCS_TLSC_EXT.2, FCS_TLSC_EXT.3, and FCS_TLSC_EXT.5.

1.1 Applicable Technical Decisions

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

1.2 Evidence

[ST]	VMware Horizon Client 8 2209 (Horizon 8.7) Security Target, Version 1.0, 4 April 2023
[CCECG]	VMware Horizon Client 8 2209 (Horizon 8.7) Common Criteria (CC) Evaluated Configuration Guide Document Version 1.0, Document Date April 4, 2023
[ADMIN]	VMware Horizon 2209 Horizon Administration, 2022
[INSTALL]	VMware Horizon 2209 Horizon Installation and Upgrade, 2022
[SECURITY]	VMware Horizon 2209 Horizon Security, 2022
[OAD]	VMware Horizon 2209 Horizon Overview and Deployment Planning, 2022
[WUG]	VMware Horizon Client for Windows 2209, Horizon Client for Windows User Guide 2022
[AUG]	VMware Horizon Client for Android 2209, Horizon Client for Android User Guide 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 OpenSSL cryptographic library included within the TOE boundary. Both platform versions of the TOE use VMware's OpenSSL FIPS Object Module 2.0.20-vmw. 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 curve P-384)	OpenSSL	FIPS PUB 186-4	A1292
FCS_CKM.2 Cryptographic Key Establishment			
Elliptic curve-based key establishment	OpenSSL	NIST SP 800-56A	A1292
FCS_COP.1/Hash Cryptographic Operation – Hashing			
SHA-384 (digest size 384 bits)	OpenSSL	FIPS PUB 180-4	A1292
FCS_COP.1/KeyedHash Cryptographic Operation – Keyed Hash Message Authentication			
HMAC-SHA-384	OpenSSL	FIPS PUB 198-1 FIPS PUB 180-4	A1292
FCS_COP.1/Sig Cryptographic Operation – Signing			
RSA (2048, 3072-bit)	OpenSSL	FIPS PUB 186-4, Section 5	A1292
FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption			
AES-GCM (256 bits)	OpenSSL	GCM as defined in NIST SP 800-38D	A1292
FCS_RBG_EXT.2 Random Bit Generation from Application			
AES-CTR_DRBG (256 bits)	OpenSSL	NIST SP 800-90A NIST SP 800-57	A1292

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[1] Cryptographic Key Generation Services

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 inspected the application and [CCECG] and determined that the application requires and implements asymmetric key generation services in support of TLS communications as described in [ST] Section 6.2 and specified in Section 5.2.1.1. Accordingly, the evaluation activities as stated in the selection-based FCS_CKM.1/AK requirement have been performed (see section 2.1.2.1).

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 identifies the supported key sizes for asymmetric key generation as ECC keys using P-384. ECC key pair generation (NIST curves P-384) in accordance with FIPS PUB 186-4 is the only supported scheme identified.

[1] Modified by TD0717

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 indicates that the TOE "implements functionality" for asymmetric key generation and 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] Sections 3.2.2.1 and 3.2.2.2 provide instructions for configuring the Android and Windows versions of the TOE to use the key generation scheme and key size for all defined uses.

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.

The TOE includes ACVP-certified VMware OpenSSL FIPS Object Module 2.0.20-vmw. 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 describes how the TOE uses its DRBG to generate 256-bit symmetric keys used for AES. To ensure sufficient key strength, the TOE implements DRBG functionality for key generation, using the AES-CTR_DRBG. The proprietary Entropy Analysis Report (EAR) describes how the TSF extracts random data from software-based sources to ensure that an amount of entropy that is at least equal to the strength of the generated keys is present (i.e., at least 256 bits when the largest supported keys are generated) when seeding the DRBG for key generation purposes. The Windows platform version of the TOE relies on a third-party entropy source provided by the platform vendor. The Android platform version of the TOE relies on the OS platform entropy source. Specifically, random numbers are obtained from the following platform APIs, depending on the platform used:

- Windows: BCryptGenRandom
- Android: invocation of /dev/random pseudo-device

In both cases, it is assumed that these platforms provide at least 256 bits of entropy.

