HCL BigFix Server Version 11.0.3 Assurance Activity Report

Version:	1.1
Date:	2025-05-07
Status:	FINAL
Classification:	Public
Product:	HCL BigFix Server Version 11.0.3
Sponsor:	HCL Technologies Limited
Evaluation Facility:	atsec information security corporation
Validation ID:	11481
Validation Body:	NIAP CCEVS
Author(s):	Joachim Vandersmissen, Hunter Barton
Quality Assurance:	Trang Huynh, Evan Barnett

atsec information security corporation
4516 Seton Center Pkwy, Suite 250
Austin, TX 78759

Phone: +1 512-615-7300
Fax: +1 512-615-7301
www.atsec.com

Classification Note

Public Information (public)

This classification level is for information that may be made available to the general public. No specific security procedures are required to protect the confidentiality of this information. Information classified “public” may be freely distributed to anyone inside or outside of atsec.

Information with this classification shall be clearly marked “public”, except that it is not required to mark “public” on printed marketing material obviously intended for publication.

Revision History

Version	Date	Author(s)	Changes to Previous Revision
1.0	2025-03-26	Joachim Vandersmissen	First version
1.1	2025-05-07	Joachim Vandersmissen	Address validator feedback

Table of Contents

1 Evaluation Basis and Documents....5

2 Evaluation Results....6

2.1 CAVP Summary....6
2.2 Security Functional Requirements....7
2.2.1 Cryptographic support (FCS)....7
2.2.1.1 Cryptographic Asymmetric Key Generation (FCS_CKM.1/AK)....7
2.2.1.2 Cryptographic Key Establishment (FCS_CKM.2)....10
2.2.1.3 Cryptographic Key Generation Services (FCS_CKM_EXT.1)....13
2.2.1.4 Password Conditioning (FCS_CKM_EXT.1/PBKDF)....13
2.2.1.5 Cryptographic Operation - Hashing (FCS_COP.1/HASH)....14
2.2.1.6 Cryptographic Operation - Keyed-Hash Message Authentication (FCS_COP.1/KEYEDHASH)....15
2.2.1.7 Cryptographic Operation - Signing (FCS_COP.1/SIG)....15
2.2.1.8 Cryptographic Operation - Encryption/Decryption (FCS_COP.1/SKC)....16
2.2.1.9 HTTPS Protocol (FCS_HTTPS_EXT.1/CLIENT)....21
2.2.1.10 HTTPS Protocol (FCS_HTTPS_EXT.1/SERVER)....22
2.2.1.11 Random Bit Generation Services (FCS_RBG_EXT.1)....24
2.2.1.12 Random Bit Generation from Application (FCS_RBG_EXT.2)....25
2.2.1.13 Storage of Credentials (FCS_STO_EXT.1)....27
2.2.1.14 TLS Protocol (FCS_TLS_EXT.1)....28
2.2.1.15 TLS Client Protocol (FCS_TLSC_EXT.1)....29
2.2.1.16 TLS Client Support for Renegotiation (FCS_TLSC_EXT.4)....35
2.2.1.17 TLS Client Support for Supported Groups Extension (FCS_TLSC_EXT.5)....36
2.2.1.18 TLS Server Protocol (FCS_TLSS_EXT.1)....37
2.2.2 User data protection (FDP)....41
2.2.2.1 Encryption Of Sensitive Application Data (FDP_DAR_EXT.1)....41
2.2.2.2 Access to Platform Resources (FDP_DEC_EXT.1)....42
2.2.2.3 Network Communications (FDP_NET_EXT.1)....44
2.2.3 Identification and authentication (FIA)....45
2.2.3.1 X.509 Certificate Validation (FIA_X509_EXT.1)....45
2.2.3.2 X.509 Certificate Authentication (FIA_X509_EXT.2)....50
2.2.4 Security management (FMT)....51
2.2.4.1 Secure by Default Configuration (FMT_CFG_EXT.1)....51
2.2.4.2 Supported Configuration Mechanism (FMT_MEC_EXT.1)....53
2.2.4.3 Specification of Management Functions (FMT_SMF.1)....54
2.2.5 Privacy (FPR)....55
2.2.5.1 User Consent for Transmission of Personally Identifiable Information (FPR_ANO_EXT.1)....55
2.2.6 Protection of the TSF (FPT)....56
2.2.6.1 Anti-Exploitation Capabilities (FPT_AEX_EXT.1)....56
2.2.6.2 Use of Supported Services and APIs (FPT_API_EXT.1)....61
2.2.6.3 Software Identification and Versions (FPT_IDV_EXT.1)....62
2.2.6.4 Use of Third Party Libraries (FPT_LIB_EXT.1)....62
2.2.6.5 Integrity for Installation and Update (FPT_TUD_EXT.1)....63
2.2.6.6 Integrity for Installation and Update (FPT_TUD_EXT.2)....65
2.2.7 Trusted path/channels (FTP)....67
2.2.7.1 Protection of Data in Transit (FTP_DIT_EXT.1)....67
2.3 Security Assurance Requirements....70
2.3.1 Development (ADV)....70
2.3.1.1 Basic functional specification (ADV_FSP.1)....70

2.3.2 Guidance documents (AGD)..... 70
2.3.2.1 Operational user guidance (AGD_OPE.1)..... 70
2.3.2.2 Preparative procedures (AGD_PRE.1)..... 71
2.3.3 Life-cycle support (ALC)..... 71
2.3.3.1 Labelling of the TOE (ALC_CMC.1)..... 71
2.3.3.2 TOE CM coverage (ALC_CMS.1)..... 71
2.3.3.3 Timely Security Updates (ALC_TSU_EXT.1)..... 72
2.3.4 Tests (ATE)..... 72
2.3.4.1 Independent testing - conformance (ATE_IND.1)..... 72
2.3.5 Vulnerability assessment (AVA)..... 74
2.3.5.1 Vulnerability survey (AVA_VAN.1)..... 74

1 Evaluation Basis and Documents

This evaluation is based on the "Common Criteria for Information Technology Security Evaluation" Version 3.1 Revision 5 [CC] , the "Common Methodology for Information Technology Security Evaluation" [CEM] and the additional assurance activities defined in the following:

- [PP_APP_V1.4]: Protection Profile for Application Software, Version 1.4, dated 2021-10-07.
- [PKG_TLS_V1.1]: Functional Package for Transport Layer Security (TLS), Version 1.1, dated 2019-03-01.

This evaluation claims Exact Compliance with the above PP and Functional Package.

The following scheme documents and interpretations have been considered:

- [CCEVS-LG]: "CCEVS LabGrams", version as of May 2025.
- [CCEVS-PL]: "CCEVS Scheme Policy Letters", version as of May 2025.
- [CCEVS-PUB]: "CCEVS Scheme Publications", version as of May 2025.
- [CCEVS-TD]: "Technical Decisions", version as of May 2025.

2 Evaluation Results

The evaluator work units have been performed, including: evaluator actions and analysis explicitly stated in the CEM; evaluator actions implicitly derived from developer action elements described in the CC Part 3; and evaluator confirmation that requirements for content and presentation of evidence elements described in the CC Part 3 have been met.

The evaluation was performed by informal analysis of the evidence provided by the sponsor.

2.1 CAVP Summary

The TOE uses the following cryptographic libraries:

- the OpenSSL cryptographic module that is included in the TOE;
- the Cryptographic API Next Generation (CNG), provided by the underlying Windows platform.

The table below shows the cryptographic services used by the TOE and provided by the OpenSSL cryptographic module that is included in the TOE, describing the algorithms, their supported key sizes, applicable standard and purpose. The table also includes the certificates obtained from the Cryptographic Algorithm Validation Program (CAVP) in the evaluated configuration for each of the cryptographic algorithms.

Table 1: Mapping of SFRs to CAVP certificates (OpenSSL cryptographic module)

SFR	Algorithm	Key sizes	Standard	CAVP cert.
FCS_CKM.1/AK - Asymmetric Key Generation	Elliptic Curve Cryptography (ECC)	P-256, P-384, P-521 (256 to 521 bits)	[FIPS186-5]	A5321
	Finite Field Cryptography (FFC)	3072 bits	[FIPS186-4]	A5321
	Finite Field Cryptography (FFC)	Safe primes ffdhe2048, ffdhe3072, ffhde4096, ffdhe6144, ffdhe8192 (2048 to 8192 bits)	[SP800-56A-Rev3]	A5331
FCS_CKM.2 - Key Establishment	RSA RSAES-PKCS1-v1_5	2048 bits or greater	[RFC8017]	CCTL Tested
	Elliptic Curve Cryptography (ECC)	P-256, P-384, P-521 (256 to 521 bits)	[SP800-56A0Rev3]	A5321
	Finite Field Cryptography (FFC)	3072 bits	[SP800-56A-Rev3]	A5331
	Finite Field Cryptography (FFC)	Safe primes ffdhe2048, ffdhe3072, ffhde4096,	[SP800-56A-Rev3]	A5331

		ffdhe6144, ffdhe8192 (2048 to 8192 bits)		
FCS_COP.1/SKC - Data Encryption and Decryption	AES in CBC mode	128, 256 bits	[SP800-38A]	A5311
	AES in GCM mode	128, 256 bits	[SP800-38D]	A5316
FCS_COP.1/Hash - Message Digest	SHA-1, SHA2-256, SHA2-384, SHA2-512	N/A	[FIPS180-4]	A5321
FCS_COP.1/Sig - Digital Signature Generation and Verification	RSA digital signature generation and verification	2048, 3072, 4096 bits	[FIPS186-5]	A5321
	ECDSA digital signature generation and verification	P-256, P-384, P-521	[FIPS186-5]	A5321
FCS_COP.1/ KeyedHash - Message Authentication Code	HMAC with SHA-1, SHA2-256, SHA2-384	160, 256, 384 bits	[FIPS198-1]	A5321
FCS_CKM_EXT.1/ PBKDF - Password-Based Key Derivation	PBKDF	512 bits	[SP800-132]	A5321
FCS_RBG_EXT.2 - Random Number Generator	DRBG (CTR_DRBG)	256 bits	[SP800-90A-Rev1]	A5309

2.2 Security Functional Requirements

2.2.1 Cryptographic support (FCS)

2.2.1.1 Cryptographic Asymmetric Key Generation (FCS_CKM.1/AK)

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM.1-AK-ASE-01

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.

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.

Summary

The evaluator verified that Section 7.1.1 *Cryptographic Support* in the TSS of [ST] identifies the two schemes supported by the TOE and their key sizes: Elliptic Curve Cryptography

(ECC) and Finite Field Cryptography (FFC). Their usage is identified as ephemeral key pair generation for TLS key exchange.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM.1-AK-AGD-01

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.

