Assurance Activities Report

for

Acronis SCS Cyber Backup 12.5 Hardened Edition Agent

Version 1.2

22 November 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:

Acronis
1225 W. Washington St, Suite #205,
Tempe, AZ 85288
United States of America

The TOE Evaluation was Sponsored by:

Acronis
1225 W. Washington St, Suite #205,
Tempe, AZ 85288
United States of America

Evaluation Personnel:

Allen Sant

Contents

1 Introduction

This document presents results from performing evaluation activities associated with the evaluation of Acronis SCS Cyber Backup Management Agent Version 12.5. This report contains sections documenting the performance of evaluation activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in the evaluation activities for the Protection Profile for Application Software, Version 1.4, 07 October 2021 [App PP] and the Functional Package for Transport Layer Security (TLS), Version 1.1, 01 March 2019 [TLS Package].

1.1 Applicable Technical Decisions

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

App PP

TD0780	FIA_X509_EXT.1 Test 4 Clarification This TD applies to the TOE. The TD modifies Test activities and has been incorporated into the AAR.
TD0756	Update for platform-provide full disk encryption This TD applies to the TOE. The TD modifies Test activities and has been incorporated into the AAR.
TD0747	Configuration Storage Option for Android
TD0743	FPT_DIT_EXT.1.1 Selection exclusivity This TD applies to the TOE but only modifies the SFR, so no EAs are affected.
TD0736	Number of elements for iterations of FCS_HTTPS_EXT.1 This TD does not apply to the TOE as it affects FCS_HTTPS_EXT.1/Server which is not claimed by this TOE.
TD0719	ECD for PP APP v1.3 and 1.4 This TD applies to the TOE. The TD provides formal definitions for the extended components.
TD0717	Format changes for PP_APP_V1.4 This TD applies to the TOE. The TD ensures consistency with CC Part 2 for FCS family SFRs.
TD0664	Testing activity for FPT_TUD_EXT.2.2 This TD applies to the TOE. The TD modifies Test activities and has been incorporated into the AAR.
TD0650	Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4

This TD does not apply to the TOE as it provides the ability to utilize the PP-Module for VPN clients which this TOE does not claim.

TD0628 Addition of Container Image to Package Format

This TD applies to the TOE. The TD modifies the SFR and Test and has been incorporated into the AAR.

TLS Package

TD0770 TLSS.2 connection with no client cert

This TD does not apply to the TOE as the TOE does not claim the relevant SFR FCS_TLSS_EXT.2.2

TD0739 PKG_TLS_v1.1 has 2 different publication dates

This TD applies to the TOE. The TD modifies a Test and has been incorporated into the AAR.

TD0726 Corrections to (D)TLSS SFRs in TLS 1.1 FP

This TD does not apply to the TOE as the TOE does not claim the relevant SFR FCS_TLSS_EXT.1.3

TD0588 Session Resumption Support in TLS Package

This TD is not applicable to the ST or TOE because the TOE does not claim FCS_TLSS_EXT.1

TD0513 CA Certificate loading

This TD applies to the TOE. The TD modifies a Test and has been incorporated into the AAR.

TD0499 Testing with pinned certificates

This TD does not apply to the TOE the TOE does not utilize pinned certificates.

TD0469 Modification of Test Evaluation Activity for FCS_TLSS_EXT.1.1 test 4.1

This TD does not apply to the TOE as the TOE does not claim the SFR FCS_TLSS_EXT.1

TD0442 Updated TLS Ciphersuites for TLS Package

This TD applies to the TOE but only modifies the SFR, so no EAs are affected.

1.2 Evidence

[App PP] *Protection Profile for Application Software*, Version 1.4, 2021-10-07

[TLS Package] *Functional Package for Transport Layer Security (TLS)*, Version 1.1, 2019-03-01

[ST] *Acronis SCS Cyber Backup 12.5 Hardened Edition Agent v12.5 Security Target*, Document Version 6.0, October 11, 2023

[USER] *Acronis SCS Acronis SCS Backup Agent User Guide*

[CCSUPP] *Acronis SCS Acronis Cyber Backup 12.5 Hardened Edition Agent Guidance Documentation Supplement*, Document Version: 0.7, October 11, 2023

[TEST] *Acronis SCS Cyber Backup 12.5 Hardened Edition Agent Common Criteria Test Report and Procedures*, Version 1.1, October 11, 2023

[AVA] *Acronis SCS Cyber Backup 12.5 Hardened Edition Agent v12.5 Vulnerability Assessment*, Version 1.3, November 22, 2023

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.

1.4 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.1	Pass
ASE_REQ.1	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
ATE_IND.1	Pass
AVA_VAN.1	Pass

The evaluation work units are listed in the proprietary ETR. The evaluators note per the PP evaluation activities that many of the SARs were successfully evaluated through completion of the associated evaluation activities presented in the claimed PP.

2 Security Functional Requirement Assurance Activities

This section describes the evaluation activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The evaluation activities are derived from the [App PP] and [TLS Package] and modified by applicable NIAP Technical Decisions. Evaluation 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 evaluation activities to identify when they are mapped to individual SFR elements rather than the component level.

Note that the TOE includes both Windows and Linux platform versions. In general, [ST] only distinguishes between these platform versions when an SFR is implemented in a different manner based on the platform. For example, FCS_STO_EXT.1 and FPT_AEX_EXT.1 are implemented differently based on the OS platform so [ST] describes how each platform version of the TOE meets these SFRs. This is in contrast with e.g. the FCS_COP.1 iterations, which claim the same algorithms for both platform versions of the TOE and therefore no distinction is made in [ST] for the claimed functionality.

2.1 Cryptographic Support (FCS)

2.1.1 FCS_CKM_EXT.1 Cryptographic Key Generation Services

Title modified by TD0717.

2.1.1.1 TSS Evaluation 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.

[ST] Section 8.1.1 FCS_CKM_EXT.1 and FCS_CKM.1/AK: states that the TOE implements asymmetric key generation using RSA, and ECC key generation schemes for key establishment and entity authentication for TLS. This is consistent with the selection of "implement asymmetric key generation" and with the inclusion of FCS_CKM.1/AK in the ST.

2.1.1.2 Guidance Evaluation Activity

None.

2.1.1.3 Test Evaluation Activity

None.

2.1.2 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation

2.1.2.1 TSS Evaluation 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 8.1.1 FCS_CKM_EXT.1 and FCS_CKM.1/AK: states that the TOE implements asymmetric key generation and Table 14 of the ST identifies those key sizes as 2048, and 3072 for RSA, and NIST curves with sizes 256, 384, 521 for ECC (ECDSA). The description indicates that RSA and ECC key generation schemes are used for key establishment and entity authentication for TLS and HTTPS.

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.

The ST does not claim “invokes platform-provided functionality for asymmetric key generation”.

2.1.2.2 Guidance Evaluation 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.

[CCSUPP] Section 3.1.1 states that there are no management options to change the settings for the implemented cryptographic library. The TOE is already configured with the key generation scheme(s) and key size(s) to meet the security requirements outlined in the Security Target.

