Assurance Activity Report

Version:	2.1
Date:	2019-01-30
Status:	RELEASED
Classification:	Public
Filename:	CSEC2017014_AAR_190130_v2.1
Product:	Oracle Linux 7.3
Sponsor:	Oracle America, Inc.
Evaluation Facility:	atsec information security AB
Certification ID:	CSEC2017014
Certification Body:	CSEC
Author(s):	Rasma Araby
Quality Assurance:	Trang Huynh

atsec information security AB
Svärdvägen 3C
S-182 33 Danderyd

Phone: +46-8-55 110 400
Fax: +46-8-55 110 401
www.atsec.com

(Evaluation facility accredited by SWEDAC according to ISO/IEC 17025 with accreditation number 1937)

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	Application Notes
0.1	2019-01-09	Rasma Araby	First version	
0.2	2019-01-09	Rasma Araby	Addressed QA comments	
1.0	2019-01-11	Rasma Araby	Ready to be submitted to the certifier	
1.1	2019-02-23	Rasma Araby	Updated the reference to the public ST	
2.0	2019-01-29	Rasma Araby	Updated reference to the updated ST to address certifier comment	
2.1	2019-01-30	Rasma Araby	Updated reference to the updated CI list	

Table of Contents

List of Tables

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 following extended methodologies: "CEM extension for the Protection Profile for General Purpose Operating Systems (OSPP) v4.1" [CEM-OSPPv4.1], "CEM extension for the Extended Package for Secure Shell (SSH) v1.0" [CEM-SSHEPv1.0] and "CC and CEM addenda - Exact Conformance, Selection-Based SFRs, Optional SFRs" [CCDB-2017-05-17], as specified in the Security Target [ST].

The following scheme documents and interpretations have been considered:

- "SP-002 Evaluation and Certification, CSEC", version 26.0 as of 2017-06-22.
- "SP-188 Scheme Crypto Policy, CSEC", version 7.0 as of 2017-04-14.
- [CCEVS-TD0104]: "FMT_SMF and FMT_MOF in OS PP", version as of 2016-09-16.
- [CCEVS-TD0107]: "FCS_CKM - ANSI X9.31-1998, Section 4.1.for Cryptographic Key Generation", version as of 2016-09-14.
- [CCEVS-TD0163]: "Update to FCS_TLSC_EXT.1.1 Test 5.4 and FCS_TLSS_EXT.1.1 Test", version as of 2017-04-05.
- [CCEVS-TD0208]: "Remote Users in OSPP", version as of 2017-06-09.
- [CCEVS-TD0239]: "Cryptographic Key Destruction in OS PP", version as of 2017-09-22.
- [CCEVS-TD0240]: "FCS_COP.1.1(1) Platform provided crypto for encryption/decryption", version as of 2017-11-27.
- [CCEVS-TD0243]: "SSH Key-Based Authentication", version as of 2017-10-03.
- [CCEVS-TD0244]: "FCS_TLSC_EXT - TLS Client Curves Allowed", version as of 2017-11-16.
- [CCEVS-TD0246]: "Assurance Activity for FIA_UAU.5.2", version as of 2017-10-31.
- [CCEVS-TD0304]: "Update to FCS_TLSC_EXT.1.2", version as of 2018-04-04.
- [CCEVS-TD0305]: "Handling of TLS connections with and without mutual authentication", version as of 2018-04-04.

2 Evaluation Results

The following table summarizes the Cryptographic Algorithm Validation Program (CAVP) certificates that apply to the TOE, including specific standards, options, and implementations for each algorithm of each SFR, and the applicable CAVP certificate for each.

Implementation abbreviations:

KCAPI
Oracle Linux 7 Kernel Cryptographic (KCAPI) Module R7-2.0.0

libgcrypt
Oracle Linux 7 libgcrypt Cryptographic Module R7-2.0.0

NSS
Oracle Linux 7 NSS Cryptographic Module R7-2.0.0

OpenSSH
Oracle Linux OpenSSH Cryptographic Module R7-2.0.0

OpenSSL
Oracle Linux OpenSSL Cryptographic Module R7-2.0.0

UEK
Oracle Linux 7 Unbreakable Enterprise Kernel (UEK) Module R7-2.0.0

Operational environment:

- Intel® Xeon® Silver 4114 w/ Oracle Linux 7.3 64 bit

The table below contains links to CAVP certificates which may break or become stale over time. It is better to go to the CAVP website to search for the certificates directly.

Table 1: Oracle OEL

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
dm_crypt	KCAPI (aesni)	AES-CBC-128, AES-CBC-256, AES-XTS-128, AES-XTS-256 [FIPS PUB 197 (AES), SP800-38A (CBC), SP800-38E (XTS)]	FCS_COP.1(1)	AES #5409
	KCAPI (shaavx2)	SHA-1, SHA2-256, SHA2-384, SHA2-512 [FIPS PUB 180-4 (SHS)]	FCS_COP.1(2)	SHS #4341
	UEK (aesni)	AES-CBC-128, AES-CBC-256, AES-XTS-128, AES-XTS-256 [FIPS PUB 197 (AES),	FCS_COP.1(1)	AES #5402

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
		SP800-38A (CBC), SP800-38E (XTS)]		
	UEK (shaavx2)	SHA-1, SHA2-256, SHA2-384, SHA2-512 [FIPS PUB 180-4 (SHS)]	FCS_COP.1(2)	SHS #4331
PBKDF2 for KEK	libgcrypt	SHA-1, SHA2-256, SHA2-384, SHA2-512 [FIPS PUB 180-4 (SHS)]	FCS_COP.1(2)	SHS #4217
		HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 Prerequisites: SHS #4217 [FIPS Pub 198-1 (HMAC), FIPS PUB 180-4 (SHS)]	FCS_COP.1(4)	HMAC #3469
		HMAC_DRBG(any) Prerequisites: HMAC #3469 [SP800-90A (DRBG)]	FCS_RBG_EXT.1	DRBG #2003
SSH-2 (OpenSSH) and RSA/ECDSA key pair generation (openssl command)	OpenSSL (AESNI, SHA1 AVX, SHA2 ASM)	RSA KeyGen Modulo: 2048, 3072 Probable Random Primes (B.3.3) Primality Test: C.2 Prerequisites: SHS #4312 DRBG #2079 [FIPS PUB 186-4]	FCS_CKM.1(1)	RSA #2873
		ECDSA KeyGen Curves: P-256, P-384, P-521 Prerequisites: SHS #4312 DRBG #2079 [FIPS PUB 186-4]	FCS_CKM.1(1)	ECDSA #1417

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
		RSA Key Establishment [SP800-56B]	FCS_CKM.2(1)	No CAVP test exists. Vendor Affirmed. (See RSA #2873)
		DSA KeyGen (for DH key establishment only) L: 2048, 3072 Prerequisites: SHS #4312 DRBG #2079 [FIPS PUB 186-4]	FCS_CKM.2(1)	DSA #1388
		KAS-FFC DH (dhEphem) FB: SHA: SHA2-224 FC: SHA: SHA2-224 Prerequisites: DSA #1388 SHS #4312 DRBG #2079 [SP800-56A]	FCS_CKM.2(1)	Component #1837
		KAS-ECC ECDH (Ephemeral Unified) EC: Hash Algorithm: SHA2-256 Curve: P-256 ED: Hash Algorithm: SHA2-384 Curve: P-384 EE: Hash Algorithm: SHA2-512 Curve: P-521 Prerequisites: ECDSA #1417 SHS #4312 DRBG #2079 [SP800-56A]	FCS_CKM.2(1)	Component #1837
		AES-CTR-128,	FCS_COP.1(1)	AES #5370