Summary

Key generation is performed for the purposes of TLS key exchange. The evaluator verified that Section 5 of [CCGUIDE] states that “the default TLS configuration of the TOE is suitable for the Common Criteria evaluated configuration.” There is no specific configuration for the required key generation schemes and key sizes.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM.1-AK-ATE-01

[TD0860] 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 186-5 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,*
- *216 ≤ e ≤ 2256 and e is an odd integer,*
- *|p-q| > 2nlen/2 - 100,*
- *p ≥ 2nlen/2 -1/2,*
- *q ≥ 2nlen/2 -1/2,*
- *2(nlen/2) < d < LCM(p-1,q-1),*
- *e·d = 1 mod LCM(p-1,q-1).*

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-5 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 186-5 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 ≤x ≤ q-1*
- *len(q) + 64 bit output of RBG, followed by a mod q-1 operation where 1≤ x≤q-1.*

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

- *g ≠ 0,1*
- *q divides p-1*

- *gq mod p = 1*
- *gx mod 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.

Summary

The evaluator verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the ECC and FFC key generation algorithms, and the certificate information is provided in Section 2.1.

2.2.1.2 Cryptographic Key Establishment (FCS_CKM.2)

2.2.1.2.1 FCS_CKM.2.1

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM.2.1-ASE-01

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

Summary

The evaluator verified that Section 7.1.1 *Cryptographic Support* in the TSS of [ST] identifies the three key establishment schemes supported by the TOE: RSA RSAES-PKCS1-v1_5, Elliptic Curve Cryptography (ECC), and Finite Field Cryptography (FFC). The ECC and FFC schemes correspond to the Elliptic Curve Cryptography (ECC), and Finite Field Cryptography (FFC) key pair generation schemes identified in FCS_CKM.1.1, respectively.

The RSA key establishment scheme does not correspond to an RSA key pair generation scheme, as the TOE only acts as a receiver in this scheme. The RSA key establishment scheme is only used by the TLS client (Section 6.1.1.15 *FCS_TLSC_EXT.1 TLS Client Protocol* of [ST]).

The usage for all key establishment schemes is identified as TLS key exchange.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM.2.1-AGD-01

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

Summary

Key establishment is performed for the purposes of TLS key exchange. The evaluator verified that Section 5 of [CCGUIDE] states that "the default TLS configuration of the TOE is

suitable for the Common Criteria evaluated configuration." There is no specific configuration for the key establishment schemes.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM.2.1-ATE-01

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.

Summary

The evaluator verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the ECC and FFC key establishment algorithms, and the certificate information is provided in Section 2.1.

The RSAES-PKCS1-v1_5 key establishment algorithm was tested by using a known good implementation (OpenSSL 3.2.0) and connecting to the TLS client and server implemented by the TOE (see FCS_TLSC_EXT.1 and FCS_TLSS_EXT.1). The successful connection indicates that the algorithm is implemented correctly.

2.2.1.3 Cryptographic Key Generation Services (FCS_CKM_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM_EXT.1.1-ASE-01

*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.*

Summary

The evaluator verified that the application needs asymmetric key generation services. The evaluation activities for FCS_CKM.1/AK are performed as described above.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.1.4 Password Conditioning (FCS_CKM_EXT.1/PBKDF)

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_CKM_EXT.1-PBKDF-ASE-01

Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1/KeyedHash). No explicit testing of the formation of the submask from the input password is required. FCS_CKM_EXT.1.1/PBKDF: The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1.

Summary

The evaluator verified that Section 7.1.1.3 *FCS_CKM_EXT.1/PBKDF* in the TSS of [ST] describes how the password is conditioned, including its output size (512 bits), encoding (none), digest algorithm, iteration count, and salt generation. The salt is generated using the DRBG implemented by the TOE.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.1.5 Cryptographic Operation - Hashing (FCS_COP.1/HASH)

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_COP.1-HASH-ASE-01

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.

Summary

The evaluator verified that Section 7.1.1.5 *FCS_COP.1/HASH* in the TSS of [ST] identifies the usage of the hash functions with the other application cryptographic functions (HMAC, signature generation and verification, and PBKDF).

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_COP.1-HASH-ATE-01

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

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

- ***Test 1:*** *Short Messages Test - Bit oriented Mode. The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 2:*** *Short Messages Test - Byte oriented Mode. The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 3:*** *Selected Long Messages Test - Bit oriented Mode. The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512*

*+ 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*

- ***Test 4:** Selected Long Messages Test - Byte oriented Mode. The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.*
- ***Test 5:** Pseudorandomly Generated Messages Test. This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.*

Summary

The evaluator verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the hash algorithms, and the certificate information is provided in Section 2.1.

2.2.1.6 Cryptographic Operation - Keyed-Hash Message Authentication (FCS_COP.1/KEYEDHASH)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_COP.1-KEYEDHASH-ATE-01

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.

Summary

The evaluator verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the keyed-hash message authentication algorithms, and the certificate information is provided in Section 2.1.

2.2.1.7 Cryptographic Operation - Signing (FCS_COP.1/SIG)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_COP.1-SIG-ATE-01

[TD0860] 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 186-5 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 186-5 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.*

RSA Signature Algorithm Tests

- ***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.*

Summary

The evaluator verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the ECDSA and RSA signature generation and verification algorithms, and the certificate information is provided in Section 2.1.

2.2.1.8 Cryptographic Operation - Encryption/Decryption (FCS_COP.1/SKC)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FCS_COP.1-SKC-AGD-01

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.

Summary

Encryption and decryption is performed for the TLS protocol. The evaluator verified that Section 5 of [CCGUIDE] states that “the default TLS configuration of the TOE is suitable for the Common Criteria evaluated configuration.” There is no specific configuration for the required modes and key sizes.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_COP.1-SKC-ATE-01

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.

AES-CBC Multi-Block Message Test

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

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

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

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

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

AES-GCM Monte Carlo Tests

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

- *128 bit and 256 bit keys*
- *Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.*
- *Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a 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.

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 2^{16} 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.

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.

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.

Summary

The evaluator verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the encryption/decryption algorithms, and the certificate information is provided in Section 2.1.

2.2.1.9 HTTPS Protocol (FCS_HTTPS_EXT.1/CLIENT)

2.2.1.9.1 FCS_HTTPS_EXT.1.1-client

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.1-CLIENT-ASE-01

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

Summary

The evaluator verified that Section 7.1.1.12 *FCS_HTTPS_EXT.1/CLIENT* in the TSS of [ST] states that the TOE uses the well-known cURL library to implement the HTTPS protocol as a client. For decades, the cURL library has been widely used to provide SSL/TLS connections and includes extensive test suites to ensure compliance with all mandatory portions of RFC 2818 (as denoted in the RFC by keywords “MUST”, “MUST NOT”, and “REQUIRED”). From this description, and reinforced by the testing, the evaluator concluded that the TOE complies with RFC 2818.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.1-CLIENT-ATE-01

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

Summary

The evaluator started a local webserver using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the “BES Support” external site to the URL of the local webserver. Additionally, the BigFix Server “ca-bundle.crt” certificate bundle was modified to include the CA certificate used to issue the certificate of the local webserver.

Finally, the evaluator instructed BigFix Server to initiate an HTTPS request to the local webserver. The evaluator observed that the connection succeeded and the traffic is identified as HTTPS.

2.2.1.9.2 FCS_HTTPS_EXT.1.2-client

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.2-CLIENT-ATE-01

Other tests are performed in conjunction with the TLS package.

Summary

The evaluator performed the applicable tests in the TLS package.

2.2.1.9.3 FCS_HTTPS_EXT.1.3-client

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.3-CLIENT-ATE-01

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

- ***Test 1:*** *The evaluator shall demonstrate that using a certificate without a valid certification path results in the selected action in the SFR. If "notify the user" is selected in the SFR, then the evaluator shall also determine that the user is notified of the certificate validation failure. Using the administrative guidance, the evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that again, using a certificate without a valid certification path results in the selected action in the SFR, and if "notify the user" was selected in the SFR, the user is notified of the validation failure.*

Summary

The evaluator started a local webserver using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the “BES Support” external site to the URL of the local webserver. Additionally, the BigFix Server “ca-bundle.crt” certificate bundle was modified to include a “fake” CA certificate with no relation to the certificate of the local webserver.

Finally, the evaluator instructed BigFix Server to initiate an HTTPS request to the local webserver. The evaluator observed that the server certificate failed to validate.

The steps above were repeated twice more: once using the “real” CA certificate used to issue the certificate of the local webserver, and once again using the “fake” CA certificate.

2.2.1.10 HTTPS Protocol (FCS_HTTPS_EXT.1/SERVER)

2.2.1.10.1 FCS_HTTPS_EXT.1.1-server

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.1-SERVER-ASE-01

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

Summary

The evaluator verified that Section 7.1.1.13 *FCS_HTTPS_EXT.1/SERVER* in the TSS of [ST] states that the TOE uses the well-known OpenSSL library to implement the TLS part of the HTTPS protocol as a server. The TLS server part is implemented using the Windows Sockets API. From this description, and reinforced by the testing, the evaluator concluded that the TOE complies with RFC 2818.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.1-SERVER-ATE-01

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

Summary

The evaluator used the REST API provided by BigFix Server to test the TOE's built-in HTTPS server. Firstly, the evaluator configured the TOE to use a server private key and certificate generated by the evaluator.

Then, the evaluator connected to the REST API using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools). The evaluator observed that the connection succeeded and the traffic is identified as HTTPS.

2.2.1.10.2 FCS_HTTPS_EXT.1.2-server

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.2-SERVER-ATE-01

Other tests are performed in conjunction with the TLS package.

Summary

The evaluator performed the applicable tests in the TLS package.

2.2.1.10.3 FCS_HTTPS_EXT.1.3-server

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_HTTPS_EXT.1.3-SERVER-ATE-01

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

Summary

The evaluator performed the applicable tests in the TLS package.

2.2.1.11 Random Bit Generation Services (FCS_RBG_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_RBG_EXT.1-ASE-01

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.

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

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

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

Summary

The evaluator verified that Section 6.1.1.11 *FCS_RBG_EXT.1 Random Bit Generation Services* of [ST] selects “implement DRBG functionality”. This is reinforced by Section 7.1.1.9 *FCS_RBG_EXT.1* in the TSS of [ST], which states that the TOE implements its own DRBG functionality using the OpenSSL cryptographic module.

The evaluator confirmed that FCS_RBG_EXT.2 is included in Section 6.1.1.12 *FCS_RBG_EXT.2 Random Bit Generation from Application* of [ST].

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_RBG_EXT.1-ATE-01

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:

Platform: 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.

Platform: 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.