2.1.2.3 Test Evaluation Activity

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 FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:
   - Provable primes
   - Probable primes
2. Primes with Conditions:
   - Primes p1, p2, q1,q2, p and q shall all be provable primes
   - Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
   - Primes p1, p2, q1,q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public

exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- $n = p \cdot q$,
- p and q are probably prime according to Miller-Rabin tests,
- GCD(p-1,e) = 1,
- GCD(q-1,e) = 1,
- $2^{16} \leq e \leq 2^{256}$ and e is an odd integer,
- $|p-q| > 2^{nlen/2 - 100}$,
- $p \geq 2^{nlen/2 -1/2}$,
- $q \geq 2^{nlen/2 -1/2}$,
- $2^{(nlen/2)} < d < LCM(p-1,q-1)$,
- $e \cdot d = 1 \bmod LCM(p-1,q-1)$.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

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

Key Generation for Finite-Field Cryptography (FFC)

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

Cryptographic and Field Primes:

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

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

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

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

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

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

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

for each FFC parameter set and key pair.

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

The evaluator examined the ST and found that in Section 8.1.1 "Cryptographic Support" Table 14 "Cryptographic Algorithms and Key Sizes" that the TOE was awarded the CAVP certificates #C1351 and #A4299 for FIPS186-4 RSA and ECDSA key generation using the key sizes 2048, 3072 and P256, P384, P521. This certificate provides assurance that the TSF performs these functions as required.

2.1.3 FCS_CKM.2 Cryptographic Key Establishment

2.1.3.1 TSS Evaluation Activity

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

[ST] Section 8.1.1 states that the TOE implements both RSA and elliptic curve-based key establishment schemes for TLS. The key establishment schemes correspond to the schemes identified in FCS_CKM.1.1/AK.

2.1.3.2 Guidance Evaluation 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).

[CCSUPP] Section 3.1.1 states that there are no management options to change the settings for the implemented cryptographic library. The TOE is already configured with the key generation scheme(s) and key size(s) to meet the security requirements outlined in the Security Target.

2.1.3.3 Test Evaluation Activity

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.

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following evaluation activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

> To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following evaluation activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

> To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS-OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For

each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses RSAES-PKCS1-v1_5.

Diffie-Hellman Group 14

The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses Diffie-Hellman group 14.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

The correctness of the TSF's implementation of RSAESPKCS1-v1_5 is verified by its implementation of TLSv1.2 as a TLS Client. The evaluator examined the ST and found that in Section 8.1.1 "Cryptographic Support" Table 14 "Cryptographic Algorithms and Key Sizes" that the TOE was awarded the CAVP certificates #C1351 and #A4299 for Elliptic curve based key establishment (NIST SP800-131A) using P256, P384 and P521. This certificate provides assurance that the TSF performs these functions as required.

2.1.4 FCS_COP.1/Hash Cryptographic Operation – Hashing

2.1.4.1 TSS Evaluation 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.

The evaluator examined the TSS to check that the association of hash functions with other application cryptographic functions was properly documented. Section 8.1 of the ST was used to determine the verdict of this work unit. The evaluator found that hash functions are used with other TOE cryptographic functions, including digital signature verification and Message Authentication Code to support the TOE's TLS cryptographic functionality.

2.1.4.2 Guidance Evaluation Activity

None.

2.1.4.3 Test Evaluation Activity

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

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

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

Test 2: Short Messages Test - Byte oriented Mode. The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 3: Selected Long Messages Test - Bit oriented Mode. The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Test 4: Selected Long Messages Test - Byte oriented Mode. The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

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

Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

The evaluator examined the ST and found that in Section 8.1.1 "Cryptographic Support" Table 14 "Cryptographic Algorithms and Key Sizes" that the TOE was awarded the CAVP certificates #C1351 and #A4299 for SHA2-256 (FIPS180-4) and SHA2-384 (FIPS180-4). This certificate provides assurance that the TSF performs these functions as required.

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

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

2.1.5.1 TSS Evaluation Activity

None.

2.1.5.2 Guidance Evaluation Activity

None.

2.1.5.3 Test Evaluation 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 evaluator examined the ST and found that in Section 8.1.1 "Cryptographic Support" Table 14 "Cryptographic Algorithms and Key Sizes" that the TOE was awarded the CAVP certificates #C1351 and #A4299 for HMAC-SHA2-256 (FIPS198) and HMAC-SHA2-384 (FIPS198). This certificate provides assurance that the TSF performs these functions as required.

2.1.6 FCS_COP.1/Sig Cryptographic Operation – Signing

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

2.1.6.1 TSS Evaluation Activity

None.

2.1.6.2 Guidance Evaluation Activity

None.

2.1.6.3 Test Evaluation Activity

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.

ECDSA Algorithm Tests

Test 1: ECDSA FIPS 186-4 Signature Generation Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

Test 2: ECDSA FIPS 186-4 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

This is not applicable as the TOE does not use or claim ECDSA algorithm support.

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 evaluator examined the ST and found that in Section 8.1.1 "Cryptographic Support" Table 14 "Cryptographic Algorithms and Key Sizes" that the TOE was awarded the CAVP certificates #C1351 and #A4299 for RSA Digital signature generation and verification (FIPS186-4) using 2048 and 3072 bit RSA keys. This certificate provides assurance that the TSF performs these functions as required.

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

2.1.7.1 TSS Evaluation Activity

None.

2.1.7.2 Guidance Evaluation 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.

[CCSUPP] Section 3.1.1 states that there are no management options to change the settings for the implemented cryptographic library. The TOE is already configured with the key generation scheme(s) and key size(s) to meet the security requirements outlined in the Security Target.

2.1.7.3 Test Evaluation Activity

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

AES-CBC Known Answer Tests

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

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

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

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

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

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

This Is not applicable as the TOE does not claim or use AES in CBC mode.

AES-CBC Multi-Block Message Test

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

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

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

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

This Is not applicable as the TOE does not claim or use AES in CBC mode.

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 non-zero 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 evaluator examined the ST and found that in Section 8.1.1 "Cryptographic Support" Table 14 "Cryptographic Algorithms and Key Sizes" that the TOE was awarded the CAVP certificates #C1351 and #A4299 for AES-GCM (NIST SP800-38D) using the key sizes 128 and 256 for Encryption and Decryption.

AES-XTS Tests

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

256 bit (for AES-128) and 512 bit (for AES-256) keys

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

Using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

This Is not applicable as the TOE does not claim or use AES in XTS mode.

AES-CCM Tests

It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

- Keys: All supported and selected key sizes (e.g., 128, 256 bits).
- Associated Data: Two or three values for associated data length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.
- Payload: Two values for payload length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported payload lengths.
- Nonces: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13) in bytes.
- Tag: All supported tag lengths (4, 6, 8, 10, 12, 14, 16) in bytes.

The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

Variable Associated Data Test

For each supported key size and associated data length, and any supported payload length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Payload Test

For each supported key size and payload length, and any supported associated data length, nonce length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Nonce Test For each supported key size and nonce length, and any supported associated data length, payload length, and tag length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Variable Tag Test For each supported key size and tag length, and any supported associated data length, payload length, and nonce length, the evaluator shall supply one key value, one nonce value, and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Decryption-Verification Process Test To test the decryption-verification functionality of AES-CCM, for each combination of supported associated data length, payload length, nonce length, and tag length, the evaluator shall supply a key value and 15 sets of input plus ciphertext, and obtain the decrypted payload. Ten of the 15 input sets supplied should fail verification and five should pass.

This is not applicable as the TOE does not claim or use AES in CCM mode.

AES-CTR Tests

Test 1: Known Answer Tests (KATs)

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

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

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

256-bit keys. To test the decrypt functionality, the evaluator shall perform the same test as for encrypt, using an all zero ciphertext value as input.

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

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

Test 2: Multi-Block Message Test

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

Test 3: Monte-Carlo Test

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

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

For AES-ECB mode # Input: PT, Key for i = 1 to 1000: CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]

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

This is not applicable as the TOE does not claim to use AES in CTR mode.