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
		AES-CTR-256, AES-CBC-128, AES-CBC-256, AES-GCM-128, AES-GCM-256 [FIPS PUB 197 (AES), SP800-38A (CBC, CTR), SP800-38D (GCM)]		
		SHA-1, SHA2-256, SHA2-384, SHA2-512 [FIPS PUB 180-4 (SHS)]	FCS_COP.1(2)	SHS #4312
		RSA SigGen Signature: PKCS 1.5 Module: 2048 Hashes: SHA2-256, SHA2-384, SHA2-512 Modulo: 3072 Hashes: SHA2-256, SHA2-384, SHA2-512 RSA SigVer Signature: PKCS 1.5 Module: 2048 Hashes: SHA-1, SHA2-256, SHA2-384, SHA2-512 Modulo: 3072 Hashes: SHA-1, SHA2-256, SHA2-384, SHA2-512 Prerequisites: SHS #4312 DRBG #2079 [FIPS PUB 186-4]	FCS_COP.1(3)	RSA #2873
		ECDSA SigGen Curves: P-256, P-384, P-521 Hashes: SHA2-256, SHA2-384, SHA2-512 ECDSA SigVer Curves: P-256, P-384, P-521 Hashes: SHA-1, SHA2-256, SHA2-384, SHA2-512 Prerequisites: SHS #4312	FCS_COP.1(3)	ECDSA #1417

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
		DRBG #2079 [FIPS PUB 186-4]		
		HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 [FIPS Pub 198-1 (HMAC), FIPS PUB 180-4 (SHS)]	FCS_COP.1(4)	HMAC #3558
		CTR_DRBG(AES) Prerequisites: AES #5370 [SP800-90A]	FCS_RBG_EXT.1	DRBG #2079
	OpenSSH	SSHv2 KDF Hash: SHA-1, SHA2-256, SHA2-384, SHA2-512 Prerequisites: SHS #4312 (from OpenSSL) [SP800-135 (KDF)]	Not defined in the PP/EP, but listed in an ST application note for FCS_CKM.1(2).	Component #1870
TLS	NSS (with AESNI)	AES-128-CBC, AES-256-CBC, AES-128-GCM, AES-256-GCM [FIPS PUB 197 (AES), SP800-38A (CBC), SP800-38D (GCM)]	FCS_COP.1(1)	AES #5654
	NSS (without AESNI)	DSA KeyGen L: 2048, 3072 Prerequisites: SHS #4535 DRBG #2284 [FIPS PUB 186-4]	FCS_CKM.1(1)	DSA #1454
		ECDSA KeyGen Curves: P-256, P-384, P-521 Prerequisites: SHS #4535 DRBG #2284	FCS_CKM.1(1)	ECDSA #1528

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
		[FIPS PUB 186-4]		
		RSA Key Establishment [SP800-56B]	FCS_CKM.2(1)	No CAVP test exists. Vendor Affirmed. (See RSA #3044)
		KAS-FFC DH (dhEphem) FC: Hash: SHA2-256 Prerequisites: DSA #1454 SHS #4535 DRBG #2284 [FIPS PUB 186-4]	FCS_CKM.2(1)	Component #2046
		KAS-ECC ECDH (Ephemeral Unified) EC: Hash: SHA2-256 Curve: P-256 ED: Hash: SHA2-384 Curve: P-384 EE: Hash: SHA2-512 Curve: P-521 Prerequisites: ECDSA #1528 SHS #4535 DRBG #2284 [SP800-56A]	FCS_CKM.2(1)	Component #2046
		SHA-1, SHA2-256, SHA2-384, SHA2-512 [FIPS PUB 180-4 (SHS)]	FCS_COP.1(2)	SHS #4535
		RSA SigGen Signature: PKCS 1.5 Module: 2048 Hashes: SHA2-256, SHA2-384, SHA2-512	FCS_COP.1(3)	RSA #3044

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
		Modulo: 3072 Hashes: SHA2-256, SHA2-384, SHA2-512 RSA SigVer Signature: PKCS 1.5 Module: 2048 Hashes: SHA-1, SHA2-256, SHA2-384, SHA2-512 Modulo: 3072 Hashes: SHA-1, SHA2-256, SHA2-384, SHA2-512 Prerequisites: SHS #4535 DRBG #2284 [FIPS PUB 186-4]		
		ECDSA SigGen Curves: P-256, P-384, P-521 Hashes: SHA2-256, SHA2-384, SHA2-512 ECDSA SigVer Curves: P-256, P-384, P-521 Hashes: SHA-1, SHA2-256, SHA2-384, SHA2-512 Prerequisites: SHS #4535 DRBG #2284 [FIPS PUB 186-4]	FCS_COP.1(3)	ECDSA #1528
		HMAC-SHA-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512 Prerequisites: SHS #4535 [FIPS Pub 198-1 (HMAC), FIPS PUB 180-4 (SHS)]	FCS_COP.1(4)	HMAC #3767
		Hash_DRBG Mode: SHA2-256 Prerequisites: SHS #4535 [FIPS Pub 198-1 (HMAC),	FCS_RBG_EXT.1	DRBG #2284

Usage	Implementation	Algorithms and standards	SFR	CAVP cert
		FIPS Pub 180-4 (SHS)]		

2.1 Security Functional Requirements

2.1.1 Security audit (FAU)

2.1.1.1 Audit Data Generation (FAU_GEN.1)

FAU_GEN.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-FAU_GEN.1.1-AGD-01

The evaluator shall check the administrative guide and ensure that it lists all of the auditable events. The evaluator shall check to make sure that every audit event type selected in the ST is included.

Summary

The evaluator examined 4.2 *Configurating the Audit Subsystem* of [CCG] which provides related guidance for auditing. This section referes to the man pages `auditd(8)`, `auditd.conf(5)`, and `auditctl(8)`. The evaluator examined these man pages `auditd(8)` describes the audit log management daemon, `auditd.conf(5)` describes configuration file for the audit daemon, and `auditctl(8)` is the utility used for configuring the audit rules. The evaluator noted that although [CCG] does not provide an explicit listing of the auditable events selected in [ST] or explicit description of each audit record, these man pages (as well as the additional man pages they referred to, e.g., `aureport`) together provide a pretty comprehensive description of all auditable events implemented by the Linux TOE. The evaluator also took into consideration that the TOE type is a general purpose Linux operating system which comes with a rather mature and expansive user documentation system. The evaluator performed a thorough examination of the relevant man pages and determined that they provide sufficient coverage of all the auditable events selected in [ST]. Thus, the evaluator accepted the man pages as appropriate operational guidance related to auditing.

Test Assurance Activities

Assurance Activity AA-FAU_GEN.1.1-ATE-01

The evaluator shall test the OS's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the ST. This should include all instance types of an event specified. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record have the proper entries.