Platform: iOS

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

Platform: Linux

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

Platform: Solaris

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

Platform: 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).

Summary

This assurance activity is not applicable, as Section 6.1.1.11 *FCS_RBG_EXT.1 Random Bit Generation Services* of [ST] selects “implement DRBG functionality”. We note that the DRBG implementation was tested as part of FCS_RBG_EXT.2.

2.2.1.12 Random Bit Generation from Application (FCS_RBG_EXT.2)

2.2.1.12.1 FCS_RBG_EXT.2.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_RBG_EXT.2.1-ATE-01

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

Implementations Conforming to FIPS 140-2 Annex C.

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

- ***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.*

Implementations Conforming to NIST Special Publication 800-90A

- ***Test 1:*** *The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality.*
  If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).
  If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.
  The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.
  Entropy input: *the length of the entropy input value must equal the seed length.*
  Nonce: *If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.*
  Personalization string: *The length of the personalization string must be less then or equal to seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.*
  Additional input: *the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.*

Summary

The evaluator verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The

results have been validated by the CAVP for the DRBG algorithms, and the certificate information is provided in Section 2.1.

2.2.1.12.2 FCS_RBG_EXT.2.2

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_RBG_EXT.2.2-ASE-01

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.

Summary

The proprietary Entropy Assessment Report (EAR) addresses this assurance activity. The evaluator reviewed the EAR document and determined that the document provides enough information to justify the entropy source of the TOE delivering sufficient entropy for RBG output.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_RBG_EXT.2.2-ATE-01

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

Summary

At this moment, no specific testing is required to complete this assurance activity.

2.2.1.13 Storage of Credentials (FCS_STO_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FCS_STO_EXT.1-ASE-01

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.

Summary

The evaluator verified that Section 7.1.1.11 *FCS_STO_EXT.1* in the TSS of [ST] brings a table listing all persistent credentials needed to meet the requirements in the ST. The evaluator confirmed that the table include the purpose, its storage location, and how it is protected.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FCS_STO_EXT.1-ATE-01

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.

Platform: Android

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

Platform: 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.

Platform: iOS

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

Platform: Linux

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

Platform: Solaris

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

Platform: macOS

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

Summary

The evaluator first verified that the operator password is hashed using PBKDF (i.e., conditioned according to FCS_CKM.1/PBKDF). This was done by inspecting the dbo.USERINFO table in the BFEnterprise database (corresponding to BigFix Server operators). The PasswordHashAlgorithm for the operator was set to PBKDF2, and the PasswordHistory value was recorded. The evaluator then changed the password for the operator using the BigFix Console. Finally, the evaluator verified that the PasswordHistory value in the database changed, therefore proving that the operator password is hashed using PBKDF.

Then, the evaluator verified that the remaining credentials (EncryptedServerSigningKey and EncryptedClientCAKey files) are protected using the Data Protection API (DPAPI). This was done by comparing the first 20 bytes of the file with the string “01 00 00 00 D0 8C 9D DF 01 15 D1 11 8C 7A 00 C0 4F C2 97 EB” in hexadecimal. These bytes indicate that the file is a DPAPI blob.

2.2.1.14 TLS Protocol (FCS_TLS_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLS_EXT.1-AGD-01

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

components.

Summary

The evaluator notes that Section 6.1.1.14 *FCS_TLS_EXT.1 TLS Protocol* of [ST] selects both "TLS as a client" and "TLS as a server". Consequently, the evaluator verified that FCS_TLSC_EXT.1 (Section 6.1.1.15 *FCS_TLSC_EXT.1 TLS Client Protocol*) and FCS_TLSS_EXT.1 (Section 6.1.1.18 *FCS_TLSS_EXT.1 TLS Server Protocol*) are claimed in [ST] as required.

Test Assurance Activities

No assurance activities defined.

2.2.1.15 TLS Client Protocol (FCS_TLSC_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1-ASE-01

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

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.

Summary

The evaluator verified that Section 7.1.1.15 *FCS_TLSC_EXT.1* in the TSS of [ST] identifies the cipher suites supported by the TOE when acting as a client. The evaluator verified that those cipher suites exactly matched those listed for this component.

The evaluator verified that Section 7.1.1.15 *FCS_TLSC_EXT.1* in the TSS of [ST] describes the client's method of establishing reference identifiers. The identity of the TLS server is verified from the X.509 certificate presented by the server using the guidelines in RFC 2818, RFC 2459, and RFC 6125. The reference identifier is established from the configuration data of the Fixlet sites. The aforementioned section in the TSS of the [ST] also confirms that the TSF supports IP address and wildcards. Certificate pinning is not supported nor used by the product.

Section 6.1.1.15 *FCS_TLSC_EXT.1 TLS Client Protocol* of [ST] does not select the option for authorizing override of invalid certificates.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1-AGD-01

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.

Summary

The evaluator verified that Section 5 of [CCGUIDE] states that “the default TLS configuration of the TOE is suitable for the Common Criteria evaluated configuration.” Consequently, there are no instructions required.

The evaluator verified that Section 5.3 of [CCGUIDE] states that the reference identifier is automatically established from the URLs that the TOE uses when connecting. There are no instructions to change this behavior.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1-ATE-01

The evaluator shall also perform the following tests:

- *Test FCS_TLSC_EXT.1:1: 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).*
- *Test FCS_TLSC_EXT.1:2: 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.
- *Test FCS_TLSC_EXT.1:3: 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.*
- *Test FCS_TLSC_EXT.1:4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.*
- *Test FCS_TLSC_EXT.1:5: The evaluator shall perform the following modifications to the traffic:*
  - *Test FCS_TLSC_EXT.1:5.1: 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.*
  - *Test FCS_TLSC_EXT.1:5.2: 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.*
  - *Test FCS_TLSC_EXT.1:5.3: [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.*

- *Test FCS_TLSC_EXT.1:5.4: 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.*
- *Test FCS_TLSC_EXT.1:5.5: [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.*
- *Test FCS_TLSC_EXT.1:5.6: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.*
- *Test FCS_TLSC_EXT.1:5.7: 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.*

[TD0499] The evaluator shall configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection. If the TOE supports certificate pinning, all pinned certificates must be removed before performing Tests 1 through 6. A pinned certificate must be added prior to performing Test 7.

- *Test FCS_TLSC_EXT.1:6: 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.
- *Test FCS_TLSC_EXT.1:7: 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.*
- *Test FCS_TLSC_EXT.1:8: [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.*
- *Test FCS_TLSC_EXT.1:9: 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.*
- *Test FCS_TLSC_EXT.1:10: 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.*
  - *Test FCS_TLSC_EXT.1:10.1: [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.*
  - *Test FCS_TLSC_EXT.1:10.2: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.*
  - *Test FCS_TLSC_EXT.1:10.3: [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.*

- *Test FCS_TLSC_EXT.1:10.4: [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.*

- *Test FCS_TLSC_EXT.1:11: [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.*
- *Test FCS_TLSC_EXT.1:12: [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.*

[TD0513] The evaluator shall demonstrate that using an invalid certificate (unless excepted) results in the function failing as follows, unless excepted:

- *Test FCS_TLSC_EXT.1:13:*
  - *Test 1a: The evaluator shall demonstrate that a server using a certifcate with a valid certification path successfully connects.*
  - *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.*
  - *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.*
- *Test FCS_TLSC_EXT.1:14: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.*
- *Test FCS_TLSC_EXT.1:15: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.*
- *Test FCS_TLSC_EXT.1:16: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.*

Summary

The evaluator started a local TLS server using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the “BES Support” external site to the URL of the local TLS server. Additionally, the BigFix Server “ca-bundle.crt” certificate bundle was modified to include the CA certificate used to issue the certificate of the local TLS server.

Finally, the evaluator instructed BigFix Server to initiate a TLS request to the OpenSSL TLS server. Depending on the specific test, the connection succeeded or failed:

Test FCS_TLSC_EXT.1:1: The connection was successfully established using each of the supported cipher suites listed in the [ST]. The evaluator confirmed from the packet dump that the selected cipher suite was used.

Test FCS_TLSC_EXT.1:2: The evaluator created a server certificate with server authentication selected in the Extended Key Usage field and verified the connection was successfully established. The evaluator then created a second, otherwise identical, server certificate with server authentication *not* selected in the Extended Key Usage field and verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:3: The evaluator created an ECDSA-based server certificate but patched the OpenSSL TLS server to select an RSA-based cipher suite. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:4: The evaluator patched the OpenSSL TLS server to always select the TLS_NULL_WITH_NULL_NULL cipher suite, regardless of the support indicated by the client. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:5.1: The evaluator patched the OpenSSL TLS server to select the “03 05” TLS version to the client, corresponding to a non-existent TLS 1.4 version, regardless of the support indicated by the client. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:5.2: The evaluator patched the OpenSSL TLS server to initiate a TLS 1.1 connection, regardless of the support indicated by the client. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:5.3: The evaluator patched the OpenSSL TLS server to modify the nonce of the server. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:5.4: The evaluator patched the OpenSSL TLS server to modify the selected server cipher suite. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:5.5: The evaluator patched the OpenSSL TLS server to modify the signature in the Server Key Exchange message. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:5.6: The evaluator patched the OpenSSL TLS server to modify the message digest in the Server Finished message. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:5.7: The evaluator patched the OpenSSL TLS server to send another Server Key Exchange message after the Change Cipher Spec message. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:6: The evaluator created a server certificate with an incorrect Common Name and no Subject Alternative Name (SAN) extension and verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:7: The evaluator created a server certificate with a correct Common Name but with an incorrect DNS name entry in the SAN extension. The evaluator verified the connection was rejected as expected. The evaluator repeated this test for an incorrect IP address in the SAN extension.

Test FCS_TLSC_EXT.1:8: The [ST] indicates that the TOE does not mandate the presence of the SAN extension, and falls back to the Common Name if the SAN extension is not present. The evaluator created a server certificate with a correct Common Name and SAN extension and verified the connection was successfully established.

Test FCS_TLSC_EXT.1:9: The evaluator created a server certificate with an incorrect Common Name but with a correct DNS name entry in the SAN extension and verified the connection was successfully established.

Test FCS_TLSC_EXT.1:10.1: The evaluator created a server certificate with a “server.*.domain.com” entry in the Common Name and SAN extension. Then, the evaluator configured the local TLS server to listen on “server.subdomain.domain.com” and initiated a TLS connection from the TOE TLS client. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:10.2: The evaluator created a server certificate with a “*.domain.com” entry in the Common Name and SAN extension. Then, the evaluator configured the local TLS server to listen on “server.domain.com”, “domain.com”, and “server.subdomain.domain.com”, and initiated TLS connections from the TOE TLS client. The evaluator verified that the first connection corresponding to “server.domain.com” was successfully established, and the latter two were rejected as expected.