2.1.8 FCS_RBG_EXT.1 Random Bit Generation Services

2.1.8.1 TSS Evaluation Activity

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.

Section 7.2.1 of the ST indicates that the ST does not select “use no DRBG functionality”.

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

The ST selects “implement DRBG functionality”, therefore, the evaluator verified that FCS_RBG_EXT.2 is included in the ST. Sections 7.2.1 and 8.1.1 of the ST were used to determine the verdict of this evaluation activity. The evaluator found that the required elements are present 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.

Section 7.2.1 of the ST indicates that the ST does not select “invoke platform-provided DRBG functionality.”

2.1.8.2 Guidance Evaluation Activity

None.

2.1.8.3 Test Evaluation 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).

This is not applicable as the TOE does not select "invoke platform-provided DRBG functionality" and instead implements DRBG functionality.

2.1.9 FCS_RBG_EXT.2 Random Bit Generation from Application

2.1.9.1 TSS Evaluation 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 examined the documentation and determined that it includes the required detailed sections necessary to thoroughly understand the entropy sources and to determine that it can be relied upon to provide sufficient entropy.

2.1.9.2 Guidance Evaluation Activity

FCS_RBG_EXT.2.1 and FCS_RBG_EXT.2.2

None.

2.1.9.3 Test Evaluation Activity

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

Implementations Conforming to FIPS 140-2 Annex C.

FCS_RBG_EXT.2.1

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.

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.

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.

This is not applicable as the TOE does not claim conformance to FIPS 140-2 Annex C and claims conformance to NIST SP800-90A.

Implementations Conforming to NIST Special Publication 800-90A

FCS_RBG_EXT.2.1

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.

The evaluator examined the ST and found that in Section 8.1.1 “Cryptographic Support” Table 14 “Cryptographic Algorithms and Key Sizes” that the TOE was awarded the CAVP certificates #C1351 and #A4299 for DRBG (NISP SP800-90A).

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.

2.1.10 FCS_STO_EXT.1 Storage of Credentials

2.1.10.1 TSS Evaluation 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 8.1.2 FCS_STO_EXT.1: The TOE leverages the Windows Data Protection API (DPAPI) to securely store the TOE’s application token for the Windows Agent. On Linux, the application token is securely stored using the Linux key ring. The application token is used by the TOE to identify itself to the Management Server when downloading configuration settings. The initial application token is generated by the Management Server when the TOE is installed and added as a device to the Management Server.

2.1.10.2 Guidance Evaluation Activity

None.

2.1.10.3 Test Evaluation Activity

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.1/PBKDF. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

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

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

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

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

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

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

The evaluator verified that the RHEL client utilized Linux keyrings and that the Windows client utilized the Windows Data Protection API (DPAPI) to store credentials.

2.1.11 FCS_TLS_EXT.1 TLS Protocol [TLS Package]

2.1.11.1 General Evaluation Activity

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

The evaluator verified that the selection in the ST of 'TLS as a client' is consistent with the dependent components. The selection of TLS as a client is consistent with the selection of TLS as a client in FTP_DIT_EXT.1 and the inclusion of the FCS_TLSC_EXT.1 SFR. The TOE does not support mutual authentication which is consistent with the absence of FCS_TLSC_EXT.2.

2.1.12 FCS_TLSC_EXT.1 TLS Client Protocol [TLS Package]

2.1.12.1 TSS Evaluation Activity

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 8.1.1 FCS_TLS_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.4, and FCS_TLSC_EXT.5: identifies the ciphersuites supported by the TOE as:

- TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289, and

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289.

These cipher suites specified are identical to those listed for the FCS_TLSC_EXT.1 component.

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

[ST] Section 8.1.1 FCS_TLS_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.4, and FCS_TLSC_EXT.5: states that the TOE looks for the common name in the subject name or the DNS name in the subject alternative name (SAN) of the server's certificate as the identifier for the Management Server. The reference identifier is established during installation when the Management Server's name is entered in the connection information. Use of IP addresses and wildcards as the identifiers are supported but are discouraged as identifiers. When constructing the certificate, the SAN is mandated for IP identifiers and not mandated for DNS identifiers. The use of certificate pinning is not supported.

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

[ST] Section 8.1.1 FCS_TLS_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.4, and FCS_TLSC_EXT.5: Without exception, the TOE will not establish a connection if the server's certificate is not valid.

2.1.12.2 Guidance Evaluation Activity

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. 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.

[CCSUPP] the Guidance document indicates that the reference identifier is set during the installation process, the Server Name/IP address provided for the Management Server during installation is the value used for the reference identifier. Specifically in step 8 of 2.2.3 (for the Windows Agent) and in step 7 of 2.2.5 (for the Linux Agent).

2.1.12.3 Test Evaluation Activity

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 verified that the TOE, on both the Windows and RHEL platforms, could successfully establish a TLS connection using each of the claimed cipher suites.

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 verified that the TOE, on both the Windows and RHEL, platforms, successfully completed connections when the server presented a certificate with the Server Authentication EKU and rejected connections where the server presented a certificate which only possessed the Client Authentication EKU.

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 verified that if a server presented a certificate type that does not match the type required by the selected cipher suite that the TOE, on both the Windows and RHEL platforms, rejects the connection.

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 verified that if a server presented the ciphersuite TLS_NULL_WITH_NULL_NULL that the TOE, on both the Windows and RHEL platforms, would reject the connection attempt.

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

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 verified that if a server selected an undefined TLS version (1.5) that the TOE, on both the Windows and RHEL platforms, would reject the connection attempt.

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 verified that if the server attempted to downgrade the TLS protocol version to the most recent unsupported version that the TOE, on both the Windows and RHEL platforms, would reject the connection attempt.

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 verified that if the server modified ServerHello.Random value after sending the ServerHello but before building the ServerKeyExchange record that the TOE, on both the Windows and RHEL platforms, rejects the connection attempt.

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 verified that if the server selected a ciphersuite that was not offered by the TOE that the TOE, on both the Windows and RHEL platforms, would reject the connection attempt.

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 verified that if the server sent the incorrect signature on the ServerKeyExchange record that the TOE, on both the Windows and RHEL platforms, would reject the connection attempt.

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 verified that if the server did not calculate the Server Finished verify data correctly that the TOE, on both the Windows and RHEL platforms, would reject the connection attempt and no application data flowed.

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 verified that if the server does not send the Finished record immediately after the Change Cipher Spec and instead sends random data instead, that the TOE, on both the Windows and RHEL platforms, rejects the connection attempt and no application data flows.

FCS_TLSC_EXT.1.2

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

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 verified that if the server presents a certificate which contains an CN that does not match the reference identifier and no SAN extensions that the TOE, on both the Windows and RHEL platforms, rejects the connection. This was tested using FQDN and IP address identifiers.

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 verified that if the server presents a certificate which contains a CN that matches the reference identifier and a SAN that does not match the reference identifier that the TOE, on both the Windows and RHEL platforms, rejects the connection. This was tested using FQDN and IP address identifiers.

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 verified that if the server presents a certificate which contains a CN that matches the reference identifier and no SAN extensions that the TOE, on both the Windows and RHEL platforms, accepts the connection. This was tested using FQDN and IP address identifiers.

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 verified that if the server presents a certificate that contains a CN that does not match the reference identifier and contains a SAN which matches the reference identifier that the TOE, on both the Windows and RHEL platforms, accepts the connection. This was tested using FQDN and IP address identifiers.

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.

The evaluator performed the tests prescribed and verified that the defined tests passed.