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 6.2 ("Cryptographic Support") indicates the TOE supports the Elliptic curve-based key establishment scheme ECDHE; used for TLS/HTTPS communications. This is consistent with FCS_CKM.1.1/AK which also identifies ECC Schemes.

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] Sections 3.2.2.1 and 3.2.2.2 provide instructions for configuring the Android and Windows versions of the TOE to use the selected key establishment scheme.

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.

The TOE includes ACVP-certified VMware OpenSSL FIPS Object Module 2.0.20-vmw. 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] Section 6.2 indicates that SHA-384 is used in the signature_algorithms extension as the hash parameter used for digital signatures. SHA-384 is used to support the ECDHE key establishment schemes used for TLS/HTTPS communications.

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 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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 OpenSSL FIPS Object Module 2.0.20-vmw. 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 OpenSSL FIPS Object Module 2.0.20-vmw. 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 OpenSSL FIPS Object Module 2.0.20-vmw. 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] Sections 3.2.2.1 and 3.2.2.2 provide instructions for configuring the Android and Windows versions of the TOE to use the required mode and key size.

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-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.

The TOE includes ACVP-certified VMware OpenSSL FIPS Object Module 2.0.20-vmw. 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 that the TOE's implementation of HTTPS conforms to RFC 2818. TOE uses TLS 1.2, both standalone and as part of HTTPS, for client communications. Additionally, the TLS implementation conforms to RFC 5246.

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.

Evidence for secured connection via TLS seen in FCS_TLSC_EXT.1.1 Test 1.

FCS_HTTPS_EXT.1.2/Client

Other tests are performed in conjunction with the TLS package.

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_RBG_EXT.1 Random Bit Generation Services

2.1.10.1 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.

[ST] “use no DRBG functionality” is not selected in FCS_RBG_EXT.1.1 in Section 5.2.1.10. 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.

[ST] Section 5.2.1.10: “implement DRBG functionality” is selected. The evaluator confirmed that the additional FCS_RBG_EXT.2 elements are included in the ST.

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.

[ST] Section 5.2.1.10: “invoke platform-provided DRBG functionality” is not selected. Therefore this activity is not applicable.

2.1.10.2 Guidance Assurance Activity

None.

2.1.10.3 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).

“invoke platform-provided DRBG functionality**”** is not selected. Therefore this activity is not applicable.

2.1.11 FCS_RBG_EXT.2 Random Bit Generation from Application

2.1.11.1 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 ensured that the documentation includes the required details in sections: design description, entropy justification, operating conditions, and health testing. The information is sufficient to understand the entropy source and why it can be relied upon to provide sufficient entropy.

2.1.11.2 Guidance Assurance Activity

FCS_RBG_EXT.2.1 and FCS_RBG_EXT.2.2

None.

2.1.11.3 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 OpenSSL FIPS Object Module 2.0.20-vmw. See Section 1.4 for the applicable ACVP certificates.

2.1.12 FCS_STO_EXT.1 Storage of Credentials

2.1.12.1 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 credential data. Both the Windows and Android platform versions store the private key for the TLS client certificate. The Windows platform version uses the Windows Certificate Store. The Android platform version uses the OS keychain.

2.1.12.2 Guidance Assurance Activity

None.

2.1.12.3 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.

Windows

The evaluator examined the TOE environment and confirmed certificates were stored in Windows Certificate Store.

Android

The evaluator examined the TOE environment and confirmed certificates were stored within the Android Keychain.

2.1.13 FCS_TLS_EXT.1 TLS Protocol [TLS Package]

2.1.13.1 TSS Assurance Activities

None.