Test FCS_TLSC_EXT.1:10.3: The evaluator created a server certificate with a “*.com” entry in the Common Name and SAN extension. Then, the evaluator configured the local TLS server to listen on “domain.com” and “server.domain.com”, and initiated TLS connections from the TOE TLS client. The evaluator verified the connections were rejected as expected.

Test FCS_TLSC_EXT.1:10.4: Not applicable as the TOE supports wildcards.

Test FCS_TLSC_EXT.1:11: Not applicable as the TOE does not support URI or Service name reference identifiers.

Test FCS_TLSC_EXT.1:12: Not applicable as the TOE does not support pinned certificates.

Test FCS_TLSC_EXT.1:13 a: The evaluator created a valid server certificate and added the issuer CA certificate to the “ca-bundle.crt” bundle, thereby creating a valid certification path. The evaluator verified the connection was successfully established.

Test FCS_TLSC_EXT.1:13 b: The evaluator created a valid server certificate, identical to the certificate created in Test FCS_TLSC_EXT.1:13 a, but using a different issuer CA certificate, thereby creating an invalid certification path. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:13 c: The evaluator created a valid server certificate, identical to the certificate created in Test FCS_TLSC_EXT.1:13 a, but without adding the issuer CA certificate to the “ca-bundle.crt” bundle, thereby creating an invalid certification path. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:14: The evaluator created a valid server certificate and added the issuer CA certificate to the “ca-bundle.crt” bundle. Then, the evaluator started a local OCSP server and configured the server certificate to be marked as “revoked”. The evaluator verified the connection was rejected as expected. This test was repeated identically for the OCSP stapling revocation functionality (with OCSP stapling enabled on the local TLS server).

Test FCS_TLSC_EXT.1:15: The evaluator created an expired server certificate and added the issuer CA certificate to the “ca-bundle.crt” bundle. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:16: The evaluator created a server certificate with a correct Common Name but with an incorrect DNS name entry in the SAN extension and added the issuer CA certificate to the “ca-bundle.crt” bundle. The evaluator verified the connection was rejected as expected.

2.2.1.16 TLS Client Support for Renegotiation (FCS_TLSC_EXT.4)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.4-ATE-01

The evaluator shall perform the following tests:

- *Test FCS_TLSC_EXT.4:1: 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.*
- *Test FCS_TLSC_EXT.4:2: 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.*
- *Test FCS_TLSC_EXT.4:3: 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.*

Summary

The evaluator started a local TLS server using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the “BES Support” external site to the URL of the local TLS server. Additionally, the BigFix Server “ca-bundle.crt” certificate bundle was modified to include the CA certificate used to issue the certificate of the local TLS server.

Finally, the evaluator instructed BigFix Server to initiate a TLS request to the OpenSSL TLS server. Depending on the specific test, the connection succeeded or failed:

Test FCS_TLSC_EXT.4:1: The evaluator verified the connection succeeded, and the SCSV cipher suite is included in the Client Hello message during the initial handshake.

Test FCS_TLSC_EXT.4:2: The evaluator patched the OpenSSL TLS server to modify the length of the renegotiation_info sent by the server to be non-zero. The evaluator verified the connection was rejected as expected. The evaluator then used an unpatched OpenSSL TLS server (i.e., renegotiation_info has a zero length) and performed the exact same steps. The evaluator verified the connection was successfully established.

Test FCS_TLSC_EXT.4:3: The evaluator patched the OpenSSL TLS server to modify the client_verify_data sent by the server during renegotiation and always automatically renegotiate after a successful handshake. The evaluator verified the renegotiation was performed, but failed as expected and the connection was terminated.

2.2.1.17 TLS Client Support for Supported Groups Extension (FCS_TLSC_EXT.5)

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.5-ASE-01

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

Summary

The evaluator verified that Section 7.1.1.17 *FCS_TLSC_EXT.5* in the TSS of [ST] describes the Supported Groups Extension and its values used for the TLS client.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.5-ATE-01

The evaluator shall also perform the following test:

- *Test FCS_TLSC_EXT.5:1: 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.*

Summary

The evaluator started a local TLS server using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the "BES Support" external site to the URL of the local TLS server. Additionally, the BigFix Server "ca-bundle.crt" certificate bundle was modified to include the CA certificate used to issue the certificate of the local TLS server.

Finally, the evaluator instructed BigFix Server to initiate a TLS request to the OpenSSL TLS server. The evaluator verified the connection was successfully established.

The steps above were repeated for each of the supported curves and groups.

2.2.1.18 TLS Server Protocol (FCS_TLSS_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FCS_TLSS_EXT.1-ASE-01

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component. The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions consistent relative to selections in FCS_TLSS_EXT.1.2. The evaluator shall verify that the TSS describes the key agreement parameters of the server's Key Exchange message.

Summary

The evaluator verified that Section 7.1.1.18 *FCS_TLSS_EXT.1* in the TSS of [ST] identifies the cipher suites supported by the TOE when acting as a server. The evaluator verified that those cipher suites exactly matched those listed for this component.

The evaluator verified that Section 7.1.1.18 *FCS_TLSS_EXT.1* in the TSS of [ST] states that all SSL and TLS versions older than TLS 1.2 are rejected by the TOE. The evaluator verified that those versions exactly matched those selected for this component.

The evaluator verified that Section 7.1.1.18 *FCS_TLSS_EXT.1* in the TSS of [ST] describes the supported groups and curves for Finite Field Cryptography and Elliptic Curve Cryptography key agreement, respectively.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSS_EXT.1-AGD-01

The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS. The evaluator shall verify that the AGD guidance includes any configuration necessary to meet this requirement. The evaluator shall verify that any configuration guidance necessary to meet the requirement must be contained in the AGD guidance.

Summary

The evaluator verified that Section 5 of [CCGUIDE] states that "the default TLS configuration of the TOE is suitable for the Common Criteria evaluated configuration." Consequently, there are no instructions required.

Test Assurance Activities

Assurance Activity AA-FCS_TLSS_EXT.1-ATE-01

The evaluator shall also perform the following tests:

- *Test FCS_TLSS_EXT.1:1: 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).*
- *Test FCS_TLSS_EXT.1:2: The evaluator shall send a Client Hello to the server with a list of cipher suites that does not contain any of the cipher suites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the server denies the connection.*
- *Test FCS_TLSS_EXT.1:3: If RSA key exchange is used in one of the selected ciphersuites, the evaluator shall use a client to send a properly constructed Key Exchange message with a modified EncryptedPreMasterSecret field during the TLS handshake. The evaluator shall verify that the handshake is not completed successfully and no application data flows.*
- *Test FCS_TLSS_EXT.1:4: The evaluator shall perform the following modifications to the traffic:*
    - *Test FCS_TLSS_EXT.1:4.1: [TD0469] This test is removed.*
    - *Test FCS_TLSS_EXT.1:4.2: Modify a byte in the data of the client's Finished handshake message, and verify that the server rejects the connection and does not send any application data.*
    - *Test FCS_TLSS_EXT.1:4.3: [TD0779] Demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption):*
    - *Test FCS_TLSS_EXT.1:4.3i [conditional]: If the TOE does not support session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2) or session tickets according to RFC5077, the evaluator shall perform the following test:*

a) The evaluator shall send a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

b) The evaluator shall verify the server does not send a NewSessionTicket handshake message (at any point in the handshake).

c) The evaluator shall verify the Server Hello message contains a zero-length session identifier or passes the following steps:

Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

d) The evaluator shall complete the TLS handshake and capture the SessionID from the ServerHello.

e) The evaluator shall send a ClientHello containing the SessionID captured in step d). This can be done by keeping the TLS session in step d) open or start a new TLS session using the SessionID captured in step d).

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

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

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

shown in Figure 2 of RFC 4346 or RFC 5246).

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

- *Test FCS_TLSS_EXT.1:4.3iii [conditional]: If the TOE supports session tickets according to RFC5077, the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):*

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE successfully resumes the session in accordance with section 3.1 of RFC 5077.

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

c) The evaluator shall send the TSF a Client Hello with a SessionTicket extension, and observe the TSF responds with a Server Hello with an empty SessionTicket extension. The evaluator shall then send the TSF a invalid Finished message, and observe that the TSF terminates the session without sending a valid newTicket message.

Note: if the TSF sends a newTicket message prior to terminating the session, the evaluator shall confirm the ticket is invalid by attempting to use the ticket to renew the session and observe that the TSF either (1) implicitly rejects the session ticket by performing a full handshake (as shown in figure 3 or 4 of RFC 5077), or (2) terminates the connection in some way that prevents the flow of application data.

- *Test FCS_TLSS_EXT.1:4.4: Send a message consisting of random bytes from the client after the client has issued the ChangeCipherSpec message and verify that the server denies the connection.*

- *Test FCS_TLSS_EXT.5: The evaluator shall send a Client Hello requesting a connection with version SSL 2.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 3.0 and TLS 1.0, and TLS 1.1 if it is selected.*

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

- *Test FCS_TLSS_EXT.6: [conditional] If RSA-based key establishment is selected, the evaluator shall configure the TOE with a certificate containing a supported RSA size and attempt a connection. The evaluator shall verify that the size used matches that which is configured and that the connection is successfully established. The evaluator shall repeat this test for each supported size of RSA-based key establishment.*
- *Test FCS_TLSS_EXT.7: [conditional] If finite-field (i.e. non-EC) Diffie-Hellman ciphers are selected, the evaluator shall attempt a connection using a Diffie-Hellman key exchange with a supported parameter size or supported group. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported parameter size or group.*
- *Test FCS_TLSS_EXT.8: [conditional] If ECDHE ciphers are selected, the evaluator shall attempt a connection using an ECDHE ciphersuite with a supported curve. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported elliptic curve.*

Summary

The evaluator used the REST API provided by BigFix Server to test the TOE's built-in HTTPS server. Firstly, the evaluator configured the TOE to use a server private key and certificate generated by the evaluator.

Then, the evaluator connected to the REST API using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools). Depending on the specific test, the connection succeeded or failed:

Test FCS_TLSS_EXT.1:1: The connection was successfully established using each of the supported cipher suites listed in the [ST]. The evaluator confirmed from the packet dump that the selected cipher suite was used.

Test FCS_TLSS_EXT.1:2: The connection was rejected as expected for each of the unsupported cipher suites (e.g., those using RSA-based key establishment). In addition, the evaluator patched the OpenSSL client to always select the TLS_NULL_WITH_NULL_NULL cipher suite and verified the connection was rejected as expected.

Test FCS_TLSS_EXT.1:3: Not applicable as RSA-based key exchange is not used.