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 verified that if the server presents a certificate containing a wildcard that is not in the left-most field of the presented identifier that the TOE, on both the Windows and RHEL platforms, rejects the connection attempt.

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.com) and verify that the connection fails.

The evaluator verified that if the server presents a certificate containing a wildcard that is in the leftmost label that the TOE, on both the Windows and RHEL platforms, behaved in the expected manner for each case. The TOE accepts and completes the connection when a single leftmost identifier is used in the reference identifier. The TOE rejects the connection when there is no leftmost identifier used in the reference identifier. The TOE rejects the connection when there are two leftmost identifiers used in the reference identifier.

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 verified that if the server presents a certificate containing a wildcard in the domain section of the FQDN (immediately preceding the public suffix) the TOE, on both the Windows and RHEL platforms, rejected the connection attempt when using a reference identifier with a leftmost label (e.g. bar.foo.com) and without a leftmost label (e.g. foo.com).

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.

This is not applicable as the TOE does support wildcards.

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

This is not applicable as the TOE does not support URI or Service name reference identifiers.

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.

This is not applicable as the TOE does not support certificate pinning.

FCS_TLSC_EXT.1.3

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

Modified by TD0513

~~**Test 1**: The evaluator shall demonstrate that a server using a certificate without a valid certification path results in an authentication failure. Using the administrative guidance, the evaluator shall then load the trusted CA certificate(s) needed to validate the server's certificate, and demonstrate that the connection succeeds. The evaluator then shall delete one of the CA certificates, and show that the connection fails.~~

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

The evaluator verified that the TOE, on both the Windows and RHEL platforms, accepts connections attempts when the presented certificate chain can be completed and anchored to a CA certificate in the platform trust store successfully.

Added by TD0513

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 verified that the TOE, on both the Windows and RHEL platforms, rejects connection attempts when the presented certificate chain cannot be completed and anchored to a CA certificate in the platform trust store successfully.

Added by TD0513
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.

This is not applicable as the TOE does not maintain a trust store that can be managed.

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

The evaluator verified that the TOE, on both the Windows and RHEL platforms, rejects connection attempts when the presented certificate contains revocation status information and has a status of revoked.

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 verified that if the server presents a certificate that is past its expiration date the TOE, on both the Windows and RHEL platforms, rejects the connection attempt.

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 verified that if the server presents a certificate which does not have a valid identifier the TOE, on both the Windows and RHEL platforms, rejects the connection. This is tested in FCS_TLSC_EXT.1.2 Tests 1-4.

2.1.13 FCS_TLSC_EXT.4 TLS Client Support for Renegotiation

2.1.13.1 TSS Evaluation Activity

No evaluation activities defined.

2.1.13.2 Guidance Evaluation Activity

No evaluation activities defined.

2.1.13.3 Test Evaluation Activity

The evaluator shall perform the following tests:

Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the “renegotiation_info” field or the SCSV cipher suite is included in the ClientHello message during the initial handshake.

The evaluator verified that the TOE, on both the Windows and RHEL platforms, presented the SCSV cipher suite (TLS_EMPTY_RENEGOTIATION_INFO_SVSC) in the ClientHello cipher suites list.

Test 2: The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the "renegotiation_info" extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.

The evaluator verified that the TOE, on both the Windows and RHEL platforms, rejected connections when the ServerHello contained a non-zero length "renegotiation_info" extension.

The evaluator verified that the TOE, on both the Windows and RHEL platforms, accepted connections when the ServerHello contains a valid "renegotiation_info" extension.

Test 3: The evaluator shall verify that ServerHello messages received during secure renegotiation contain the "renegotiation_info" extension. The evaluator shall modify either the "client_verify_data" or server_verify_data" value and verify that the client terminates the connection.

The evaluator verified that the TOE, on both the Windows and RHEL platforms, properly validated the info field of the "renegotiation_info" extension during secure renegotiation and terminated connections when the value provided is not correct.

2.1.14 FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension

2.1.14.1 TSS Evaluation Activity

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

[ST] Section 8.1.1 FCS_TLS_EXT.1, FCS_TLSC_EXT.1, FCS_TLSC_EXT.4, and FCS_TLSC_EXT.5: states that the TOE presents the Supported Groups Extension in the ClientHello with NIST curves secp256r1, secp384r1, and secp521r1.

2.1.14.2 Guidance Evaluation Activity

No evaluation activities defined.

2.1.14.3 Test Evaluation Activity

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 verified that the TOE, on both the Windows and RHEL platforms, can successfully establish a connection to a TLS server using each of the specified curves.

2.2 User Data Protection (FDP)

2.2.1 FDP_DAR_EXT.1 Encryption of Sensitive Application Data

2.2.1.1 TSS Evaluation 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 8.1.2: The TOE protects sensitive data in accordance with FCS_STO_EXT.1 when it is stored in non-volatile memory. The application token used to identify the TOE to the Management Server is the only sensitive data that the TOE stores. No other forms of sensitive data are stored by the TOE. The TOE runs as a service in the evaluated configuration and does not require any user credentials to operate.

2.2.1.2 Guidance Evaluation Activity

None.

2.2.1.3 Test Evaluation Activity

Modified by TD0756

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.

If "implement functionality to encrypt sensitive data as defined in the PP-Module for File Encryption" or "protect sensitive data in accordance with FCS_STO_EXT.1" is selected, 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.

This is not applicable as the TOE only uses methods covered by FCS_STO_EXT.1 to store sensitive data.

2.2.2 FDP_DEC_EXT.1 Access to Platform Resources

2.2.2.1 TSS Evaluation Activity

FDP_DEC_EXT.1.1 and FDP_DEC_EXT.1.2

None.

2.2.2.2 Guidance Evaluation Activity

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.

[CCSUPP] Section 3.1.4 states that the TOE leverages the platform's networking hardware to communicate with other systems in the environment. The Windows Firewall is configured automatically during installation to allow the TOE to communicate with the systems in the environment over HTTPS and TLS.

[CCSUPP] Section 3.1.4.1: The backup agent (TOE) establishes secure communications with the Management Server using TLSv1.2 over TCP ports 7770-7800 after validating the X.509 certificate received from the Management Server.

FDP_DEC_EXT.1.2

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

The [CCSUPP] and [USER] do no identify any instances of the application accessing to sensitive repositories and this is consistent with the [ST] Sections 7.2.2 and 8.1.2 that state that the TOE does not access any of the sensitive information repositories on the host platform.

2.2.2.3 Test Evaluation Activity

FDP_DEC_EXT.1.1

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

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

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

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

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

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

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

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

The evaluator verified that the guidance documentation (section 3.1.4 of [CCSUPP]) indicates the hardware capabilities that are required by each platform version of the TOE.

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

For 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 the sensitive information repositories it accesses.

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

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

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

The ST claims no sensitive information repositories are relied upon by the TOE so there are no repositories listed in the guidance documentation in this context.

2.2.3 FDP_NET_EXT.1 Network Communications

2.2.3.1 TSS Evaluation Activity

None.

2.2.3.2 Guidance Evaluation Activity

None.

2.2.3.3 Test Evaluation Activity

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 observed the traffic between the Agent, on both the Windows and RHEL platforms, and the Management Server. The evaluator verified that the Agent did not send any traffic that is not documented in the TSS or is user-initiated. Specifically, the outbound traffic observed for the TOE was bound for the Management Server on ports 9877 and 7780, consistent with the claims made in the TSS.

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 scanned the system running the TOE, on both the Windows and RHEL platforms, with the tool NMAP and verified that there were no undocumented ports opened by the TOE in comparison against those opened by the OS itself.