2.1.13.2 Guidance Assurance Activities

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

In Section 5.2.1.13 of [ST] ("FCS_TLS_EXT.1 TLS Protocol (TLS Package)"), "TLS as a client" is selected in FCS_TLS_EXT.1.1. The ST includes FCS_TLSC_EXT.1 from the selection-based requirements in [PKG_TLS_V1.1], consistent with the selection made in FCS_TLS_EXT.1.1.

2.1.13.3 Test Assurance Activities

None.

2.1.14 FCS_TLSC_EXT.1 TLS Client Protocol [TLS Package]

2.1.14.1 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 states that the TLS client offers the following cipher suite in its evaluated configuration:

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

This cipher suite is the same as the cipher suite specified in the SFR.

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 DNS name presented in the Subject Alternative Name (SAN) certificate field to the hostname of the server. 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.

The selection in FCS_TLSC_EXT.1.3 in [ST] Section 5.2.1.14 is "with no exceptions" and therefore this activity is not applicable.

2.1.14.2 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] Sections 3.2.2.1 and 3.2.2.2 provide instructions for configuring the Android and Windows versions of the TOE so that TLS conforms to the description in the TSS.

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] Sections 3.2.3.1 and 3.2.3.2 indicate that for both Windows and Android, the reference identifier is set as the DNS name of the server contained in the SAN of the certificate; and that this is checked automatically when PKI and Thumbprint Verification is enabled.

FCS_TLSC_EXT.1.3
None defined.

2.1.14.3 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 opened a TLS connection from the TOE to a VMware UAG device using each of the ciphers claimed in the ST. A wire capture was made of each of these connection attempts showing the successful use of the claimed cipher.

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 connected to the TOE using a certificate that lacked Server Auth extendedKeyUsage extension, resulting in a failed connection.

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 used a TLS tool to send a server certificate that did not match the server-selected ciphersuite, resulting in a failed connection.

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 used a TLS tool that attempted a connection with the TOE using the TLS_NULL_WITH_NULL_NULL ciphersuite and confirmed that the connection was denied.

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 the TLS tool to send a Server Hello using an undefined TLS version. The connection was denied.

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 the TLS tool to send a Server Hello using an unsupported TLS version. The connection was denied.

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 the TLS tool to modify a byte in the server's nonce in the Server Hello. The connection was denied.

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 the TLS tool to modify the Server Hello's selected ciphersuite to an unsupported cipher. The connection was denied.

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 the TLS tool to modify the signature block in the server's Key Exchange handshake message. The connection was denied.

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 the TLS tool to modify a byte in the Server Finished handshake message. The connection was denied.

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 the TLS tool to send a garbled message to the TOE after the ChangeCipherSpec message. The connection broke as a result.

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 had no SAN. The connection was denied.

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 SAN did not. The connection was denied.

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 server certificate whose CN and SAN matched the reference identifier. The connection succeed.

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 server certificate whose CN does not match the reference identifier, but SAN was appropriate. The connection succeeded.

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 server certificate containing a wildcard that was not in the left-most label. The connection was denied.

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 that contained a single left-most label. This connection succeeded. The test was repeated with a certificate without a left-most label, and this connection failed. The test was repeated with a certificate with two left-most labels, and this connection 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 wildcard in the left-most label immediately preceding the public suffix. This connection failed. The test was repeated using a certificate two left-most labels. This connection 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 – Wildcards are supported.

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 name reference identifiers 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 certificates 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 presented a viable certificate path, resulting in a successful connection.

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 broke the certificate chain by removing the intermediate certificate. The connection using this broken chain 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 TOE does not have its own trust store.

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 presented 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 presented an expired certificate, resulting in a broken 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 presented multiple certificates containing a variety of invalid identifier results. Each invalid certificate resulted in a failed connection.

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

2.1.15.1 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 use of client-side certificates for TLS mutual authentication for FIA_X509_EXT.2.1. The client certificate that the TOE presents to a server when performing TLS mutual authentication is user-configured. For the Windows platform version of the TOE, this is expected to be a certificate stored on a physical smart card and unlocked with a PIN that is associated with the certificate. For the Android platform version of the TOE, the user configures a certificate to be associated with a virtual smart card that is unlocked using a derived credential that is generated from a PIN.