Summary

The evaluator verified audit record generation for all events defined in the ST table 7. The evaluator examined the audit format and verified that the audit logs included all the necessary information.

FAU_GEN.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-FAU_GEN.1.2-AGD-01

The evaluator shall check the administrative guide and ensure that it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall ensure that the fields contains the information required.

Summary

As stated previously, the evaluator examined 4.2 *Configurating the Audit Subsystem* of [CCG] which provides related guidance for auditing. This section referes to the man pages `auditd(8)`, `auditd.conf(5)`, and `auditctl(8)`. The evaluator examined these man pages `auditd(8)` describes the audit log management daemon, `auditd.conf(5)` describes configuration file for the audit daemon, and `auditctl(8)` is the utility used for configuring the audit rules. The evaluator noted that although [CCG] does not provide an explicit listing of the auditable events selected in [ST] or explicit description of each audit record, these man pages (as well as the additional man pages they referred to, e.g., `aureport`) together provide a pretty comprehensive description of all auditable events implemented by the Linux TOE. The evaluator also took into consideration that the TOE type is a general purpose Linux operating system which comes with a rather mature and expansive user documentation system. The evaluator performed a thorough examination of the relevant man pages and determined that they provide sufficient coverage of all the auditable events selected in [ST]. Thus, the evaluator accepted the man pages as appropriate operational guidance related to auditing.

Test Assurance Activities

Assurance Activity AA-FAU_GEN.1.2-ATE-01

The evaluator shall test the OS's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the ST. The evaluator shall ensure the audit records generated during testing match the format specified in the administrative guide, and that the fields in each audit record provide the required information.

Summary

The evaluator verified audit record generation for all events defined in the ST table 7. The evaluator examined the audit format and verified that the audit logs included all the necessary information.

2.1.2 Cryptographic support (FCS)

2.1.2.1 Cryptographic Key Generation (FCS_CKM.1(1))

TSS Assurance Activities

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

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

Summary

The evaluator reviewed sections 7.1.1 through 7.1.3 and ensured that for every cryptographic provider supported by the TOE (kernel (KCAPI and UEK), OpenSSL, OpenSSH, Libgcrypt and NSS), the key sizes were specified for RSA (2048, 3072 and 4096 bits), ECC (NIST P-256, NIST P-384, NIST P-521) and FFC (2048, 3072 and 4096 bits).

Guidance Assurance Activities

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

The evaluator will verify that the AGD guidance instructs the administrator how to configure the OS to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

Summary

The sponsor (Oracle) provided to the customer a large set of online documentation comprising general administrator and user guidance for Oracle Linux (OL) 7 via https://docs.oracle.com/cd/E52668_01/index.html.

The documentation includes the following:

- Manual pages (man pages), each of which describes one command or one interface function. Collectively, they contain descriptions of all security-relevant interfaces to the TOE. Man pages are further assessed in the work units of ADV_FSP.1E.
- Release Notes and Licensing Information
- Administrator's Guide
- Installation Guide
- Security Guide

In addition, the sponsor provided the following documentation as administrator and user guidance specific to the TOE:

- The Common Criteria Guide [CCG], which specifically describes all security requirements for administrators and users of the TOE in its evaluated configuration. It is the only document outlining the exact configuration and environment consistent with [ST]. It is also the document that takes precedence over all other user documentation in case there is a conflict of information.

The evaluator examined section 3.7.3 *SSH key-based Authentication* which states that to generate keys that can be used for key-based authentication, the utility `ssh-keygen(8)` is provided. This utility allows the user to specify the key algorithm and their allowable key sizes. The evaluator examined the man pages of `ssh-keygen(8)` and verified that it allows for specification of the ken generation keys and the key sizes listed in FCS_CKM.1.

Test Assurance Activities

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

Assurance Activity Note: The following tests may require the vendor to furnish a developer environment and developer tools that are typically not available to end-users of the OS.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator will verify the implementation of RSA Key Generation by the OS 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 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 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 will 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 will have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- *n = p·q,*
- *p and q are probably prime according to Miller-Rabin tests,*
- *GCD(p-1, e) = 1,*
- *GCD(q-1, e) = 1,*
- *2^16 <= e <= 2^256 and e is an odd integer*
- *|p-q| > 2^(nlen/2 - 100)*
- *p >= 2(nlen/2 - 1/2)*
- *q >= 2(nlen/2 - 1/2)*
- *2^(nlen/2) < d < LCM(p-1, q-1)*
- *e·d = 1 mod LCM(p-1, q-1)*

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

Assurance Activity AA-FCS_CKM.1-1-ATE-02

Key Generation for ANSI X9.31-1998 RSA Schemes

If the TSF implements the ANSI X9.31-1998 scheme, the evaluator will check to ensure that the TSS describes how the key-pairs are generated. In order to show that the TSF implementation complies with ANSI X9.31-1998, the evaluator will ensure that the TSS contains the following information:

- *The TSS shall list all sections of the standard to which the OS complies;*
- *For each applicable section listed in the TSS, for all statements that are not "shall" (that is, "shall not", "should", and "should not") , if the OS implements such options it shall be described in the TSS. If the included functionality is indicated as "shall not" or "should not" in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the OS;*
- *For each applicable section of Appendix B, any omission of functionality related to "shall" or "should" statements shall be described.*

Summary

Not implemented by TSF.

Assurance Activity AA-FCS_CKM.1-1-ATE-03

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator will 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 will submit the generated key pairs to the public key verification (PKV) function of a known good implementation. FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P-256, P-384 and P-521, the evaluator will 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 will obtain in response a set of 10 PASS/FAIL values.

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

Assurance Activity AA-FCS_CKM.1-1-ATE-04

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*
- *g^q mod p = 1*
- *g^x mod p = y*

for each FFC parameter set and key pair

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

2.1.2.2 Cryptographic Key Establishment (FCS_CKM.2(1))

TSS Assurance Activities

Assurance Activity AA-FCS_CKM.2-1-ASE-01

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

Summary

The evaluator reviewed section 7.1 *Cryptographic Support* and ensured that the two schemes for key establishment are Diffie-Hellman and EC Diffie-Hellman, and that they correspond to the key generation schemes identified in FCS_CKM.1.1. The TSS specifies that both schemes are used for the TLS, SSH protocols.

Assurance Activity AA-FCS_CKM.2-1-ASE-02

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

Summary

Section 7.1 of the ST specifies that for both OpenSSH (SSH) and NSS (TLS), the TOE acts both as a sender and a recipient for RSA-based key establishment schemes.

Assurance Activity AA-FCS_CKM.2-1-ASE-03

> *The evaluator will ensure that the TSS describes how the OS handles decryption errors.*

Summary

The end of sections 7.1.1, 7.1.2, 7.1.3 and 7.1.4 specify that for the kernel, OpenSSL, libgcrypt and NSS:

> "*In case of decryption errors, the TOE will return an error to the remote entity.*"

Guidance Assurance Activities

Assurance Activity AA-FCS_CKM.2-1-AGD-01

> *The evaluator will verify that the AGD guidance instructs the administrator how to configure the OS to use the selected key establishment scheme(s).*

Summary

[CCG] section 2.3.4.2 *Configure SSH server* provides related guidance to configure the SSH server.