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 uses-permission-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.

This portion is not applicable as the Android platform is not claimed.

2.3 Identification and Authentication (FIA)

2.3.1 FIA_X509_EXT.1 X.509 Certificate Validation

2.3.1.1 TSS Evaluation 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 8.1.3 FIA_X509_EXT.1 and FIA_X509_EXT.2: describes the certificate validation process that takes place when server certificates are presented to the TOE. Specifically, the TOE validates certificates and certificate paths in accordance with RFC 5280. All certificate paths must terminate with a trusted CA certificate. Validation of the basicConstraints extension and CA flag are performed for all CA certificates. Certificate revocation checking is performed using CRLs. The extendedKeyUsage field is checked for appropriate purpose.

FIA_X509_EXT.1.2
None.

2.3.1.2 Guidance Evaluation Activity

FIA_X509_EXT.1.1, FIA_X509_EXT.1.2
None.

2.3.1.3 Test Evaluation Activity

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 selfsigned Root CA. If the application supports a maximum trust depth of two, then a chain with no Intermediate CA should instead be created.

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 verified that the TOE, on both the Windows and RHEL platforms, rejects certificates which chain to CA certificates which are not valid CA certificates in the specific cases of; the basicConstraints value either not set or set to false, the Certificate Signing KeyUsage bit is not set, the Path Length is not valid for the presented path length. The evaluator verified that the TOE

accepted certificates when the full certificate chain could be validated. The evaluator verified that the TOE rejected certificates when the full certificate chain could not be completed.

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

The evaluator verified that the TOE, on both the Windows and RHEL platforms, rejects certificates which are expired.

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 verified that the TOE, on both the Windows and RHEL platforms, checked the revocation status of a presented certificates and certificate chains. The evaluator verified that the TOE rejected certificates which have a status of revoked.

Modified by TD0780

Test 4: If any OCSP option is selected, the evaluator shall ensure the TSF has no other source of revocation information available and configure the OCSP server or use a man-in-the-middle tool to present an OCSP response signed by a certificate that does not have the OCSP signing purpose and which is the only source of revocation status information advertised by the CA issuing the certificate being validated. The evaluator shall verify that validation of the OCSP response fails and that the TOE treats the certificate being checked as invalid and rejects the connection. If CRL is selected, the evaluator shall likewise configure the CA to be the only source of revocation status information, and sign a CRL with a certificate that does not have the cRLsign key usage bit set~~, and~~ . The evaluator shall verify that validation of the CRL fails and that the TOE treats the certificate being checked as invalid and rejects the connection.

Note: The intent of this test is to ensure a TSF does not trust invalid revocation status information. A TSF receiving invalid revocation status information from the only advertised certificate status provider should treat the certificate whose status is being checked as invalid. This should generally be treated differently from the case where the TSF is not able to establish a connection to check revocation status information, but it is acceptable that the TSF ignore any invalid information and attempt to find another source of revocation status (another advertised

provider, a locally configured provider, or cached information) and treat this situation as not having a connection to a valid certificate status provider.

The evaluator verified that the TOE, on both the Windows and RHEL platforms, rejected CRL updates that are not signed by a valid CA, specifically updates which are signed by a certificate lacking the CRLsign Key Usage. The evaluator verified that the TOE rejected certificates in this scenario.

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 verified that the TOE, on both the Windows and RHEL platforms, rejected certificates that fail to parse correctly (the first byte is not correct).

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 verified that the TOE, on both the Windows and RHEL platforms, rejected certificates that fail to validate correctly, the signature is invalid when the last byte of the presented certificate has been modified.

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 verified that the TOE, on both the Windows and RHEL platforms, rejected certificates that fail to validate correctly, the public key on the presented certificate has been modified.

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.

This is not applicable as the TOE does not support EC certificates according to FCS_COP.1/Sig.

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.

This is not applicable as the TOE does not support EC certificates according to FCS_COP.1/Sig.

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

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 verified that the TOE, on both the Windows and RHEL platforms, rejected Certificates which chain to a CA that does not contain the basicConstraints extension value.

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 verified that the TOE, on both the Windows and RHEL platforms, rejected Certificates which chain to a CA that has the basicConstraints extension set to FALSE.

2.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication

2.3.2.1 TSS Evaluation 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.

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 8.1.3 FIA_X509_EXT.1 and FIA_X509_EXT.2: Indicates that the TOE chooses what certificates to use based on what is presented to it as part of establishing a TLS session. Since the TOE only validates server certificates that it receives and does not present any certificates to identify itself to others there is no need to select a specific certificate from a repository.

[ST] Section 8.1.3 FIA_X509_EXT.1 and FIA_X509_EXT.2: States that if the TOE cannot establish a connection to the CA server’s CRL to determine the revocation status of a certificate, it does not accept the certificate.

2.3.2.2 Guidance Evaluation Activity

None.

2.3.2.3 Test Evaluation Activity

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 verified that if the TOE, on both the Windows and RHEL platforms, cannot access the revocation provider that the TOE rejects the certificate.

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 non-TOE IT entity cannot be accepted.

The evaluator verified that the TOE, on both the Windows and RHEL platforms, did not accept an invalid certificate which required validation with a non-TOE IT entity.

2.4 Security Management (FMT)

2.4.1 FMT_CFG_EXT.1 Secure by Default Configuration

2.4.1.1 TSS Evaluation 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 8.1.4 FMT_CFG_EXT.1: The TOE does not install with any default credentials. Rather, it uses the platform's service accounts to run and is available to any user logged into the platform.

FMT_CFG_EXT.1.2
None.

2.4.1.2 Guidance Evaluation Activity

FMT_CFG_EXT.1.1 and FMT_CFG_EXT.1.2
None.

2.4.1.3 Test Evaluation Activity

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.

This is not applicable as the TOE does not utilize 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.

This is not applicable as the TOE does not utilize 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.

This is not applicable as the TOE does not utilize default credentials.

FMT_CFG_EXT.1.2

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

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

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

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

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

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

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

The evaluator examined the TOE files on the RHEL and Windows file system and confirmed they were protected with the adequate permissions.

2.4.2 FMT_MEC_EXT.1 Supported Configuration Mechanism

2.4.2.1 TSS Evaluation 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.

[ST] Section 8.1.4 FMT_MEC_EXT.1: states that the TOE does not store or set any security-related settings. Non-security-related settings are stored on the Management Server and are queried when performing tasks. The TOE does not provide any management features that write or change settings.

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.

The TOE does not implement functionality to encrypt and store configuration options as defined by FDP_PRT_EXT.1 in the PP-Module for File Encryption and does not claim conformance to this Module. Therefore, this assurance activity is N/A.

2.4.2.2 Guidance Evaluation Activity

None.

2.4.2.3 Test Evaluation Activity

Modified per TD0747.

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 inspect the TSS and verify that it describes what Android API is used (and provides a link to the documentation of the API) when storing configuration data**.** The evaluator shall run the application and verify that the behavior of the TOE is consistent with where and how the API documentation says the configuration data will be stored.

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 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 system-specific 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 system-specific 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.

The evaluator performed this test in conjunction with FMT_CFG_EXT.1.2.

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.

This is not applicable as the TOE does not select the PP-Module for File Encryption.

2.4.3 FMT_SMF.1 Specification of Management Functions

2.4.3.1 TSS Evaluation Activity

None.

2.4.3.2 Guidance Evaluation Activity

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

The TOE claims no management functions.

2.4.3.3 Test Evaluation 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.

This is not applicable as the TOE claims no management functions.