2.1.15.2 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 3.2.1.1 and 3.2.1.2 provide instructions for configuring TLS client authentication, including configuring a smart card as the source of the X.509 client certificate used for mutual authentication.

2.1.15.3 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 connected to a server not configured for mutual authentication. The TOE did not send a Client Certificate 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 connected to a server configured for mutual authentication. The TOE properly responded with a non-empty Client Certificate and Certificate Verify message.

2.1.16 FCS_TLSC_EXT.3 TLS Client Support for Signature Algorithms Extension [TLS Package]

2.1.16.1 TSS Assurance Activities

The evaluator shall verify that TSS describes the signature_algorithm extension and whether the required behavior is performed by default or may be configured.

[ST] Section 6.2 states that the TSF presents SHA-384 in the signature_algorithms extension as the hash parameter used in digital signatures. The evaluated configuration of the TOE is to support appropriate certificates using the RSA+SHA384 configuration string that is applied during initial configuration.

2.1.16.2 Guidance Assurance Activities

If the TSS indicates that the signature_algorithm extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the signature_algorithm extension

[CCECG] Sections 3.2.2.1 and 3.2.2.2 includes instructions for configuring the SHA-384 signature_algorithm extension for each of the TOE platform versions.

2.1.16.3 Test Assurance Activities

The evaluator shall also perform the following tests:

Test 1: The evaluator shall configure the server to send a certificate in the TLS connection that is not supported according to the Client's HashAlgorithm enumeration within the signature_algorithms extension (for example, send a certificate with a SHA-1 signature). The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.

The evaluator presented a certificate with a SHA-1 signature. The unsupported algorithm resulted in a broken connection.

Test 2: [conditional] If the client supports a DHE or ECDHE cipher suite, the evaluator shall configure the server to send a Key Exchange handshake message including a signature not supported according to the client's HashAlgorithm enumeration (for example, the server signed the Key Exchange parameters using a SHA-1 signature). The evaluator shall verify that the product disconnects after receiving the server's Key Exchange handshake message.

The evaluator used the TLS tool to modify the Key Exchange handshake message into one with an unsupported SHA-256 signature. This resulted in a failed connection since the TOE only supports SHA-384.

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

2.1.17.1 TSS Assurance Activities

The evaluator shall verify that TSS describes the Supported Groups Extension

[ST] Section 6.2 states that the TSF presents secp384r1 in the Supported Groups extension as the parameter used for key establishment.

2.1.17.2 Guidance Assurance Activities

None defined.

2.1.17.3 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 successfully connected to the TOE using all supported curves.

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 indicates that the sensitive data is credential data protected by platform storage mechanisms identified in FCS_STO_EXT.1 (platform credential storage repositories). No other data is considered to be sensitive.

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.

As noted in the ST all sensitive data is protected in accordance with FCS_STO_EXT.1. As such this test is not applicable.

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.

[CCECG] Section 3.1.1 identifies the application's access to system resources. The evaluator compared the list to those selected in the [ST] and verified that the two lists are consistent. The list is followed by justification as to why access is required. The justification provided is that the Horizon Client is used to facilitate virtual desktop sessions, which means that a user may be authorized to access anything from individual applications to a full-fledged desktop experience on enterprise resources. Network access is a prerequisite to using the Horizon Client since its purpose is to access external systems. System information is used for logging 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.

[CCECG] Section 3.1.1 identifies the application's access to sensitive information repositories. The evaluator compared the list to those selected in the [ST] and verified that the two lists are consistent. The list is followed by justification as to why access is required. The justification provided is that access to these resources are needed because the Horizon Client is used to facilitate virtual desktop sessions, which means that a user may be authorized to access anything from individual applications to a full-fledged desktop experience on enterprise resources (this includes the clipboard). File system and device access is needed to take full advantage of the resources accessed through the Horizon Client. Network access is a prerequisite to using the Horizon Client since its purpose is to access external systems. System information is used for logging 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.