[CCG] section 2.3.4.3 *Configure SSH client* provides related guidance to configure the SSH client.

Per these sections, configuration for the SSH server and SSH client requires specifying the configuration files `/etc/ssh/sshd_config` and `etc/ssh/ssh_config`, respectively, with the following configuration options including options for key established schemes:

Ciphers aes256-ctr,aes128-ctr,aes256-gcm@openssh.com, aes128-gcm@openssh.com, aes256-cbc,aes128-cbc

MACs hmac-sha2-512,hmac-sha2-256,hmac-sha1,hmac-sha1-96

KexAlgorithms ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256, diffie-hellman-group14-sha1

For the client side, the following are additional;

HostKeyAlgorithms ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521, ssh-rsa

PubkeyAcceptedKeyTypes ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521, ssh-rsa

Test Assurance Activities

Assurance Activity AA-FCS_CKM.2-1-ATE-01

Assurance 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 will verify the implementation of the key establishment schemes supported by the OS using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator will verify the OS'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 the OS has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the discrete logarithm cryptography (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 will 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 MAC data and the calculation of MAC tag.

Function Test

The Function test verifies the ability of the OS 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 OS's 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 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 will obtain the DKM, the corresponding OS's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information field OI and OS id fields.

If the OS does not use a KDF defined in SP 800-56A, the evaluator will obtain only the public keys and the hashed value of the shared secret.

The evaluator will 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 OS shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the OS to recognize another party’s valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator will obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the OS should be able to recognize. The evaluator generates a set of 30 test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator’s public keys, the OS’s public/private key pairs, MAC tag, and any inputs used in the KDF, such as the other info and OS id fields.

The evaluator will inject an error in some of the test vectors to test that the OS recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MAC'd, or the generated MAC tag. If the OS 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 OS's static private key to assure the OS 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 OS shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator will compare the OS's results with the results using a known good implementation verifying that the OS detects these errors.

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

Assurance Activity AA-FCS_CKM.2-1-ATE-02

SP800-56B Key Establishment Schemes

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

To conduct this test the evaluator will generate or obtain test vectors from a known good implementation of the OS's 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 MAC key and MAC tag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the OS with the same inputs (in cases where key confirmation is incorporated, the test shall use the MAC key from the test vector instead of the randomly generated MAC key used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

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

To conduct this test the evaluator will generate or obtain test vectors from a known good implementation of the OS's 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, any additional input parameters if applicable, the MAC tag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator will perform the key establishment decryption operation on the OS and ensure that the outputted plaintext keying material is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator will perform the key confirmation steps and ensure that the outputted MAC tag is equivalent to the MAC tag in the test vector.

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

Assurance Activity AA-FCS_CKM.2-1-ATE-03

In accordance with NIST Special Publication 800-56B, the OS 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 will 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 will 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.

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

2.1.2.3 Cryptographic Key Destruction (FCS_CKM.4)

TSS Assurance Activities

Assurance Activity AA-FCS_CKM.4-ASE-01

[TD0239]

The evaluator examines the TSS to ensure it describes how the keys are managed in volatile memory. This description includes details of how each identified key is introduced into volatile memory (e.g. by derivation from user input, or by unwrapping a wrapped key stored in non-volatile memory) and how they are overwritten.

The evaluator shall check to ensure the TSS lists each type of key that is stored in non-volatile memory, and identifies how the TOE interacts with the underlying platform to manage keys (e.g., store, retrieve, destroy). The description includes details on the method of how the TOE interacts with the platform, including an identification and description of the interfaces it uses to manage keys (e.g., file system APIs, platform key store APIs).

The evaluator examines the interface description for each different media type to ensure that the interface supports the selection(s) and description in the TSS.

If the ST makes use of the open assignment and fills in the type of pattern that is used, the evaluator examines the TSS to ensure it describes how that pattern is obtained and used. The evaluator shall verify that the pattern does not contain any CSPs.

The evaluator shall check that the TSS identifies any configurations or circumstances that may not strictly conform to the key destruction requirement.

Summary

The evaluator examined section 7.1 and verified that symmetric key material as well as (EC) Diffie-Hellman keys are stored in volatile memory and are always considered ephemeral. Asymmetric keys other than (EC) Diffie-Hellman are considered to be reused multiple times and are stored on the hard disk. Section 7.1 specifies the location where the persistent asymmetric keys are stored:

- /etc/ssh, $HOME/.ssh, /etc/ssh/moduli for OpenSSH
- /etc/pki for NSS

For the kernel crypto API, OpenSSL, libgcrypt, and NSS, all RAM buffers holding sensitive data or keys are overwritten before release the memory (see last sentence in sections 7.1.1, 7.1.2, 7.1.3, and 7.1.4).

Guidance Assurance Activities

Assurance Activity AA-FCS_CKM.4-AGD-01

There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS and any other relevant Required Supplementary Information. The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer and how such situations can be avoided or mitigated if possible.

Some examples of what is expected to be in the documentation are provided here.

When the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-leveling and garbage collection. This may create additional copies of the key that are logically inaccessible but persist physically. In this case, to mitigate this the drive should support the TRIM command and implements garbage collection to destroy these persistent copies when not actively engaged in other tasks.

Drive vendors implement garbage collection in a variety of different ways, as such there is a variable amount of time until data is truly removed from these solutions. There is a risk that data may persist for a longer amount of time if it is contained in a block with other data not ready for erasure. To reduce this risk, the operating system and file system of the OE should support TRIM, instructing the non-volatile memory to erase copies via garbage collection upon their deletion. If a RAID array is being used, only set-ups that support TRIM are utilized. If the drive is connected via PCI-Express, the operating system supports TRIM over that channel.

The drive should be healthy and contains minimal corrupted data and should be end of lifed before a significant amount of damage to drive health occurs, this minimizes the risk that small amounts of potentially recoverable data may remain in damaged areas of the drive.

Summary

Section 3.15.5 *Cryptographic Key Desctruction* of [CCG] provides related guidance for key destrcution. It states the following:

- Keys that are stored in files as described in [CCG] must be destroyed using the `scrub(1)` command only and no other methods are allowed.
- When using an SSD, the wear levelling mechanism prevents software from overwriting the exact physical location where the keys reside. An SSD must be physically destroyed at the end of life to assure that no cryptographic keys remain.
- Ephemeral keys that are maintained in RAM are erased by the TOE system when no longer in use. No user intervention required.

Test Assurance Activities

Assurance Activity AA-FCS_CKM.4-ATE-01

[TD0239]

- ***Test 1:*** *Applied to each key held as in volatile memory and subject to destruction by overwrite by the TOE (whether or not the value is subsequently encrypted for storage in volatile or non-volatile memory). In the case where the only selection made for the destruction method key was removal of power, then this test is unnecessary. The evaluator shall:*
  1. *Record the value of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Cause the TOE to stop the execution but not exit.*
  5. *Cause the TOE to dump the entire memory of the TOE into a binary file.*
  6. *Search the content of the binary file created in Step #5 for instances of the known key value from Step #1.*

  Steps 1-6 ensure that the complete key does not exist anywhere in volatile memory. If a copy is found, then the test fails.
- ***Test 2:*** *Applied to each key help in non-volatile memory and subject to destruction by the TOE. The evaluator shall use special tools (as needed), provided by the TOE developer if necessary, to ensure the tests function as intended.*
  1. *Identify the purpose of the key and what access should fail when it is deleted. (e.g. the data encryption key being deleted would cause data decryption to fail.)*
  2. *Cause the TOE to clear the key.*
  3. *Have the TOE attempt the functionality that the cleared key would be necessary for.*

  The test succeeds if step 3 fails.