2.5 Privacy (FPR)

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

2.5.1.1 TSS Evaluation Activity

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

The evaluator examined the TSS to determine if it identifies the application functionality that transmits PII. The evaluator found in section 8.1.5 of the ST that the product may back up and restore data that is considered PII. However, the TOE does not inspect the data that is to be backed up or recovered and thus uses the Application Note case of the PP applies where the PII is not specifically requested, and the user would be volunteering the information through the act of placing the PII data in a location that is backed up by the TOE.

2.5.1.2 Guidance Evaluation Activity

None.

2.5.1.3 Test Evaluation Activity

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.

This test is not applicable because the ST does not make the require user approval before executing selection.

2.6 Protection of the TSF (FPT)

2.6.1 FPT_AEX_EXT.1 Anti-Exploitation Capabilities

2.6.1.1 TSS Evaluation 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 8.1.6: The TOE is compiled with ASLR enabled and can run successfully with Windows Defender Exploit Guard 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 TOE is also compatible with SELinux enabled and in enforcing mode. The TOE does not write user-modifiable files to directories that contain executable files.

The Windows Agent TOE is compiled with the /GS flag enabled by default for stack-based buffer overflow protection and the /NXCOMPAT flag to enable DEP protections for the application. The

Linux Agent uses the __stack_chk_fail symbol in ELF executable files for stack-based buffer overflow protection.

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 Evaluation 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 Evaluation Activity

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

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

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

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

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

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

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

The evaluator exercised the appropriate utility for each platform, Windows and RHEL, and verified that there were no shared memory mapped locations.

FPT_AEX_EXT.1.2

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

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

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

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

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

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

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

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

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

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

The evaluator examined the Windows application files and verified the executables all indicated that they passed the NXCheck and were NX Compatible.

The evaluator performed static analysis on the Linux application and verified that mmap is never invoked with both PROT_WRITE and PROT_EXEC and that mprotect is never invoked with PROT_EXEC.

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.

Apple macOS: The evaluator shall ensure that the application can successfully run on macOS without disabling any security features.

For the RHEL platform, the evaluator installed and tested the TOE with SELinux enabled.

For the Windows platform, the evaluator confirmed the TOE was installed and tested with the listed 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.

This test is performed in conjunction with FMT_CFG_EXT.1.2.

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 verified that each platform version of the TOE includes the relevant stack-based buffer overflow protections for its platform.

2.6.2 FPT_API_EXT.1 Use of Supported Services and APIs

2.6.2.1 TSS Evaluation Activity

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

The evaluator examined the TSS to determine if it lists the platform APIs used in the application. A complete list of APIs used by the TOE is located in Appendix A.

2.6.2.2 Guidance Evaluation Activity

None.

2.6.2.3 Test Evaluation 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 performed web searches to confirm that the claimed APIs were valid and documented by the developer. Valid documented API's are determined by examining the results of a public search (i.e., Google) for the API and verifying that the platform vendor references the API in question (i.e., Windows APIs cited in the ST are documented by guidance at docs.microsoft.com and Linux APIs cited in the ST are documented at linux.die.net or man7.org).

2.6.3 FPT_IDV_EXT.1 Software Identification and Versions

2.6.3.1 TSS Evaluation Activity

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

"other version information" is not selected in this SFR. The TOE uses SWID tags that comply with minimum requirements from ISO/IEC 19770-2:2015 for versioning.

2.6.3.2 Guidance Evaluation Activity

None.

2.6.3.3 Test Evaluation Activity

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 TOE, on both the Windows and RHEL platforms, possessed a SWID tag file and verified that the tag contains a Software Identity and an Entity element.

2.6.4 FPT_LIB_EXT.1 Use of Third Party Libraries

2.6.4.1 TSS Evaluation Activity

None.

2.6.4.2 Guidance Evaluation Activity

None.

2.6.4.3 Test Evaluation Activity

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 confirmed that any third-party libraries packaged with the TOE, on both the Windows and RHEL platforms, were listed in the ST.

2.6.5 FPT_TUD_EXT.1 Integrity for Installation and Update

2.6.5.1 TSS Evaluation Activity

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 8.1.6: The TOE's installation package and its updates are digitally signed using a 2048-bit RSA key and SHA-256 digest algorithm so that the platform can verify their signatures

before installation. The authorized sources of the Linux and Windows installer signatures are Acronis International GmbH, issued by GlobalSign and Acronis SCS, Inc., issued by DigiCert respectively.

[ST] Section 8.2 describes how candidate updates are obtained. The description indicates that when a new version of the TOE is available, it can be downloaded from the Acronis SCS website.

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

[ST] Section 8.1.6 indicates that the TOE is distributed as an additional software package to the platform OS. It is packaged in the standard executable (.exe) format for Windows and the executable binary (.x86_64) format for Linux. It is packaged such that its removal results in the deletion of all traces of the application, with the exception of configuration settings, output files, and audit/log events.

2.6.5.2 Guidance Evaluation Activity

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

[CCSUPP] Section 3.1.3 has the procedures to perform an update. On Linux, to check for an update, the user of the platform runs the shell script “Update.sh”. On Windows, to check for an update, the user of the platform runs the Windows PowerShell script “Update.ps1”.

If an update is available, an administrative user will use the Management Server’s web interface to get the download link from the About page and download a copy of the installation file and follow additional instructions provided in the [CCSUPP] Section 3.1.3.

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

[CCSUPP] Section 3.1.3 provides the instructions to query the current version. On Linux, to check the current version of the TOE, the user of the platform runs the shell script “Update.sh”. On Windows, to check the current version of the TOE, the user of the platform runs the Windows PowerShell script “Update.ps1”. Regardless of whether an update is found, the script will report the current installed version 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 Evaluation Activity

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 a TOE update on each platform, Windows and RHEL, and verified that the TOE version was reported and there was no newer version available.

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.

This test was performed in conjunction with FPT_TUD_EXT.1.1.

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).
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 hashed, on both the Windows and RHEL platforms, the TOE executables and utilized the TOE. The evaluator then hashed the executables again and verified the hashes did not change, thus the executables were not modified.

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 Evaluation 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.

Section 8.1.5 states that the installation packages are digitally signed using a 2048-bit RSA key and SHA-256 digest algorithm. The authorized sources of the Linux and Windows installer signatures are Acronis International GmbH, issued by GlobalSign and Acronis SCS, Inc., issued by DigiCert respectively.

2.6.6.2 Guidance Evaluation Activity

FPT_TUD_EXT.2.1, FPT_TUD_EXT.2.2, and FPT_TUD_EXT.2.3
None.

2.6.6.3 Test Evaluation Activity

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 ~~**supported by the platform**~~. 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.

For RHEL, the evaluator confirmed that the TOE was rpm formatted and for Windows the evaluator confirmed that the TOE was .EXE formatted.

FPT_TUD_EXT.2.2
Modified by TD0664
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.~~

All Other Platforms...

The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.

The evaluator surveyed all the files present on the system prior to installing the TOE, on both the Windows and RHEL platforms. The evaluator installed and utilized the TOE. The evaluator then uninstalled the TOE. After the TOE was uninstalled, the evaluator surveyed all of the files present on the system. The evaluator verified that the TOE did not leave any executable files on the system.

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 Evaluation 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.

The TOE does not utilize the platform for encryption of transmitted data, the TOE instead implements functionality to encrypt data in transit using TLS. Therefore, this activity is N/A.

2.7.1.2 Guidance Evaluation Activity

None.

2.7.1.3 Test Evaluation Activity

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.