Windows

The evaluator examined documentation and confirmed it lists required hardware resources.

Android

The evaluator examined the APK installer and confirmed the AndroidManifest.xml file contains all requisite permissions.

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.

Windows

The evaluator studied the documentation and verified it lists sensitive information repositories it accesses.

Android

The evaluator examined the APK installer and verified the AndroidManifest.xml file contains permissions sensitive information repository.

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 ran the application as normal and confirmed all network traffic are documented 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 examined network port scans and confirmed 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 – This condition does not apply.

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 states that the TOE uses X.509 to validate the TLS server certificates of the VMware components that it communicates with (UAG and Horizon Agent). The TOE invokes platform-provided functionality for validating the certificates when establishing trusted communications and includes a list of the functions. These include a description of the certificate validation and certificate path validation in accordance with RFC 5280. The TOE uses X.509 certificates to validate the TLS server certificates of the VMware components that it communicates with (Horizon Connection Server and Horizon Client, both through a connection to the internal network that is initially established through a UAG). Because 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. The client certificate that the TOE presents to a server when performing TLS mutual authentication is user-configured.

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 certificates with the listed modifications and verified the connection failed for each condition.

FIA_X509_EXT.1.1

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

The evaluator presented an expired certificate and verified the connection failed.

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 presented revoked certificates to the TOE and verified the connection failed as a result.

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 presented a certificate that was signed by an invalid signer and confirmed the connection 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 the TLS tool to modify a byte in the first eight bytes of the certificate. The connection was denied.

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 the TLS tool to modify a byte in the last eight bytes of the certificate. The connection was denied.

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 the TLS tool to modify a byte in the certificate's public key. The connection was denied.

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 installed a CA that does not contain the basicConstraints extension and started a connection. 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 installed a CA whose basicConstraints was set to FALSE and started a connection. 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 because 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. The client certificate that the TOE presents to a server when performing TLS mutual authentication is user-configured. For the Windows platform version of the TOE, this is expected to be a certificate stored on a physical smart card and unlocked with a PIN that is associated with the certificate. For the Android platform version of the TOE, the user configures a certificate to be associated with a virtual smart card that is unlocked using a derived credential that is generated from a PIN. The [CCECG] Section 3.2.3 provides instructions for configuring X.509 validity settings and installing CA certificates. Section 3.2 provides instructions for configuring the smart card as the source of the X.509 client certificate uses in mutual authentication.

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 CRL cannot be reached), administrative configuration determines whether the TOE will accept or reject the certificate. This behavior is configured external to the TSF using a GPO or appconfig setting, depending on which platform version of the client is configured.

[CCECG] Section 3.2.3 contains instructions on how to configure the TOE to accept or reject the certificate when the revocation status cannot be determined.

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 disabled the CRL distribution server and started a connection with the TOE. Since the TOE was unable to acquire a new CRL from the disabled server, the TOE was unable to verify the certificate validation and broke the connection as a result.

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 presented a revoked certificate to the TOE. The TOE communicated with CRL distribution server to acquire a CRL list that listed the certificate as revoked. As a result, the connection failed.

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 indicates that the TOE is run locally as an application on the host platform. A user must input a valid credential to gain access to the virtual desktop, but this credential is supplied to the server and is based on an organizational credential (i.e. a user defined in an organization’s Active Directory). The TOE itself is launched in the context of the user session on the host OS platform so no separate authentication is required to access the application itself. The only exception to this is when a user configures a virtual smart card on the Android platform version of the TOE. When this is done, the user is prompted to re-authenticate to the device to ensure the user session is valid. This does not apply to the Windows platform version of the TOE because client certificates are configured by a system administrator on the platform.

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.

The evaluator noted the TOE does not rely on 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.

The evaluator noted the TOE does not rely on 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.

The evaluator noted the TOE does not rely on 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.