The following tests apply only to selection a), since the TOE in this instance has more visibility into what is happening within the underlying platform (e.g., a logical view of the media). In selection b), the TOE has no visibility into the inner workings and completely relies on the underlying platform, so there is no reason to test the TOE beyond test 2.

For selection a), the following tests are used to determine the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.

- ***Test 3:*** *Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media (e.g., MBR file system):*
  1. *Record the value of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Search the logical view that the key was stored in for instances of the known key value from Step #1. If a copy is found, then the test fails.*
- ***Test 4:*** *Applied to each key held as non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator shall use a tool that provides a logical view of the media:*
  1. *Record the logical storage location of the key in the TOE subject to clearing.*
  2. *Cause the TOE to perform a normal cryptographic processing with the key from Step #1.*
  3. *Cause the TOE to clear the key.*
  4. *Read the logical storage location in Step #1 of non-volatile memory to ensure the appropriate pattern is utilized.*

  The test succeeds if correct pattern is used to overwrite the key in the memory location. If the pattern is not found the test fails.

Summary

These tests have been executed for SSH, TLS (NSS), and disk encryption keys.

2.1.2.4 Cryptographic Operation - Encryption/Decryption (FCS_COP.1(1))

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

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

The evaluator will verify that the AGD documents contain instructions required to configure the OS to use the required modes and key sizes. The evaluator will execute all instructions as specified to configure the OS to the appropriate state.

Summary

[CCG] section 2.3.4.2 *Configure SSH server* provides related guidance to configure the SSH server.

[CCG] section 2.3.4.3 *Configure SSH client* provides related guidance to configure the SSH client.

Per these sections, configuration for the SSH server and SSH client requires specifying the configuration files `/etc/ssh/sshd_config` and `etc/ssh/ssh_config`, respectively, with the following configuration options including options data encryption and decryption for:

`Ciphers aes256-ctr,aes128-ctr, aes256-gcm@openssh.com, aes128-gcm@openssh.com, aes256-cbc,aes128-cbc`

`MACs hmac-sha2-512,hmac-sha2-256,hmac-sha1,hmac-sha1-96`
KexAlgorithms ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman-group14-sha1

And for the client side, the following are additional: HostKeyAlgorithms ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-rsa
PubkeyAcceptedKeyTypes ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521,ssh-rsa

Test Assurance Activities

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

The evaluator will perform all of the following tests for each algorithm implemented by the OS 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 will 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 will 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 will 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 will 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 will 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 will 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 will 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 will 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 will 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 will test the encrypt functionality by encrypting an i-block message where 1 < i ≤ 10. The evaluator will 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 will also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i ≤10. The evaluator will 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 will 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 will 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 will 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 will 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 will 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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-CCM Tests

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

- *128 bit and 256 bit keys*
- *Two payload lengths. One payload length shall be the shortest supported payload length, greater than or equal to zero bytes. The other payload length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits).*
- *Two or three associated data lengths. One associated data length shall be 0, if supported. One associated data length shall be the shortest supported payload length, greater than or equal to zero bytes. One associated data length shall be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 2 16 bytes, an associated data length of 216 bytes shall be tested.*
- *Nonce lengths. All supported nonce lengths between 7 and 13 bytes, inclusive, shall be tested.*
- *Tag lengths. All supported tag lengths of 4, 6, 8, 10, 12, 14 and 16 bytes shall be tested.*

To test the generation-encryption functionality of AES-CCM, the evaluator will perform the following four tests:

- ***Test 1:** For EACH supported key and associated data length and ANY supported payload, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- ***Test 2:** For EACH supported key and payload length and ANY supported associated data, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*
- ***Test 3:** For EACH supported key and nonce length and ANY supported associated data, payload and tag length, the evaluator will supply one key value and 10 associated data, payload and nonce value 3-tuples and obtain the resulting ciphertext.*
- ***Test 4:** For EACH supported key and tag length and ANY supported associated data, payload and nonce length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and obtain the resulting ciphertext.*

To determine correctness in each of the above tests, the evaluator will compare the ciphertext with the result of generation-encryption of the same inputs with a known good implementation.

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 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator will supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

Additionally, the evaluator will use tests from the IEEE 802.11-02/362r6 document "Proposed Test vectors for IEEE 802.11 TGi", dated September 10, 2002, Section 2.1 AESCCMP Encapsulation Example and Section 2.2 Additional AES CCMP Test Vectors to further verify the IEEE 802.11-2007 implementation of AES-CCMP.

AES-GCM Test

The evaluator will 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 will test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of peter 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 will 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 will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

XTS-AES Test

The evaluator will 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 nonzero 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 will 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 Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test

The evaluator will test the authenticated encryption functionality of AES-KW for EACH combination of the following input parameter lengths:

- *128 and 256 bit key encryption keys (KEKs)*
- *Three plaintext lengths. One of the plaintext lengthsFCS_COP.1.1(2) shall be two semi-blocks (128 bits). One of the plaintext lengths shall be three semi-blocks (192 bits). The third data unit length shall be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).*

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption.

To determine correctness, the evaluator will use the AES-KW authenticated-encryption function of a known good implementation.

The evaluator will test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption.

The evaluator will test the authenticated-encryption functionality of AES-KWP using the same test as for AES-KW authenticated-encryption with the following change in the three plaintext lengths:

- *One plaintext length shall be one octet. One plaintext length shall be 20 octets (160 bits).*
- *One plaintext length shall be the longest supported plaintext length less than or equal to 512 octets (4096 bits).*

The evaluator will test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption.

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

2.1.2.5 Cryptographic Operation - Hashing (FCS_COP.1(2))

TSS Assurance Activities

Assurance Activity AA-FCS_COP.1-2-ASE-01

The evaluator will 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 checked that section 7.1 documents which hash functions will be used for which purpose. OpenSSL, Libgcrypt, and NSS will use SHA-1, SHA-224, SHA-256, SHA-384, SHA-512 for HMAC used for integrity for the TLS, SSH protocols, as well as hashing for signature generation and verification.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_COP.1-2-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 only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs. The evaluator will 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 evaluator will generate 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 evaluator will 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 evaluator will generate 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 evaluator will 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 evaluator will generate 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 evaluator will 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 evaluator will generate 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 evaluator will 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 evaluator will 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 evaluator will then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluator will then ensure that the correct result is produced when the messages are provided to the TSF.*

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

2.1.2.6 Cryptographic Operation - Signing (FCS_COP.1(3))

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_COP.1-3-ATE-01

The evaluator will perform the following activities based on the selections in the ST. The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

ECDSA Algorithm Tests

- ***Test 1:*** *ECDSA FIPS 186-4 Signature Generation Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator will 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 will use the signature verification function of a known good implementation.*
- ***Test 2:*** *ECDSA FIPS 186-4 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator will 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 will verify that 5 responses indicate success and 5 responses indicate failure.*

RSA Signature Algorithm Tests

- ***Test 1:*** *Signature Generation Test. The evaluator will verify the implementation of RSA Signature Generation by the OS 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 will have the OS use its private key and modulus value to sign these messages. The evaluator will 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 will perform the Signature Verification test to verify the ability of the OS to recognize another party's valid and invalid signatures. The evaluator will 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 evaluator will verify that the OS returns failure when validating each signature.*

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

2.1.2.7 Cryptographic Operation - Keyed-hash Message Authentication (FCS_COP.1(4))

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_COP.1-4-ATE-01

The evaluator will perform the following activities based on the selections in the ST. For each of the supported parameter sets, the evaluator will compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator will have the OS generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared against the result of generating HMAC tags with the same key and IV using a known-good implementation.