This test was performed in conjunction with FDP_NET_EXT.1.1 Test 1.

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.

This test was performed in conjunction with FDP_NET_EXT.1.1 Test 1.

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.

This test was performed in conjunction with FDP_NET_EXT.1.1 Test 1.

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 uses-permission-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.

This is not applicable as the TOE does not claim the use of Android or Apple iOS.

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 Evaluation Activity

None defined.

3.3.1.2 Guidance Evaluation 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.

The TOE provides cryptographic functions through its Acronis SCS Cryptographic Library. The [CCSUPP] Section 3.1.1 states that there are no management options to change the settings for the implemented cryptographic library. The TOE is already configured with the appropriate setting to meet the security requirements outlined in the Security Target.

[CCSUPP] Section 3.1.3 describes how to obtain and perform an update and how the update is made. If an update is available after following the steps to check for an update, use the Management Server's web interface to get the download link from the About page and download a copy of the installation file. If downloaded to a machine that is not the TOE's host, transfer the file to the TOE's host before continuing. Verification of the installation file prior to installation is covered in sections 2.2.2.2 for Windows and 2.2.4.2 for Linux.

3.3.1.3 Test Evaluation Activity

None defined.

3.3.2 AGD_PRE.1 Preparative Procedures

3.3.2.1 TSS Evaluation Activity

None defined.

3.3.2.2 Guidance Evaluation 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.

[ST] Section 1.4: In the evaluated configuration, the TOE is installed on a Microsoft Windows 10 OS and on a RHEL v7.9 OS that are on a network connected to a Management Server (Acronis Backup Server) in the TOE environment. The [CCSUPP] and [USER] cover installation of the TOE on the Windows and Linux platforms.

3.3.2.3 Test Evaluation 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 Evaluation Activity

None defined.

3.4.1.2 Guidance Evaluation 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 1.2 of [ST] ("Security Target and TOE Reference") includes the TOE identification. The TOE is identified in terms of software included in the evaluated configuration: Acronis SCS Cyber Backup 12.5 Hardened Edition Agent v12.5. Section 1.4 "TOE Overview" describes the Agent as comprising Windows Agent installed on a Windows 10 machine and Linux Agent installed on a RHEL v7.9 machine both with a separately downloadable version-check tool: Acronis SCS Version-check v1.19 (as identified by Table 2).

The Acronis SCS website: https://acronisscs.com/ identifies the product "Acronis SCS Hardened Backup" which is the product "Acronis SCS Cyber Backup 12.5 Hardened Edition". This identifier is clearly distinguished from the other products identified on the website and is consistent with the TOE version identified in the ST.

Testing confirmed that after installing the Server component according to [CCSUPP], the version was Hardened Edition version 12.5. The version-check tool was observed as included in the download path and the [CCSUPP] provided clear instruction to download the tool as part of the TOE. The version was confirmed as v1.19.

The security target, the guidance, and the evaluated version of the TOE are all in agreement with the TOE definition in the ST.

3.4.1.3 Test Evaluation Activity

None defined.

3.4.2 ALC_CMS.1 TOE CM Coverage

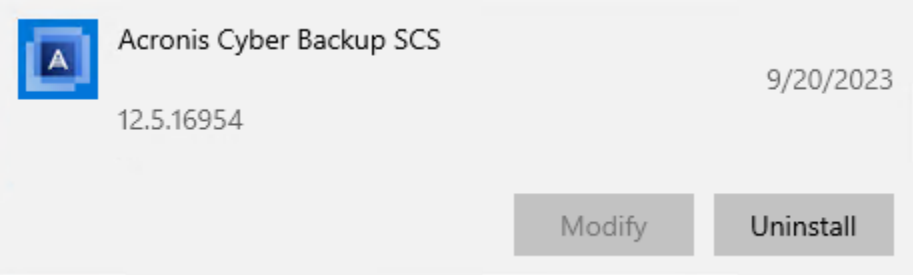
3.4.2.1 TSS Evaluation Activity

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.

3.4.2.2 Guidance Evaluation 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.

As described in Section **Error! Reference source not found.** above, the evaluator confirmed the TOE is labelled with a unique software version identifier.




```
[acronisscs@linuxagent Downloads]$ sudo rpm --query --all | grep Agent
BackupAndRecoveryAgent-12.5.16954-1.x86_64
[acronisscs@linuxagent Downloads]$
```

[CCSUPP] Section 2.1 identifies the specific operating system platforms for the TOE as Microsoft Windows 10 and RHEL v7.9.

Section 2.2.3 indicates that the Windows Agent TOE application was built with stack buffer overflow protection using the /GS flag and automatically installs the required Microsoft Visual C++ files in the operating environment. These files are needed for the TOE to operate correctly. There are no manual steps for configuring stack buffer overflow protection and it is enabled automatically.

Section 2.2.5 states that the Linux Agent TOE application was built with stack buffer overflow protection using the __stack_chk_fail symbol in ELF executable files. There are no manual steps for configuring stack buffer overflow protection and it is enabled automatically.

The [USER] and [CCSUPP] are identified with the TOE's unique identifying information: "Acronis Cyber Backup 12.5 SCS Hardened Edition". The [CCSUPP] provides instructions specific to installing and configuring the Acronis Cyber Backup 12.5 SCS Hardened Edition Agent with references into the [USER] specific for the TOE. The User Guide covers both the Server (operational environment) and Agent (TOE) components. "12.5 SCS Hardened Edition" provides

the unique identifier (with respect to other products from the TSF vendor) as there is also a free 30-day trial version labelled as Acronis Cyber Backup.

3.4.2.3 Test Evaluation Activity

None defined.

3.4.3 ALC_TSU_EXT.1 Timely Security Updates

3.4.3.1 TSS Evaluation 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.

[ST] Section 8.2 contains the description of the timely security update process used by the developer to create and deploy security updates to the application. It describes the developer process and mechanism of deployment. Security updates are categorized according to severity and fixed accordingly. Issues categorized as critical are fixed immediately, issues categorized as high are provided within 3-4 weeks (15-20 business days), issues categorized as low or medium are provided with the next major version or update. Issue severity is calculated according to CVSSv3 methodology. Any update that is released is deployed to the Acronis SCS website for download. Customers may refer to the email or use the check for update process to see if a new version is available for their installation. Updates can then be downloaded and applied to the TOE as needed.

The Acronis SCS Support team will notify customers about security issues related to the TOE for critical and high severity issues where the issue is known to 3rd-party (external report or a known exploitation).

The notification will be sent to customers and include enough information to understand the following:

1. The risk associated with the issue

2. Conditions under which a customer's system is vulnerable

3. Necessary steps to mitigate the risk

Customers that purchase the TOE may email appsupport@acronisscs.com to report security issues pertaining to the TOE. A public key and disclosure policy are posted to the Acronis SCS GitHub (https://github.com/acronisscs/public_disclosure) for use in securing the contents of any security related email.

3.4.3.2 Guidance Evaluation Activity

None defined.

3.4.3.3 Test Evaluation Activity

None defined.

3.5 Class ATE: Tests

3.5.1 ATE_IND.1 Independent Testing – Conformance

3.5.1.1 TSS Evaluation Activity

None defined.

3.5.1.2 Guidance Evaluation Activity

None defined.