Windows

The evaluator ran icacls.exe and observed standard users cannot modify the application or data files.

Android

The evaluator ran the command -L . -perm /002 and did not see any printout.

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: The Windows platform version of the TOE is installed by default to %ProgramFiles%\VMware\VMware Horizon View Client and %ProgramFiles%\Common Files\VMware. These directories are owned by the Administrator account on the host OS platform, who has write access to them. All other users and groups have read-only access. Security-relevant configuration data is stored in the Windows Registry.

The Android platform version of the TOE similarly does not store its binaries or data files with world-writable access, and its configuration data is stored in SharedPreferences (located at /data/data/packages/shared_prefs).

[ST] Section 6.5 provides a list of the Security-relevant configuration data defined during initial setup of the TOE.

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.

Windows

The evaluator ran ProcMon and observed changes in the configuration were saved in the Registry.

Android

The evaluator made changes to the TOE and confirmed the changes were reflected in an appropriate file.

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.

[CCECG] Once the TOE is placed into its evaluated configuration, the full extent to which users can manage the product does not relate to any security-relevant behavior with respect to the claimed PP. This is consistent with the selection in FMT_SMF.1 “no management functions”. The operational guidance contains all necessary information.

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 product includes a number of management functions, but these relate primarily to usability and compatibility functions such as setting the desired Blast encoding/decoding method, virtual desktop resolution, or proxy server settings. With respect to the TSF, the product provides the capability to configure the allowed TLS versions and cipher suites as well as its behavior when a server certificate is invalid, but these settings are not modified once the TOE is placed into its evaluated configuration. Therefore, the TSF does not have any security-relevant management functions with respect to its operational use.

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 indicates that the TOE does not have an interface to request PII from a user and describes cases where PII can only be transmitted over the network if initiated by the user.

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 over a network.

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 is compiled with stack overflow protection through the use of the /GS (Windows) and –fstack-protector (Android) compiler flags.

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.

Windows

The evaluator ran the same application on two different systems and verified they did not share mapping locations. The evaluator also ran BinScope and confirmed that ASLR was enabled.

Android

The evaluator scanned the memory maps of applications and confirmed they did not match.

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.

Windows

The evaluator ran BinScope and found NXcheck passed.

Android

The evaluator analyzed the TOE and found mmap is not invoked with both PROT_WRITE and PROT_EXEC permissions, and mprotect is never invoked.

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 checked that Windows Defender Exploit Guard was enabled and functioned normally with required exploit protections enabled.

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.

Windows

The evaluator ran the TOE through normal usage and verified no executable files were stored in the same directory as user-modifiable files.

Android

The evaluator ran the TOE through normal usage and verified no executable files were found in the data/data directory.

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 confirmed 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] Section 6.7 refers to the list of platform APIs in Appendix A.1 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 listed all APIs listed in TSS and confirmed they were 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 5.2.6.3 of [ST] ("FPT_IDV_EXT.1 Software Identification and Versions") selects "other version information" in FPT_IDV_EXT.1.1 and completes the assignment with "date-based versioning, major/minor release versioning". Section 6.7 of [ST] ("Protection of the TSF") explains the TOE versioning methodology. The TOE uses both 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; 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 verified 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 inspected the installation directory and found only expected libraries as stated in the ST.

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 Windows platform version of the TOE is packaged as an .exe file and the Android platform version of the TOE is packaged as an .apk file. All installation packages are signed by VMware using 2048-bit RSA and the Windows installer is signed with Microsoft Authenticode. Section 6.7 indicates that the updates are obtained as follows. The user of the Windows platform version of the TOE can check for software updates manually within the application. If an update is available, the user is prompted to download and install it. The Android platform version of the TOE relies on the OS platform to check for software updates through the Google Play store. Both platform versions of the TOE identify the application version in the application itself as well as through standard OS reporting mechanisms (e.g. the Windows application version can be identified through Add/Remove Programs).

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.