Test FCS_TLSS_EXT.1:4.1: [TD0469] This test is removed.

Test FCS_TLSS_EXT.1:4.2: The evaluator patched the OpenSSL client to modify the message digest in the Client Finished message. The evaluator verified the connection was rejected as expected.

Test FCS_TLSS_EXT.1:4.3i: Not applicable as the TOE supports session resumption.

Test FCS_TLSS_EXT.1:4.3ii: The evaluator first connected and resumed the session to the TOE server with a valid session ID. The evaluator verified the connection and session resumption were successful. Then, the evaluator patched the OpenSSL client to send the Client Finished message before the Change Cipher Spec message, and verified the connection was rejected as expected. Finally, upon resumption of the interrupted session, the TOE server implicitly rejected the attempt by sending a different session ID and performing a full handshake.

Test FCS_TLSS_EXT.1:4.3iii: The evaluator first connected and resumed the session to the TOE server with a valid session ticket. The evaluator verified the connection and session resumption were successful. The evaluator also verified that the TOE server implicitly rejects a modified session ticket by performing a full handshake. Then, the evaluator patched the OpenSSL client to send the Client Finished message before the Change Cipher Spec message, and verified the connection was rejected as expected. Finally, upon resumption of the interrupted session, the TOE server implicitly rejected the attempt by performing a full handshake.

Test FCS_TLSS_EXT.1:4.4: The evaluator patched the OpenSSL client to send a Next Protocol message after the Client Finished message. The evaluator verified the connection was rejected as expected.

Test FCS_TLSS_EXT.1:5: the evaluator verified that all connections for SSL 2.0, SSL 3.0, TLS 1.0, and TLS 1.1 were rejected as expected.

Test FCS_TLSS_EXT.1:6: Not applicable as RSA-based key exchange is not used.

Test FCS_TLSS_EXT.1:7: The connection was successfully established using each of the supported parameter sizes listed in the [ST]. The evaluator confirmed from the packet dump that the selected parameter size was used.

Test FCS_TLSS_EXT.1:8: The connection was successfully established using each of the supported curves listed in the [ST]. The evaluator confirmed from the packet dump that the selected curve was used.

2.2.2 User data protection (FDP)

2.2.2.1 Encryption Of Sensitive Application Data (FDP_DAR_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FDP_DAR_EXT.1-ASE-01

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.*

Summary

The evaluator verified that Section 7.1.2.1 *FDP_DAR_EXT.1* in the TSS of [ST] states that the TOE protects only credentials in accordance with FCS_STO_EXT.1. These credentials are listed in Section 7.1.1.11 *FCS_STO_EXT.1* in the TSS of [ST], including their purpose, storage location, and protection techniques. No other forms of sensitive data are processed by the application.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FDP_DAR_EXT.1-ATE-01

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.

Platform: 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.

Platform: 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.

Platform: 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.

Platform: 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.

Platform: 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.

Platform: 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.

Summary

The evaluator inventoried the directories where BigFix Server creates and stores data files. Then, the evaluator determined that the only sensitive data stored by BigFix Server are the EncryptedServerSigningKey and EncryptedClientCAKey files, whose encryption is verified as part of FCS_STO_EXT.1.

2.2.2.2 Access to Platform Resources (FDP_DEC_EXT.1)

2.2.2.2.1 FDP_DEC_EXT.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FDP_DEC_EXT.1.1-AGD-01

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.

Summary

The evaluator verified that Section 1.3 of [CCGUIDE] states that network connectivity is the only hardware resource accessed by BigFix Server to communicate with other BigFix

platform components (e.g., BigFix Clients) and certain Internet servers (e.g., Fixlet Servers). This is consistent with the selections in [ST].

Test Assurance Activities

Assurance Activity AA-ASPP-FDP_DEC_EXT.1.1-ATE-01

Platform: 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.

Platform: Windows

[TD0822] For Windows Universal Applications the evaluator shall check the AppxManifest.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.

Platform: iOS

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

Platform: Linux

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

Platform: Solaris

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

Platform: macOS

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

Summary

BigFix Server is not a Windows Universal Application. Consequently, the evaluator verified that the list of required hardware resources is provided in the documentation ([CCGUIDE]). Network connectivity is the only hardware resource accessed by BigFix Server.

2.2.2.2.2 FDP_DEC_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FDP_DEC_EXT.1.2-AGD-01

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.

Summary

The evaluator verified that Section 1.3 of [CCGUIDE] states that BigFix Server restricts its access to the Windows Registry and MSSQL Database sensitive information repositories. This is consistent with the selections in [ST]. [CCGUIDE] also includes the justification as to why access is required for these information repositories.

Test Assurance Activities

Assurance Activity AA-ASPP-FDP_DEC_EXT.1.2-ATE-01

Platform: 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.

Platform: Windows

[TD0822] For Windows Universal Applications the evaluator shall check the AppxManifest.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.

Platform: iOS

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

Platform: Linux

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

Platform: Solaris

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

Platform: macOS

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

Summary

BigFix Server is not a Windows Universal Application. Consequently, the evaluator verified that the list of accessed sensitive information repositories is provided in the documentation ([CCGUIDE]). The Windows Registry and the MSSQL database are the only sensitive information repositories accessed by BigFix Server.

2.2.2.3 Network Communications (FDP_NET_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FDP_NET_EXT.1-ATE-01

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.*
- *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).*

Platform: 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.

Summary

Test 1: The evaluator used Wireshark to sniff the network traffic on the wired network interface (NIC1). While Wireshark was running, the evaluator mimicked normal usage of the TOE using BigFix Console. The evaluator verified that all network communications associated with BESRootServer.exe are documented in [ST].

Test 2: The evaluator started BigFix Server, then used netstat in PowerShell to list the open ports on the test machine. The evaluator verified that the ports opened by BESRootServer.exe (LISTENING state) are consistent with [ST].

2.2.3 Identification and authentication (FIA)

2.2.3.1 X.509 Certificate Validation (FIA_X509_EXT.1)

2.2.3.1.1 FIA_X509_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-ASPP-FIA_X509_EXT.1.1-ASE-01

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.

Summary

The evaluator verified that Section 7.1.3.1 *FIA_X509_EXT.1* in the TSS of [ST] states that the certificate validation is implemented in OpenSSL, part of the TOE. The description of the validation algorithm is also provided in the aforementioned section.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FIA_X509_EXT.1.1-ATE-01

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

- *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.*
- *Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing.*
- *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.*
- *Test 4: If any OCSP option is selected, the evaluator shall configure the TSF to reject certificates if it cannot access valid status information, if so configurable. Then 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. 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.*
- *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.)*

- *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.)*
- *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.)*
- *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.*
- *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.*

Summary

The evaluator started a local TLS server using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the “BES Support” external site to the URL of the local TLS server. Additionally, the BigFix Server “ca-bundle.crt” certificate bundle was modified to include the root CA of a certificate chain consisting of four certificates (one root CA, two intermediate CAs, and one server node certificate).

Finally, the evaluator instructed BigFix Server to initiate a TLS request to the OpenSSL TLS server. Depending on the specific test, the connection succeeded or failed:

Test 1a: The evaluator created an (otherwise valid) intermediate CA certificate with a basicConstraints extension containing the “CA=false” entry. The evaluator verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 1b: The evaluator created an (otherwise valid) intermediate CA certificate without a basicConstraints extension. The evaluator verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 1c: The evaluator created an (otherwise valid) intermediate CA certificate with a basicConstraints extension containing the “CA=false” entry. The evaluator verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 1d: The evaluator created an (otherwise valid) intermediate CA certificate without the certificate signing purpose set in the keyUsage extension. The evaluator verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 1e: The evaluator created an (otherwise valid) intermediate CA certificate with a path length set to 0. The evaluator verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 1f: The evaluator created a valid certificate chain. The evaluator verified the connection was successfully established, indicating that the certificate chain validation succeeded. The evaluator then removed an intermediate certificate from the chain and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 2: The evaluator created an (otherwise valid) expired intermediate CA certificate. The evaluator verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 3a: The evaluator created a valid certificate chain. Then, the evaluator started a local OCSP server and configured the server node certificate to be marked as “revoked”. The evaluator verified the connection was rejected as expected. This test was repeated identically for the OCSP Stapling revocation functionality (with OCSP stapling enabled on the local TLS server).

Test 3b: The evaluator created a valid certificate chain. Then, the evaluator started a local OCSP server and configured an intermediate certificate to be marked as “revoked”. The evaluator verified the connection was rejected as expected.

Test 3c: The evaluator created a valid certificate chain. Then, the evaluator started a local OCSP server and configured the server node certificate to be marked as “valid”. The evaluator verified the connection was accepted as expected. Finally, the tester configured the server node certificate to be marked as “revoked” and verified the connection was rejected as expected. This test was repeated identically for the OCSP Stapling revocation functionality (with OCSP stapling enabled on the local TLS server).

Test 4: The evaluator created a valid certificate chain. Then, the evaluator started a local OCSP server, configured the server node certificate to be marked as “valid”, but did not add the OCSP signing purpose to the certificate used to sign the OCSP response. The evaluator verified the connection was rejected as expected. This test was repeated identically for the OCSP Stapling revocation functionality (with OCSP stapling enabled on the local TLS server).

Test 5: The evaluator patched the OpenSSL TLS server to flip a bit in the first byte of the server node certificate, immediately before sending it to the client. The evaluator verified the connection was rejected as expected, indicating that the certificate failed to parse correctly.

Test 6: The evaluator patched the OpenSSL TLS server to flip a bit in the last byte of the server node certificate signature, immediately before sending it to the client. The evaluator verified the connection was rejected as expected, indicating that the certificate failed to validate.

Test 7: The evaluator patched the OpenSSL TLS server to flip a bit in the first byte of the server node certificate public key, immediately before sending it to the client. The evaluator verified the connection was rejected as expected, indicating that the certificate failed to validate.

Test 8: The evaluator created a valid certificate chain, using named elliptic curve parameters for each certificate. The evaluator verified the connection was successfully established, indicating that the certificate chain validation succeeded. These steps were repeated for each of the supported curves.

Test 9: The evaluator created a valid certificate chain, using explicit elliptic curve parameters for an intermediate CA certificate. The evaluator verified the connection was rejected as expected, indicating that the certificate chain validation failed. These steps were repeated for each of the supported curves.

2.2.3.1.2 FIA_X509_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FIA_X509_EXT.1.2-ATE-01

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.*
- *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.*

Summary

The evaluator started a local TLS server using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the “BES Support” external site to the URL of the local TLS server. Additionally, the BigFix Server “ca-bundle.crt” certificate bundle was modified to include the root CA of a certificate chain consisting of four certificates (one root CA, two intermediate CAs, and one server node certificate).