3.5.1.3 Test Evaluation Activity

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

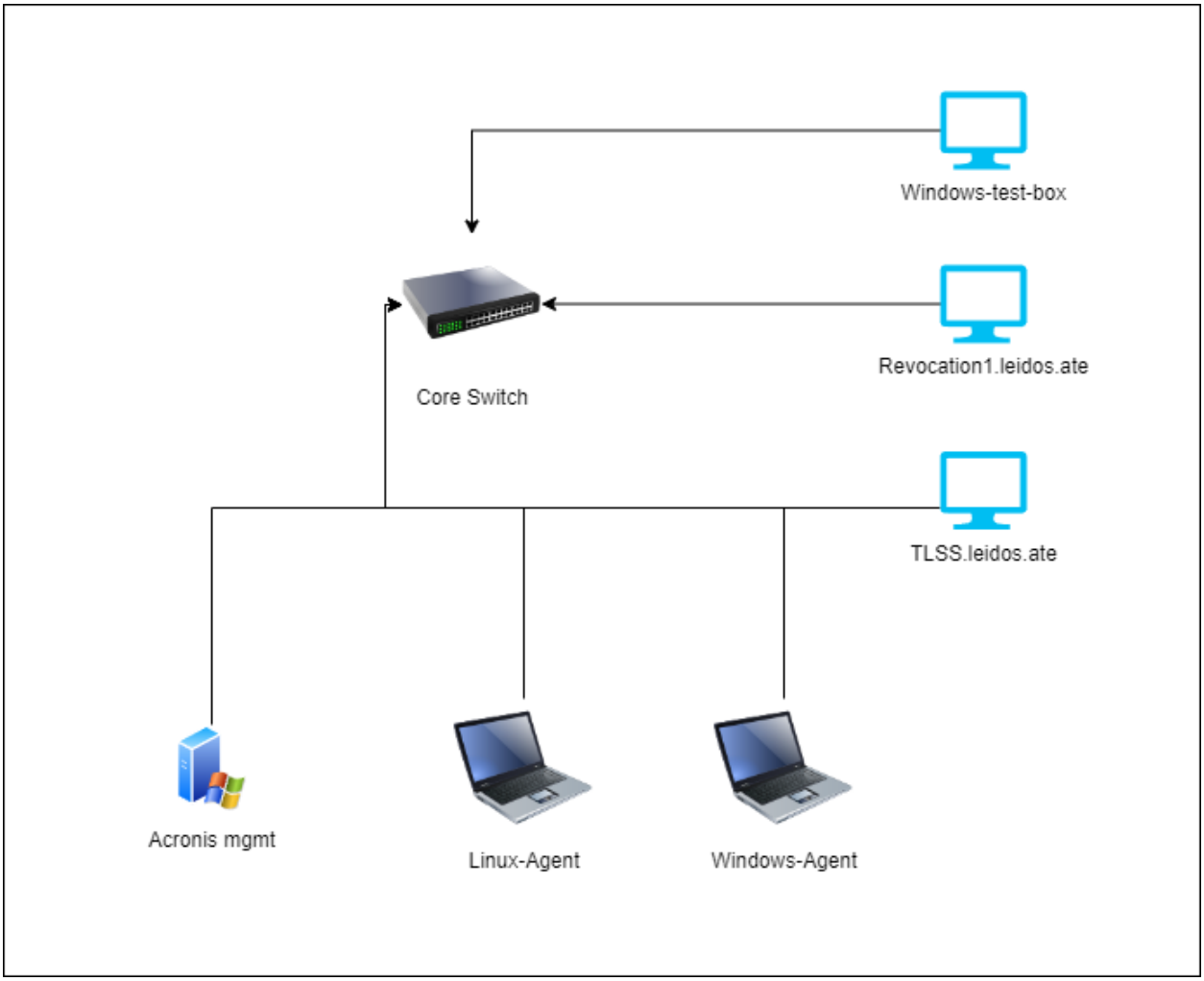
While it is not necessary to have one test case per test listed in an evaluation activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test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

This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this PP and used by the cryptographic protocols being evaluated (e.g SSH). The test plan identifies high-level test

objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

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

The evaluator prepared a test plan and report documenting the testing aspects of the system of the testing actions contained in the [CEM] and the body of this PP's Assurance Activities. A proprietary test report was created which contains all testing test results, test configurations, test platforms and rational. The AGD documentation was followed to install and setup the TOE on each platform. No application crashes were encountered during product testing. No equivalency rational was utilized or generated. Testing of the TOE was performed at the Leidos Accredited Testing and Evaluation Lab located in Columbia, Maryland from November 28, 2022, to September 21, 2023



The evaluation team used the following components to create the test configuration:

TOE Platform:

Windows 10 and RHEL 7 agent to host the TOE.

Test Configuration Devices:

Windows Server 2016 Standard hosts the Management server.

Windows test box to access the Management server.

Lab equipment to facilitate specific tests against the TOE (TLSS, revocation1)

3.6 Class AVA: Vulnerability Assessment

3.6.1 AVA_VAN.1 Vulnerability Survey

3.6.1.1 TSS Evaluation Activity

None defined.

3.6.1.2 Guidance Evaluation Activity

None defined.

3.6.1.3 Test Evaluation 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 evaluator performed vulnerability searches using the following databases:

- https://nvd.nist.gov/
- https://www.kb.cert.org/vuls/

Searches were performed using the following search terms:

- Acronis SCS

- Cyber Backup
- TLS 1.2
- Acronis SCS Cryptographic Library
- SCS Version-check
- All third-party libraries listed in the ST.

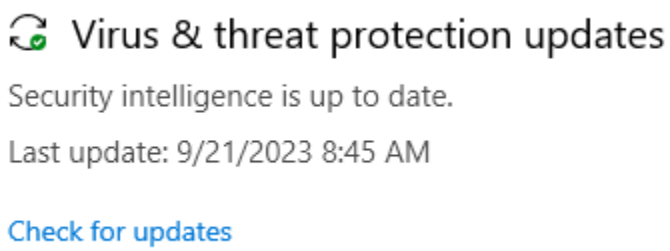
No vulnerabilities were identified for the TOE. Searches of public vulnerability repositories were performed during the evaluation on June 12, 2023, and then re-performed a final time on October 11, 2023, to ensure that no additional public vulnerabilities were disclosed prior to the completion of the evaluation. During this search, the following CVEs were uncovered with regards to "Acronis Agent" product that is sold to the public (as opposed to the TOE which is intended for government customers). While this is not the same product as the Acronis SCS Hardened Edition, they do share some code. Therefore, the CCTL relayed them to the vendor for further diagnosis. The following CVEs were determined to affect the TOE and required corrective action via the vendor's timely security update process outlined in the Security Target:

- CVE-2023-44209
- CVE-2023-44210
- CVE-2023-44212

A fourth CVE, CVE-2023-45244 was initially considered to apply to the TOE but was determined not to be applicable on further review. The identified CVEs were adequately mitigated via build 17000 of the product, released on November 22, 2023. Specific details on these CVEs are included in [AVA], as well as the remainder of the vulnerability analysis.

The evaluator performed virus scans on all TOE directories (including executables) and installer files and verified that no files were identified as threats.

The virus scanner utilized on windows was Windows Defender using the following definitions:



The virus scanner utilized on Red Hat Enterprise Linux (RHEL) was ClamAV using the following definitions:

```
[root@linuxagent bin]# ./freshclam
ClamAV update process started at Thu Sep 21 12:03:03 2023
daily.cvd database is up-to-date (version: 27038, sigs: 2041081, f-level: 90, builder: raynman)
main.cvd database is up-to-date (version: 62, sigs: 6647427, f-level: 90, builder: sigmgr)
bytecode.cvd database is up-to-date (version: 334, sigs: 91, f-level: 90, builder: anvilleg)
[root@linuxagent bin]#
```