Summary

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

2.1.2.8 Random Bit Generation (FCS_RBG_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FCS_RGB_EXT.1-ASE-01

Documentation shall be produced - and the evaluator will perform the activities - in accordance with Appendix E and the Clarification to the Entropy Documentation and Assessment Annex. In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates.

Summary

The evaluator reviewed the Entropy Documentation and Assessment documentation containing information regarding the entropy noise sources and the statistical testing.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_RBG_EXT.1-ATE-01

The evaluator will perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator will perform 15 trials for each configuration. The evaluator will 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 will 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 will 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 values should be set/generated as described:

- ***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 than 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 will 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

This test is covered by CAVS-tests, please see ATE_OSPP.1-1.T1 for certificate numbers.

2.1.2.9 Storage of Sensitive Data (FCS_STO_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FCS_STO_EXT.1-ASE-01

The evaluator will check the TSS to ensure that it lists all persistent sensitive data for which the OS provides a storage capability. For each of these items, the evaluator will confirm that the TSS lists for what purpose it can be used, and how it is stored.

Summary

The evaluator reviewed the TSS and verified that the following data is persistently stored:

- asymmetric key materials (except Diffie-Hellman and EC Diffie-Hellman public and private keys): these keys are used for the SSH, TLS protocols and are stored in /etc/ssh, $HOME/.ssh, $HOME/.ssh/authorized_keys for OpenSSH, and in /etc/pki for NSS
- master volume AES key used by dm-crypt for encrypted storage is stored on the protected partition in a LUKS header

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_STO_EXT.1-ATE-01

The evaluator will confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1(1). The evaluator will also consult the developer documentation to verify that an interface exists for applications to securely store credentials.

Summary

The assurance activity does not require any testing activities.

2.1.2.10 TLS Client Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.1-ASE-01

The evaluator will check the TSS to ensure that it lists all persistent sensitive data for which the OS provides a storage capability. For each of these items, the evaluator will confirm that the TSS lists for what purpose it can be used, and how it is stored.

Summary

Please refer to AA-FCS_STO_EXT.1-ASE-01.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.1-AGD-01

The evaluator will also check the operational guidance to ensure that it contains instructions on configuring the OS so that TLS conforms to the description in the TSS.

Summary

The evaluator examined the following section of [CCG] for related guidance to FCS_TLSC_EXT.1:

- Section 3.15.2 *TLS Configuration* for related guidance for TLS.

The evaluator noted that from [ST], TLS is implemented by OpenSSL that is only a library to be used by some application. Thus, the allowable ciphersuites can only be set by the consuming application. In other words, no configuration of the OpenSSL library is possible without the consuming application. The FIPS mode/FIPS 140-2 compliant mode as described in section 2.3.4.5 *Enable FIPS Mode* of [CCG], however, enforces that only FIPS-approved algorithms/functions and by extension those claimed in [ST] are supported in the evaluated configuration.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.1-ATE-01

The evaluator will check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator will check the TSS to ensure that the cipher suites specified include those listed for this component. The evaluator will also check the operational guidance to ensure that it contains instructions on configuring the OS so that TLS conforms to the description in the TSS. The evaluator will also perform the following tests:

- ***Test 1:*** *The evaluator will 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 2:*** *The evaluator will attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection isFCS_TLSC_EXT.1.2 established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.*
- ***Test 3:*** *The evaluator will 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 will verify that the OS disconnects after receiving the server's Certificate handshake message.*
- ***Test 4:*** *The evaluator will configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.*
- ***Test 5:*** *The evaluator will perform the following modifications to the traffic:*
  - ***Test 5.1:*** *Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.*
  - ***Test 5.2*** *Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE cipher suite) or that the server denies the client's Finished handshake message.*
  - ***Test 5.3:*** *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 will verify that the client rejects the connection after receiving the Server Hello*
  - ***Test 5.4 (conditional):*** *[TD0163] If an ECDHE or DHE ciphersuite is selected, modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.*
  - ***Test 5.5:*** *Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.*
  - ***Test 5.6:*** *Send a garbled message from the Server after the Server has issued the Change Cipher Spec message and verify that the client denies the connection.*

Summary

Test 1: The evaluator generated all necessary certificates, keys, CRLS and chains for this test and connected the TOE to an OpenSSL TLS server configured to only accept the ciphersuites mandated by the Security Target and made sure that the connection was successful.

Test 2: The evaluator generated all necessary certificates, keys, CRLS and chains for this test and connected the TOE to an OpenSSL TLS server that contains the Server Authentication purpose in the extendedKeyUsage field and verified that a connection was established. For the second part of this test, a server certificate has been generated without the Server Authentication purpose in the extendedKeyUsage field. The evaluator made sure that both certificates are identical except for the serverAuth value in the extendedKeyUsage field. The connection was not established.

Test 3: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used an ECDSA certificate while forcing the server to use the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite. The evaluator verified that the OS disconnects after receiving the server's Certificate handshake message.

Test 4: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a proxy in order to force the server to use the TLS_NULL_WITH_NULL_NULL cipher suite and verified. that the client denies the connection

Test 5.1: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a proxy to modify the TLS version during the handshake in the server hello message and verified that the client rejects the connection.

Test 5.2: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a proxy to modify a byte in the nonce in the server hello message and verifier that the client rejects the server key exchange message.

Test 5.3: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a proxy to modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the CLinet Hello handshake message. THe evaluator verified that the client rejected the connection after receiving the Server Hello.

Test 5.4: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a proxy to modify the signature block in the server certificate in the key exchange handshake message, and verified that the client rejects the connection after receiving and verifying the certificate.

Test 5.5: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a proxy to modify a byte in the server finished message and verified that the client send a fatal alert upon receipt and did not send any application data.

Test 5.6: he evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a proxy to send a garble message from the server after the server has issued the change cipher spec message, and verified that the client denied the connection.

FCS_TLSC_EXT.1.2

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.2-ASE-01

The evaluator will 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 will ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the OS.