Finally, the evaluator instructed BigFix Server to initiate a TLS request to the OpenSSL TLS server. In all instances, the connections were rejected as expected, indicating that the certificate chain validation failed.

Test 1: The evaluator created an (otherwise valid) intermediate CA certificate without a basicConstraints extension. The evaluator verified the connection was rejected as expected. Then, the evaluator removed the (valid) root CA certificate from the “ca-bundle.crt” file and added the (invalid) intermediate CA certificate. Once again, the evaluator verified the connection was rejected as expected.

Test 2: The evaluator created an (otherwise valid) intermediate CA certificate with a basicConstraints extension containing the “CA=false” entry. The evaluator verified the connection was rejected as expected. Then, the evaluator removed the (valid) root CA certificate from the “ca-bundle.crt” file and added the (invalid) intermediate CA certificate. Once again, the evaluator verified the connection was rejected as expected.

2.2.3.2 X.509 Certificate Authentication (FIA_X509_EXT.2)

TSS Assurance Activities

Assurance Activity AA-ASPP-FIA_X509_EXT.2-ASE-01

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.

Summary

The evaluator verified that Section 7.1.3.2 FIA_X509_EXT.2 in the TSS of [ST] describes that the TOE uses the certificate obtained during the TLS handshake to authenticate the remote TLS server. The server certificate is verified using a certificate bundle included in the installation. This behavior is by design and cannot be configured.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FIA_X509_EXT.2-ATE-01

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.*
- *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.*

Summary

The evaluator started a local TLS server using the OpenSSL library. All packets were logged using tshark (part of the Wireshark tools).

Then, the evaluator modified the BigFix Server database to change the URL associated with the “BES Support” external site to the URL of the local TLS server. Additionally, the BigFix Server “ca-bundle.crt” certificate bundle was modified to include the root CA of a certificate chain consisting of four certificates (one root CA, two intermediate CAs, and one server node certificate).

Finally, the evaluator instructed BigFix Server to initiate a TLS request to the OpenSSL TLS server. Depending on the specific test, the connection succeeded or failed:

Test 1: The evaluator created a valid certificate chain. Then, the evaluator started a local OCSP server and configured the server node certificate to be marked as “valid”. The evaluator verified the connection was accepted as expected. Finally, the tester stopped the local OCSP server and verified the connection was rejected as expected (as the TOE could not reach the OCSP server).

Test 2: The evaluator created a valid certificate chain. Then, the evaluator started a local OCSP server and configured the server node certificate to be marked as “revoked”. The evaluator verified the connection was rejected as expected.

2.2.4 Security management (FMT)

2.2.4.1 Secure by Default Configuration (FMT_CFG_EXT.1)

2.2.4.1.1 FMT_CFG_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-ASPP-FMT_CFG_EXT.1.1-ASE-01

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

Summary

The evaluator verified that Section 7.1.4.1 *FMT_CFG_EXT.1* in the TSS of [ST] states that the TOE does not include default credentials; credentials are configured as part of installation.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FMT_CFG_EXT.1.1-ATE-01

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

- *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.*

- *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.*
- *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.*

Summary

This assurance activity is not applicable, as BigFix Server does not use any default credentials.

2.2.4.1.2 FMT_CFG_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FMT_CFG_EXT.1.2-ATE-01

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.

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.

Platform: 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.

Platform: iOS

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

Platform: 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.

Platform: 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.

Platform: 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.

Summary

The evaluator first determined the directories where BigFix Server creates and stores executables and data files. Then, for each directory, the evaluator verified that the owner of the directory is the Administrators group. Moreover, only the directory owner, the TrustedInstaller group, the SYSTEM group, and the Administrators group have “Full control” or “Modify” access to the directory. By definition, a standard unprivileged user is not a member of these privileged groups.

2.2.4.2 Supported Configuration Mechanism (FMT_MEC_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FMT_MEC_EXT.1-ASE-01

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

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

Summary

The evaluator verified that Section 7.1.4.2 *FMT_MEC_EXT.1* in the TSS of [ST] lists the configuration options related to the security functionality. All options are stored in the Windows Registry, which is a mechanism supported by the platform. BigFix Server does not use the PP-Module for File Encryption.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FMT_MEC_EXT.1-ATE-01

[TD0893] 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:

Platform: 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.

Platform: 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/ or https://learn.microsoft.com/en-us/previous-versions/dotnet/netframework-4.0/zzdt0e7f(v=vs.100) or https://learn.microsoft.com/en-us/aspnet/core/fundamentals/configuration/ or https://learn.microsoft.com/en-us/aspnet/core/host-and-deploy/iis/web-config 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.

Platform: iOS

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

Platform: 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).

Platform: 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).

Platform: macOS

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

Summary

BigFix Server is not a Windows Universal Application, nor a .NET application. Consequently, the evaluator used ProcMon to verify the configuration location. The evaluator first set the appropriate ProcMon filters (RegQueryValue and RegSetValue), then used BigFix Console to perform various configuration changes. The evaluator verified that ProcMon showed the RegQueryValue and RegSetValue operations are used to query and save the configuration.

2.2.4.3 Specification of Management Functions (FMT_SMF.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FMT_SMF.1-AGD-01

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.

Summary

The evaluator verified that the management functions selected in Section 6.1.4.3 *FMT_SMF.1 Specification of Management Functions* of [ST] are described in Section 4.2 of [CCGUIDE]. The evaluator confirmed [CCGUIDE] contains sufficient information to perform the management duties associated with the management function.

Test Assurance Activities

Assurance Activity AA-ASPP-FMT_SMF.1-ATE-01

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.

Summary

For each management function, the evaluator verified that BigFix Server implements the documented functionality:

Create Console Operators: the evaluator used BigFix Console to create a new Console Operators named “test” and verified that it was possible to log in to BigFix Console using the newly created Console Operators.

Change the password of the logged-in Console Operator: the evaluator used BigFix Console to log in as the Master Operator and verified that it could change both the password of the “test” Console Operator as well as its own password.

Obtain TOE name and version: this was verified as part of FPT_TUD_EXT.1.2.

Verify updates for the TOE: this was verified as part of FPT_TUD_EXT.1.1.

2.2.5 Privacy (FPR)

2.2.5.1 User Consent for Transmission of Personally Identifiable Information (FPR_ANO_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FPR_ANO_EXT.1-ASE-01

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

Summary

The evaluator verified that Section 7.1.5.1 *FPR_ANO_EXT.1* in the TSS of [ST] states that the TOE does not contain any functionality related to PII.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPR_ANO_EXT.1-ATE-01

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.

Summary

This assurance activity is not applicable, as BigFix Server does not contain any functionality that relates to PII.

2.2.6 Protection of the TSF (FPT)

2.2.6.1 Anti-Exploitation Capabilities (FPT_AEX_EXT.1)

2.2.6.1.1 FPT_AEX_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-ASPP-FPT_AEX_EXT.1.1-ASE-01

The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled. If any explicitly-mapped exceptions are claimed, the evaluator shall check that the TSS identifies these exceptions, describes the static memory mapping that is used, and provides justification for why static memory mapping is appropriate in this case.

Summary

The evaluator verified that Section 7.1.6.1 *FPT_AEX_EXT.1 (Anti-Exploitation Capabilities)* in the TSS of [ST] identifies two compiler and linker flags when the TOE is built. One of these flags is /DYNAMICBASE, which enables address space layout randomization (ASLR). Moreover, the TSS states that there are no explicitly-mapped exceptions and no static memory mappings used during the building of the TOE.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_AEX_EXT.1.1-ATE-01

The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address except for any exceptions claimed in the SFR. For these exceptions, the evaluator shall verify that this analysis shows explicit mappings that are consistent with what is claimed in the TSS. . 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.

Platform: 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.

Platform: 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.

Platform: 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.

Platform: 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.

Platform: 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.

Platform: 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.

Summary

The evaluator verified the memory mappings for BESRootServer.exe (the TOE application executable) are different when the application is reinstalled. The evaluator first used VMMap to copy all memory mappings of the running BESRootServer.exe process to a file. Then, the evaluator reinstalled BigFix Server and rebooted the Microsoft Windows server. Finally, the evaluator again collected the memory mappings using VMMap and confirmed the mappings were different.

The evaluator also used Microsoft BinSkim to analyze BESRootServer.exe. The BinSkim output “pass BA2009: 'BESRootServer.exe” confirmed to the evaluator that the application has ASLR enabled.

2.2.6.1.2 FPT_AEX_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_AEX_EXT.1.2-ATE-01

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

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.*

Platform: 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.

Platform: iOS

The evaluator shall perform static analysis on the application to verify that mprotect is never invoked with the

PROT_EXEC permission.

Platform: 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.*

Platform: 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.*

Platform: macOS

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

Summary

The evaluator used Microsoft BinSkim to analyze BESRootServer.exe (the TOE application executable). The BinSkim output “notapplicable BA2016: 'BESRootServer.exe' was not evaluated for check 'MarkImageAsNXCompatible' as the analysis is not relevant based on observed metadata: image is a 64-bit binary” confirmed to the evaluator that the application is a 64-bit binary with DEP protections enabled by default.

2.2.6.1.3 FPT_AEX_EXT.1.3

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_AEX_EXT.1.3-ATE-01

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

Platform: Android

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

Platform: Windows

[TD0823] 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://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/enable-exploit-protection?view=o365-worldwide.

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).

Platform: iOS

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

Platform: Linux

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

Platform: Solaris

The evaluator shall ensure that the application can run with Solaris Trusted Extensions enabled and enforcing.

Platform: macOS

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

Summary

The evaluator first used Get-ProcessMitigation to verify which mitigations are enabled by default on BESRootServer.exe (the TOE application executable). Then, the evaluator used Set-ProcessMitigation to enable the Control Flow Guard, Bottom-Up ASLR, EAF, IAF, and Data Execution Prevention mitigations. Finally, the evaluator used Get-ProcessMitigation again to verify the required mitigations are enabled. The evaluator also verified the BigFix Server can run correctly using the specified mitigations.

2.2.6.1.4 FPT_AEX_EXT.1.4

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_AEX_EXT.1.4-ATE-01

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:

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.

Platform: 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.

Platform: iOS

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

Platform: 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.

Platform: 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.

Platform: 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.

Summary

The evaluator first determined the directories containing the BigFix Server executables. Then, the evaluator reinstalled BigFix Server and noted down all files in the relevant directories. Finally, the evaluator ran BigFix Server, mimicking normal usage, and stopped BigFix Server again. The evaluator verified that only two files are modified or added in the relevant directories: BESRelay.log and server_audit.log. These files are not user-modifiable as they contain log/audit data.