[TSS Notes] Section 6.7 of [ST] states the TOE is distributed as either an .exe file for the Windows platform or as an .apk file for the Android platform. The TOE is a standalone application that is not natively bundled as part of a host OS.

Section 5.2.6.5 specifies the application is distributed "as an additional software package to the platform OS". Refer to Section 2.6.6 for evaluation activities associated with 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 2.5.2 describes how updates are performed. 2.5.2.1 describes how for Android, updates to the application are made automatically by the platform. Section 2.5.2.2 describes how for Windows, the application can be configured to check for updates to itself at a specified location. The referenced link describes how to download and install the update.

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 2.5.3 includes descriptions of how to query the current version of both the Windows and Android versions of the TOE.

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 checked for an update and verified no error was issued.

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 checked the current version of the software and confirmed it matches that of the documented and installed version.

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).

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 collected a hash of all executable files, ran the TOE as normal, then collected a second set of hashes. These hashes still 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 Windows platform version of the TOE is packaged as an .exe file and the Android platform version of the TOE is packaged as an .apk file. All installation packages are signed by VMware using 2048-bit RSA and the Windows installer is signed with Microsoft Authenticode.

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 examined and confirmed that the Windows installer file was in .exe format, and the Android installer was in APK 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 collected the path of every file on the system prior to installation, then installed the TOE and ran it. Afterwards, the evaluator uninstalled the TOE and recollected the path of every file. No changes apart from audit log files and configuration were done.

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 6.8 provides details on how the TOE relies on its own mechanisms to secure all data in transit between itself and its operational environment using HTTPS and/or TLS. This is consistent with Section 5.2.7.1 of [ST] (“FTP_DIT_EXT.1 Protection of Data in Transit”) that specifies the application shall encrypt all transmitted data with HTTPS as client and TLS. As such, the TOE does not invoke platform-provided functionality for protection of data in transit.

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 examined a PCAP of typical TOE uses and confirmed 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 examined a PCAP of typical TOE use and confirmed 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 examined a PCAP of typical TOE use and confirmed no credentials were transmitted in plaintext.

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 – This does not apply to the TOE.

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 2.5.3 describes the process for verifying software updates using digital signature which is performed by the TOE. A failed integrity check will prevent installation of the application and the version of the application can be checked to if the update was successful or not. Section 2.5.2.1 states that the updates to the Android application are made automatically by the platform through the Play store. No user interaction is necessary. For the Windows version, Section 2.5.2.2 provides a reference link to the Horizon Client for Windows Guide for instructions on configuring the update process. This referenced section, "Update Horizon Client Online" also describes how to obtain and initiate an update and indicates that the interactive installation wizard assists with the installation after the update has been downloaded.

Section 2.6.61 above also covers the process for verifying updates to the TOE by verifying a digital signature.

[CCECG] Section 1.3 identifies features and functions not included in the TOE evaluation. Section 3.1.2 provides instructions for configuring the cryptographic engine and indicates no other engine or configuration was used in the evaluation.

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.

The [ST] claims Android and Windows platforms. The evaluator checked and ensured that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

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 Client 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.

[CCECG] identifies Windows and Android environments. Section 6.7 of [ST] ("Protection of the TSF") describes how each TOE version uses security features and APIs provided by its platform. This includes data execution protection, stack-based buffer overflow protection, and compatibility with platform security features.

As described in Section 3.4.1 above, the evaluator confirmed the TOE and guidance documentation are labelled with unique software version identifiers.

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 for the Windows Client from https://customerconnect.vmware.com, while the Android Client uses the Google Play store. 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. 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 cumlative 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

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.html.

The evaluation team performed searches on 4/20/2023 and again on 6/7/2023, using the following search terms:

- VMware Horizon
- Horizon Client
- VMware's OpenSSL FIPS Object Module 2.0.20-vmw
- Centralized content server
- Enterprise resource delivery
- Enterprise content delivery
- OpenSSL 1.0.2zg (third party library)
- 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 and Linux 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.