Summary

The last paragraph of section 7.1.3 describes the client's method of establishing the reference identifier. It also specifies that wildcards are supported.:

"*NSS supports all TLS cipher suites listed in FCS_TLSC_EXT.1. NSS allows using the following reference identifies to be verified during TLS channel establishment:*

- *DNS host name or IP address found in Common Name of the X.509 certificate. Wild cards are supported.*
- *DNS host name found in the SAN for DNS names of the X.509 certificate.*
- *URI name found in the SAN for URI names of the X.509 certificate.*

"

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.2-AGD-01

The evaluator will 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 noted that from [ST], TLS is implemented by OpenSSL that is only a library to be used by some application. Thus, setting the reference identifier to be used for TLS certificate validation can only done by the consuming application. In other words, no configuration of the OpenSSL library is possible without the consuming application.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.2-ATE-01

The evaluator will 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 will ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the OS. The evaluator will verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS. The evaluator will configure the reference identifier according to the AGD guidance and perform the following tests during a TLS connection:

- ***Test 1:*** *The evaluator will present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator will verify that the connection fails.*
- ***Test 2:*** *The evaluator will 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 will repeat this test for each supported SAN type.*
- ***[TD0304] Test 3 [conditional]:*** *If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.*
- ***Test 4:*** *The evaluator will 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 will verify that the connection succeeds.*
- ***Test 5:*** *The evaluator will perform the following wildcard tests with each supported type of reference identifier:*
  - ***Test 5.1:*** *The evaluator will 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 5.2:*** *The evaluator will present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *. example.com). The evaluator will configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator will 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 will configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails.*
  - ***Test 5.3:*** *The evaluator will present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator will configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator will configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.*
- ***Test 6:*** *[conditional] If URI or Service name reference identifiers are supported, the evaluator will configure the DNS name and the service identifier. The evaluator will 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 will repeat this test with the wrong service identifier (but correct DNS name)FCS_TLSC_EXT.1.3 FDP_ACF_EXT.1.1 and verify that the connection fails.*

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

Summary

Test 1: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator presented a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator verified that the connection failed.

Test 2: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator presented a server certificate that contained a CN that matched the reference identifier, contained the SAN extension, but did not contain an identifier in the SAN that matched the reference identifier. The evaluator verified that the connection failed. The evaluator repeated this test for each supported SAN type.

Test 3: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The removed the SAN extension from the certificate and kept the CN. The evaluator verified that the connection succeeded.

Test 4: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator set an incorrect CN but kept a correct SAN extension. The evaluator verified that the connection succeeded.

Test 5.1: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator generated a server certificate containing a wildcard that is not in the left-most label of the presented identifier and verified that the connection failed.

Test 5.2: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator generated a server certificate containing a wildcard in the left-most label but not preceding the public suffix. The evaluator configured the reference identifier with a single left-most label (https://fedora.laptop.cstlab) and verified that the connection succeeded. The evaluator configured the reference identifier without a left-most label as in the certificate (https://laptop.cstlab) and verified that the connection failed. The evaluator configured the reference identifier with two left-most labels (https://test.fedora.laptop.cstlab) and verified that the connection failed.

Test 5.3: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator generated a server certificate containing a wildcard in the left-most label immediately preceding the public suffix. The evaluator configured the reference identifier with a single left-most label (https://laptop.cstlab) and verified that the connection fails. The evaluator configured the reference identifier with two left-most labels (https://fedora.laptop.cstlab) and verified that the connection failed.

Test 6: This test is not applicable. URI or Service name reference identifiers are not supported.

Test 7: This test is not applicable. Certificate pinning is not claimed.

FCS_TLSC_EXT.1.3

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.3-ATE-01

The evaluator will use TLS as a function to verify that the validation rules in FIA_X509_EXT.1.1 are adhered to and shall perform the following additional test:

- ***Test 1:*** *The evaluator will demonstrate that a peer using a certificate without a valid certification path results in an authenticate failure. Using the administrative guidance, the evaluator will then load the trusted CA certificate(s) needed to validate the peer's certificate, and demonstrate that the connection succeeds. The evaluator then shall delete one of the CA certificates, and show that the connection fails.*
- ***Test 2:*** *The evaluator will demonstrate that a peer using a certificate which has been revoked results in an authentication failure.*
- ***Test 3:*** *The evaluator will demonstrate that a peer using a certificate which has passed its expiration date results in an authentication failure.*
- ***Test 4:*** *the evaluator will demonstrate that a peer using a certificate which does not have a valid identifier shall result in an authentication failure.*

Summary

Test 1: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator removed one CA in the CA chain and verified that the connection failed. The evaluator connected the TOE to an OpenSSL TLS server configured with the trusted CA certificates needed to validate the peer's certificate and verified that the connection was successful.

Test 2: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator generated a certificate for the server and then revoked it. The evaluator confirmed that the connection failed.

Test 3: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a certificate which passed its expiration date and verified that the connection resulted in authentication failure.

Test 4: The evaluator generated all necessary certificates, keys, CRLS and chains for this test. The evaluator used a certificate which did not have a valid identifier and verified that the connection resulted in authentication failure.

2.1.2.11 SSH Protocol (FCS_SSH_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FCS_SSH_EXT.1-SSH-ASE-01

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

Summary

The evaluator reviewed section 6.1.9 of [ST] and ensured than the selections indicated in the ST are consistent with the dependent components by reviewing the [SSH-EP]: the TOE is a Linux-based operating system and therefore implements both a client and server SSH application complying with RFCs 4251, 4252, 4253, 4254 and 5647, 5656 and 6668.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.1.2.12 SSH Protocol - Client (FCS_SSHC_EXT.1)

FCS_SSHC_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.1-SSH-ASE-01

> *The evaluator will check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHC_EXT.1.4, and ensure that password-based authentication methods are also allowed.*

Summary

The evaluator reviewed section 7.8 of [ST], which specifies that RSA and ECDSA are the public key algorithms that can be used for authentication. This is in accordance with FCS_SSHC_EXT.1.4 which specifies that

"*rsa, ecdsa-sha2-nistp256 and ecdsa-sha2-nistp384*"

can be used. Moreover, section 7.8.1.2.2 specifies again that RSA and ECDSA can be used, which is consistent with the rest of the ST.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1-SSH-ATE-01

> - ***Test 1:*** *The evaluator will, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection to an SSH server. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.*
> - ***Test 2:*** *Using the guidance documentation, the evaluator will configure the TOE to perform password-based authentication to an SSH server, and demonstrate that a user can be successfully authenticated by the TOE to an SSH server using a password as an authenticator.*

Summary

Test successful.

FCS_SSHC_EXT.1.2

TSS Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.2-SSH-ASE-01

> *The evaluator will check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.*

Summary

The evaluator checked section 7.8.1.1 and 7.8.1.2 describing the SSH Client and Server implementations in the TOE, and verified that for both sides of the SSH protocol, packets greater than 2^28 bytes are discarded.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.2-SSH-ATE-01

The evaluator will demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

Summary

The evaluator demonstrated that too large packets are rejected. The evaluator verified it through reviewing the SSH code. Specifically two locations were relevant: packet.c which defines the maximum packet size and channel.c which compares the packet size of the current package of the processed communication channel and reacts when the size if not within the limit.