2.2.6.1.5 FPT_AEX_EXT.1.5

TSS Assurance Activities

Assurance Activity AA-ASPP-FPT_AEX_EXT.1.5-ASE-01

[TD0815] (Conditional: The PE or ELF automated tests fail) The evaluator shall ensure that the TSS describes the stack-based buffer overflow compiler flags.

Summary

The PE or ELF automated tests did not fail. Nevertheless, the evaluator verified that Section 7.1.6.1 *FPT_AEX_EXT.1 (Anti-Exploitation Capabilities)* in the TSS of [ST] identifies two compiler and linker flags when the TOE is built. One of these flags is /GS, which enables buffer security checks.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_AEX_EXT.1.5-ATE-01

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

Platform: Windows *[TD0815] 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, BinSkim, 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.* *[TD0815] If these automated tests fail, the evaluator shall perform the above, conditional TSS activity.*

Summary

The evaluator used Microsoft BinSkim to analyze BESRootServer.exe (the TOE application executable). The BinSkim output “pass BA2011: 'BESRootServer.exe' is a C or C++ binary built with the stack protector buffer security feature enabled for all modules, making it more difficult for an attacker to exploit stack buffer overflow memory corruption vulnerabilities.” confirmed to the evaluator that the application has GS enabled.

The evaluator also used cande (Canary Detector) to verify the presence of the stack canary instructions in BESRootServer.exe.

2.2.6.2 Use of Supported Services and APIs (FPT_API_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FPT_API_EXT.1-ASE-01

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

Summary

The evaluator verified that Section 7.1.6.2 *FPT_API_EXT.1* in the TSS of [ST] lists all API functions provided by the Windows platform used in the application.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_API_EXT.1-ATE-01

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.

Summary

For each of the API functions listed in the TSS of [ST], the evaluator verified it is documented and supported on the Microsoft Learn website: https://learn.microsoft.com/.

2.2.6.3 Software Identification and Versions (FPT_IDV_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FPT_IDV_EXT.1-ASE-01

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

Summary

This assurance activity is not applicable, as Section 6.1.6.3 *FPT_IDV_EXT.1 Software Identification and Versions* of [ST] selects “SWID tags that comply with minimum requirements from ISO/IEC 19770-2:2015”.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_IDV_EXT.1-ATE-01

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.

Summary

The evaluator verified the hcltechsw.com-BigFix_Platform_Server-11.0.3.82.swidtag file was present in the application’s installation directory. The swidtag file contained a SoftwareIdentity element with name “BigFix Platform Server” and version “11.0.3.82”, and an Entity element with name “HCL”.

2.2.6.4 Use of Third Party Libraries (FPT_LIB_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_LIB_EXT.1-ATE-01

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.

Summary

The evaluator used PowerShell to recursively list all dynamic link libraries (DLLs) in the BigFix Server installation directory. The evaluator verified that only the fips.dll file (corresponding to the OpenSSL FIPS Provider 3.0.8) was found.

Note that Section 6.1.6.3 *FPT_LIB_EXT.1 Use of Third Party Libraries* of [ST] lists many more libraries. Those libraries are statically linked (compiled into the application) and therefore not present as DLL files in the installation directory.

2.2.6.5 Integrity for Installation and Update (FPT_TUD_EXT.1)

2.2.6.5.1 FPT_TUD_EXT.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.1.1-AGD-01

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

Summary

The evaluator verified that Section 4.2.4 and Section 4.3 of [CCGUIDE] describe respectively how updates can be checked and how updates are performed. Updates are installed as a new version of BigFix Server, so the instructions in Section 2 of [CCGUIDE] are applicable.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.1.1-ATE-01

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.

Summary

As described in the documentation, the evaluator used BigFix Console to list the Upgrade Fixlets for the BigFix Server 11.0. The evaluator found that the most recently published version of BigFix Server was displayed, which was also the installed version. Therefore, no update was available.

2.2.6.5.2 FPT_TUD_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.1.2-AGD-01

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

Summary

The evaluator verified that Section 4.2.3 of [CCGUIDE] describes how to query the current version of BigFix Server.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.1.2-ATE-01

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.

Summary

As described in the documentation, the evaluator opened the server_audit.log file to verify the current version of the software. The evaluator verified that the file contained a line starting with “BES Root Server version 11.0.3.82”, which was also the installed version.

2.2.6.5.3 FPT_TUD_EXT.1.3

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.1.3-ATE-01

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

Platform: 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.*

Summary

The evaluator used PowerShell to recursively list all executable files (EXEs) in the BigFix Server installation directory and their SHA-256 hash values. The evaluator then ran BigFix Server, mimicking normal usage, and stopped BigFix Server again. Finally, the evaluator again recursively listed the EXE files and their SHA-256 hash values. The evaluator confirmed that the hash values were identical.

2.2.6.5.4 FPT_TUD_EXT.1.4

TSS Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.1.4-ASE-01

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.

Summary

The evaluator verified that Section 7.1.6.5 *FPT_TUD_EXT.1* in the TSS of [ST] states that the TOE and its updates are distributed as an EXE file signed using Microsoft Authenticode by the authorized source, HCL America Inc.. The TOE and its updates can be downloaded from the BigFix Enterprise Suite Download Center.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.6.5.5 FPT_TUD_EXT.1.5

TSS Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.1.5-ASE-01

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.

Summary

The evaluator verified that Section 7.1.6.5 *FPT_TUD_EXT.1* in the TSS of [ST] states that the TOE and its updates are distributed as an EXE file signed using Microsoft Authenticode by the authorized source, HCL America Inc. The evaluator also performed the tests in FPT_TUD_EXT.2, since “as an additional package” is selected in 6.1.6.5 *FPT_TUD_EXT.1 Integrity for Installation and Update* of [ST].

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.6.6 Integrity for Installation and Update (FPT_TUD_EXT.2)

2.2.6.6.1 FPT_TUD_EXT.2.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.2.1-ATE-01

The evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:

Platforms: Android

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

Platforms: 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.

Platforms: iOS

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

Platforms: 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.

Platforms: Solaris

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

Platforms: macOS

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

Summary

The evaluator downloaded the latest version of HCL BigFix Server version 11.0.3 from the BigFix Enterprise Suite Download Center and verified the file is an EXE file. By inspecting the properties of the file, the evaluator confirmed the package was digitally signed using Microsoft Authenticode by HCL America Inc.

2.2.6.6.2 FPT_TUD_EXT.2.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.2.2-ATE-01

Platforms: Android

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

Platforms: Apple iOS

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

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..

Summary

Because of the large amount of files present on a typical Windows system, it is infeasible to list every single file on the entire file system. Instead, the evaluator limited the listing to the C:\Users\Administrator\AppData and C:\Program Files (x86), as those will store the files created by BigFix Server.

The evaluator installed BigFix Server using the installation package downloaded from the BigFix Enterprise Suite Download Center. After installation, the evaluator confirmed BigFix Server was operational. Then, the evaluator uninstalled BigFix Server using the procedures described in the operational guidance and rebooted the machine. Finally, the evaluator listed the files again in the two aforementioned directories.

The evaluator verified that only audit/log files were added.

2.2.6.6.3 FPT_TUD_EXT.2.3

TSS Assurance Activities

Assurance Activity AA-ASPP-FPT_TUD_EXT.2.3-ASE-01

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.

Summary

The evaluator verified that Section 7.1.6.6 *FPT_TUD_EXT.2* in the TSS of [ST] refers to Section 7.1.6.5 *FPT_TUD_EXT.1*, which states that the TOE and its updates are distributed as an EXE file signed using Microsoft Authenticode. The authorized source is defined as HCL America Inc.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.7 Trusted path/channels (FTP)

2.2.7.1 Protection of Data in Transit (FTP_DIT_EXT.1)

TSS Assurance Activities

Assurance Activity AA-ASPP-FTP_DIT_EXT.1-ASE-01

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

Summary

The evaluator verified that the TOE does not use platform-provided functionality to protect data in transit. Section 7.1.6.6 *FPT_TUD_EXT.2* in the TSS of [ST] states that the TOE implements trusted channels using OpenSSL and cURL, both part of the TOE.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-ASPP-FTP_DIT_EXT.1-ATE-01

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.*
- *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.*
- *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.*

Platforms: 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.

Platforms: 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.

Summary

Test 1: The evaluator used Wireshark to sniff the network traffic on the wired network interface (NIC1). While Wireshark was running, the evaluator mimicked normal usage of the TOE using BigFix Console. The evaluator verified that all network communications associated with BESRootServer.exe is encrypted using HTTPS or TLS, as specified in [ST].

Test 2: BigFix Server encrypts all transmitted data, including sensitive data, using HTTPS. Therefore, this test is covered by Test 1.

Test 3: BigFix Server does not transmit user credentials; user credentials are only stored locally on the machine. Therefore, this test is not applicable.

2.3 Security Assurance Requirements

2.3.1 Development (ADV)

2.3.1.1 Basic functional specification (ADV_FSP.1)

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 5.1 Security Functional Requirements, 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.

2.3.2 Guidance documents (AGD)

2.3.2.1 Operational user guidance (AGD_OPE.1)

Assurance Activity AA-ASPP-AGD_OPE.1-AGD-01

Some of the contents of the operational guidance will be verified by the evaluation activities in Section 5.1 Security Functional Requirements 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.*

Summary

The evaluator reviewed the operational guidance and found that the required instructions are present:

- There is no configuration required to configure the cryptographic engine. The default configuration is suitable for the Common Criteria evaluated configuration.
- The process of verifying the digital signature is described in Section 2.1 of [CCGUIDE]: the binaries of the TOE are signed using Microsoft Authenticode.
- Instructions for obtaining the update are described in Section 2.1 of [CCGUIDE].
- Instructions for initiating the update process are described in Section 2.2 and Section 2.3 of [CCGUIDE].

2.3.2.2 Preparative procedures (AGD_PRE.1)

Assurance Activity AA-ASPP-AGD_PRE.1-AGD-01

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.

Summary

The evaluator verified that Section 1.3 of [CCGUIDE] explains the platforms claimed for BigFix Server, and these platforms match those listed in [ST]. The instructions in Section 2 of [CCGUIDE] are adequate to address the platforms, to install BigFix Server according to the evaluated configuration.

2.3.3 Life-cycle support (ALC)

2.3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activity AA-ASPP-ALC_CMC.1-ALC-01

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.

Summary

The evaluator verified that Section 1.2 of [ST] and [CCGUIDE] identify the TOE identically as “HCL BigFix Server version 11.0.3”. The evaluator was able to download BigFix Server from the vendor web site (https://support.bigfix.com/bes/install/downloadbes.html) and found that it matches the TOE version in [ST] and [CCGUIDE].

2.3.3.2 TOE CM coverage (ALC_CMS.1)

Assurance Activity AA-ASPP-ALC_CMS.1-ALC-01

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.

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.

Summary

The evaluator verified that the developer uses the Microsoft Visual Studio development environment, which is appropriate to develop applications targeting the Microsoft Windows platform. The Visual Studio compiler toolchain can be configured to enable buffer overflow protection mechanisms using the /DYNAMICBASE and /GS flags (also documented in [ST]). These protections are enabled by default on the compiled binaries and libraries.

The TOE is uniquely identified, as explained above for ALC_CMC.1.

2.3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activity AA-ASPP-ALC_TSU_EXT.1-ALC-01

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.

Summary

The evaluator verified that the "ALC_TSU_EXT.1" section in the TSS of [ST] describes the process used by HCL to create and deploy security updates. HCL uses CVSS version 3.1 to prioritize vulnerabilities. Deployment is performed as part of regular BigFix product patch releases. HCL provides fixes and/or mitigations within 180 days.

This section also includes a link to a web portal (protected using TLS) to report security issues pertaining to the TOE.

2.3.4 Tests (ATE)

2.3.4.1 Independent testing - conformance (ATE_IND.1)

Assurance Activity AA-ASPP-ATE_IND.1-ATE-01

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.

Summary

The evaluators prepared a Detailed Test Report (DTR) to specify all the tests covering the assurance activities in this evaluation. The DTR, which contains test requirements i.e., test assurance activities), test procedures, expected test results, actual test results, along with the verdict, is provided to the validation body but is not published.

The test environment was set up according to a setup strategy that followed the evaluated configuration requirements specified in the guidance [CCGUIDE] and Security Target [ST], supplemented by configurations required to perform testing. The testing was performed on all platforms claimed in [ST]. The following tools were used for testing:

- Dell PowerEdge R430, running the TOE (version 11.0.3.82)
- BigFix Console version 11.0.3
- MSys2 version 20240113 with the following packages:
  - MinGW64 tool chain, including: gcc, make, patch, etc.
  - OpenSSL 3.2.0 or newer
- SQL Server Management Studio version 20.1
- Wireshark version 4.2.4 (including tshark)
- Process Monitor v3.95
- VMMap v3.4
- cande (https://github.com/commoncriteria/canary-detector)
- Proprietary test scripts developed by the CCTL

There is no configuration required to configure the cryptographic engine. The default configuration is suitable for the Common Criteria evaluated configuration.

2.3.5 Vulnerability assessment (AVA)

2.3.5.1 Vulnerability survey (AVA_VAN.1)

Assurance Activity AA-ASPP-AVA_VAN.1-AVA-01

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.

Summary

The evaluator searched for publicly known vulnerabilities applicable to HCL BigFix Server Version 11.0.3 using the following sources:

- MITRE Common Vulnerabilities and Exposures (CVE) List
- National Vulnerability Database
- CISA Known Exploited Vulnerabilities Catalog
- OpenSSL Vulnerabilities (https://openssl-library.org/news/vulnerabilities-3.0/ and https://openssl-library.org/news/vulnerabilities-3.2/)

Additionally, vendor-provided security bulletins and vulnerability databases were examined. The evaluator searched for TOE components (including software and hardware dependencies), the operational environment, as well as relevant technical terms identified in the PPs for which the ST claims conformance.

The evaluator found no vulnerabilities that impact the TSF and are not mitigated.

Subsequently, the evaluator performed a generic port scan on the TOE to search for any undocumented open network ports. The evaluator found that no ports are unexpectedly open. In other words: all ports are either expected to be open as part of the operational environment, or are associated with the TOE and publicly documented as such.

Finally, as required by the assurance activity, the evaluator ran a virus scan against the TOE directory C:\Program Files (x86)\BigFix Enterprise\BES Server. No threats were found (i.e., no files were flagged as malicious).

A Appendixes

A.1. References

CC	**Common Criteria for Information Technology Security Evaluation**	
	Version	3.1R5
	Date	April 2017
	Location	http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
	Location	http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf
	Location	http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf
CCEVS-LG	**CCEVS LabGrams**	
	Author(s)	NIAP
	Date	May 2025
	Location	https://www.niap-ccevs.org/labgrams
CCEVS-PL	**CCEVS Scheme Policy Letters**	
	Author(s)	NIAP
	Date	May 2025
	Location	https://www.niap-ccevs.org/policies
CCEVS-PUB	**CCEVS Scheme Publications**	
	Author(s)	NIAP
	Date	May 2025
	Location	https://www.niap-ccevs.org/publications
CCEVS-TD	**Technical Decisions**	
	Author(s)	NIAP
	Date	May 2025
	Location	https://www.niap-ccevs.org/technical-decisions/
CCGUIDE	**HCL BigFix Server Version 11.0.3 Common Criteria Configuration Guide**	
	Author(s)	atsec information security corporation
	Version	1.0
	Date	2025-02-07
CEM	**Common Methodology for Information Technology Security Evaluation**	
	Version	3.1R5
	Date	April 2017
	Location	http://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R5.pdf
PKG_TLS_V1.1	**Functional Package for Transport Layer Security (TLS)**	
	Author(s)	NIAP
	Version	1.1
	Date	2019-03-01
PP_APP_V1.4	**Protection Profile for Application Software**	
	Author(s)	NIAP
	Version	1.4
	Date	2021-10-12
ST	**HCL BigFix Server Version 11.0.3 Security Target**	
	Author(s)	atsec information security corporation
	Version	1.2
	Date	2025-05-06

A.2. Glossary

Augmentation

The addition of one or more requirement(s) to a package.

Authentication data

Information used to verify the claimed identity of a user.

Authorised user

A user who may, in accordance with the SFRs, perform an operation.

Class

A grouping of CC families that share a common focus.

Component

The smallest selectable set of elements on which requirements may be based.

Connectivity

The property of the TOE which allows interaction with IT entities external to the TOE. This includes exchange of data by wire or by wireless means, over any distance in any environment or configuration.

Dependency

A relationship between components such that if a requirement based on the depending component is included in a PP, ST or package, a requirement based on the component that is depended upon must normally also be included in the PP, ST or package.

Deterministic RNG (DRNG)

An RNG that produces random numbers by applying a deterministic algorithm to a randomly selected seed and, possibly, on additional external inputs.

Element

An indivisible statement of security need.

Entropy

The entropy of a random variable X is a mathematical measure of the amount of information gained by an observation of X.

Evaluation

Assessment of a PP, an ST or a TOE, against defined criteria.

Evaluation Assurance Level (EAL)

An assurance package, consisting of assurance requirements drawn from CC Part 3, representing a point on the CC predefined assurance scale.

Evaluation authority

A body that implements the CC for a specific community by means of an evaluation scheme and thereby sets the standards and monitors the quality of evaluations conducted by bodies within that community.

Evaluation scheme

The administrative and regulatory framework under which the CC is applied by an evaluation authority within a specific community.

Exact conformance

a subset of Strict Conformance as defined by the CC, is defined as the ST containing all of the requirements in the Security Requirements section of the PP, and potentially requirements from Appendices of the PP. While iteration is allowed, no additional requirements (from the CC parts 2 or 3) are allowed to be included in the ST. Further, no requirements in the Security Requirements section of the PP are allowed to be omitted.

Extension

The addition to an ST or PP of functional requirements not contained in Part 2 and/or assurance requirements not contained in Part 3 of the CC.

External entity

Any entity (human or IT) outside the TOE that interacts (or may interact) with the TOE.

Family

A grouping of components that share a similar goal but may differ in emphasis or rigour.

Formal

Expressed in a restricted syntax language with defined semantics based on well-established mathematical concepts.

Guidance documentation

Documentation that describes the delivery, preparation, operation, management and/or use of the TOE.

Identity

A representation (e.g. a string) uniquely identifying an authorised user, which can either be the full or abbreviated name of that user or a pseudonym.

Informal

Expressed in natural language.

Object

A passive entity in the TOE, that contains or receives information, and upon which subjects perform operations.

Operation (on a component of the CC)

Modifying or repeating that component. Allowed operations on components are assignment, iteration, refinement and selection.

Operation (on an object)

A specific type of action performed by a subject on an object.

Operational environment

The environment in which the TOE is operated.

Organisational Security Policy (OSP)

A set of security rules, procedures, or guidelines imposed (or presumed to be imposed) now and/or in the future by an actual or hypothetical organisation in the operational environment.

Package

A named set of either functional or assurance requirements (e.g. EAL 3).

PP evaluation

Assessment of a PP against defined criteria.

Protection Profile (PP)

An implementation-independent statement of security needs for a TOE type.

Random number generator (RNG)

A group of components or an algorithm that outputs sequences of discrete values (usually represented as bit strings).

Refinement

The addition of details to a component.

Role

A predefined set of rules establishing the allowed interactions between a user and the TOE.

Secret

Information that must be known only to authorised users and/or the TSF in order to enforce a specific SFP.

Secure state

A state in which the TSF data are consistent and the TSF continues correct enforcement of the SFRs.

Security attribute

A property of subjects, users (including external IT products), objects, information, sessions and/or resources that is used in defining the SFRs and whose values are used in enforcing the SFRs.

Security Function Policy (SFP)

A set of rules describing specific security behaviour enforced by the TSF and expressible as a set of SFRs.

Security objective

A statement of intent to counter identified threats and/or satisfy identified organisation security policies and/or assumptions.

Security Target (ST)

An implementation-dependent statement of security needs for a specific identified TOE.

Seed

Value used to initialize the internal state of an RNG.

Selection

The specification of one or more items from a list in a component.

Semiformal

Expressed in a restricted syntax language with defined semantics.

ST evaluation

Assessment of an ST against defined criteria.

Subject

An active entity in the TOE that performs operations on objects.

Target of Evaluation (TOE)

A set of software, firmware and/or hardware possibly accompanied by guidance.

TOE evaluation

Assessment of a TOE against defined criteria.

TOE resource

Anything useable or consumable in the TOE.

TOE Security Functionality (TSF)

A set consisting of all hardware, software, and firmware of the TOE that must be relied upon for the correct enforcement of the SFRs.

Transfers outside of the TOE

TSF mediated communication of data to entities not under control of the TSF.

True RNG (TRNG)

A device or mechanism for which the output values depend on some unpredictable source (noise source, entropy source) that produces entropy.

Trusted channel

A means by which a TSF and a remote trusted IT product can communicate with necessary confidence.

Trusted path

A means by which a user and a TSF can communicate with necessary confidence.

TSF data

Data created by and for the TOE, that might affect the operation of the TOE.

TSF Interface (TSFI)

A means by which external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF.

User

See external entity

User data

Data created by and for the user, that does not affect the operation of the TSF.