FCS_SSHC_EXT.1.3

TSS Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.3-SSH-ASE-01

The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator will check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Summary

The evaluator checked the TSS and verified that section 7.8 specifies a description of this protocol. In this section, optional characteristics of the SSH implementation are provided, such as:

- SSH v2.0 is used
- encryption is defined as per section 4.3 of RFC4253
- Diffie-Hellman key exchange is defined as per section 6.1 of RFC4253
- Keyed hash is defined as par section 4.4 of RFC4253
- SSH supports password-based authentication
- SSH supports key-based authentication
- SSH perform an integrity check on the exchanged messages (and terminates the connection if an error is encountered)
- OpenSSL is the cryptographic provider for OpenSSH

Section 7.1.2 specifies the encryption algorithms that OpenSSL supports for the SSH v2.0 protocol, as being: AES-128 CBC, AES-256 CBC, AES-128 CTR, AES-256 CTR, AES-128-GCM and AES-256-GCM, which is in accordance with FCS_SSHC_EXT.1.3.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.3-SSH-AGD-01

[The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well.]

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

Summary

FCS_SSHC_EXT.1.3 (section 6.1.8.2 of [ST]) defines that the SSH software shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms:

`aes128-ctr,aes256-ctr, aes128-cbc, aes256-cbc, AEAD_AES_128_GCM, AEAD_AES_256_GCM.`

The TSS section 7.1.2 *OpenSSL* specifies the encryption algorithms that OpenSSL supports for the SSH v2.0 protocol which are AES-128 CBC, AES-256 CBC, AES-128 CTR, AES-256 CTR, AES-128-GCM and AES-256-GCM which are idential to those defined in FCS_SSHC_EXT.1.3.

The evaluator checked [CCG] section 2.3.4.3 *Configure SSH client* where the guidance for the SSH client is provided. The evaluator determined the following:

- The SSH client is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH client to use only approved encryption algorithms, specify the approved ciphers in the configuration file `etc/ssh/ssh_config` with the following the option:

  ```
  Ciphers aes256-ctr,aes128-ctr,aes256-gcm@openssh.com,aes256-gcm@openssh.com,
   aes128-gcm@openssh.com,aes256-cbc,aes128-cbc
  ```
  .

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.3-SSH-ATE-01

- ***Test 1:*** *The evaluator will establish an SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.*
- ***Test 2:*** *The evaluator will configure an SSH server to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator will attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.*

Summary

Test 1: The evaluator established an SSH connection using each of the encryption algorithms specified by the requirement: aes128-ctr, aes256-ctr, aes128-cbc, aes256-cbc, AEAD_AES_128_GCM, AEAD_AES_256_GCM

Test 2: The evaluator configured an SSH server to only allow the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator attempted to establish an SSH connection from the TOE to the SSH server and observed that the connection is rejected.

```
/usr/bin/ssh-copy-id: ERROR: Unable to negotiate with 10.4.148.210 port 22: no
matching cipher found. Their offer: 3des-cbc
```

FCS_SSHC_EXT.1.4

TSS Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.4-SSH-ASE-01

The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator will check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

Summary

The evaluator checked that section 7.8 specifies that the supported public key algorithms are RSA and ECDSA, as specified in FCS_SSHC_EXT.1.4. The RSA and ECDSA keys can be used for the host keys as well as for the per-user keys. Whenever a user registers his public with the server side, a key-based authentication takes place instead of a password-based authentication.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.4-SSH-AGD-01

[The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well.]

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

Summary

FCS_SSHC_EXT.1.4 (section 6.1.8.2 of [ST]) defines that the SSH client shall ensure that the SSH transport implementation uses ssh-rsa, ecdsa-sha2-nistp256 and ecdsa-sha2-nistp384 as its public key algorithm(s) and rejects all other public key algorithms.

The TSS section 7.1 *Cryptographic Support* states that OpenSSH supports RSA 2048 through 4096, ECDSA with P-256, P-384 and P-521 using SHA-1 or SHA-2 for authentication.

The TSS section 7.1.2 *OpenSSL* specifies the public key algorithms that OpenSSL/OpenSSH supports for the SSH v2.0 protocol are RSA and ECDSA which the evaluator found to be consistent with those claimed in FCS_SSHC_EXT.1.4.

The TSS section 7.8 *Secure Shell* specifies the public key algorithms that the TOE supports are RSA and ECDSA which the evaluator found to be consistent with those claimed in FCS_SSHC_EXT.1.4.

The evaluator checked [CCG] section 2.3.4.3 *Configure SSH client* where the guidance for the SSH client is provided. The evaluator determined the following:

- The SSH client is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH client to use only approved algorithms for public key, specify the approved ciphers in the configuration file `etc/ssh/ssh_config` with the following the option:

  ```
  PubkeyAcceptedKeyTypes ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,
  ecdsa-sha2-nistp521,ssh-rsa
  ```

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.4-SSH-ATE-01

- ***Test 1:*** *The evaluator will establish a SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.*
- ***Test 2:*** *The evaluator will configure an SSH server to only allow the ssh-dsa public key algorithm and no other public key algorithms. The evaluator will attempt to establish an SSH connection from the TOE to the SSH server and observe that the connection is rejected.*

Summary

Test 1: The evaluator created rsa and ecdsa keys, distributes them to the SSH server, and tested the usage of the respective pubkey methods. The evaluator observed successful negotiation.

Test 2: The evaluator created a dsa key, distributed it to the SSH server, and then attempted an pubkey authentication with that key which was expected to fail (because dsa should be not implemented or disabled). The evaluator observed that the connection was rejected.

FCS_SSHC_EXT.1.5

TSS Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.5-SSH-ASE-01

The evaluator will check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

Summary

Section 7.1 specifies that OpenSSL supports the following encryption algorithms: hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512. OpenSSL also supports the AEAD ciphers AES-128-GCM and AES-256-GCM, which also perform integrity mechanisms on the ciphertext and the associated data, which is in accordance with FCS_SSHC_EXT.1.5.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.5-SSH-AGD-01

The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the "none" MAC algorithm is not allowed).

Summary

FCS_SSHC_EXT.1.5 (section 6.1.8.2 of [ST]) defines that the SSH client shall ensure that the SSH transport implementation uses hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512 and AEAD_AES_128_GCM, AEAD_AES_256_GCM as its data integrity MAC algorithm(s) and rejects all other MAC algorithm(s).

The TSS section 7.1 *Cryptographic Support* specifies the allowed data integrity algorithms used in SSH connections with the TOE as AES-128 CBC, AES-256 CBC, AES-128 CTR, AES-256 CTR, HMAC SHA-1, HMAC SHA-256, HMAC SHA-512.

The TSS section section 7.1 *Cryptographic Support* specifies the data integrity algoirthms that SSH supports as HMAC SHA-1, HMAC SHA-256, HMAC SHA-512 which the evaluator found to be consistent with those claimed in FCS_SSHC_EXT.1.5.

The evaluator checked [CCG] section 2.3.4.3 *Configure SSH client* where the guidance for the SSH client is provided. The evaluator determined the following:

- The SSH client is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH client to use only approved ciphers for data integrity specify the approved ciphers in the configuration file `etc/ssh/ssh_config` with the following the option:

  ```
  MACs hmac-sha2-512,hmac-sha2-256,hmac-sha1,hmac-sha1-96
  ```

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.5-SSH-ATE-01

- ***Test 1:*** *The evaluator will establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.*
- ***Test 2:*** *The evaluator will configure an SSH server to only allow the "none" MAC algorithm. The evaluator will attempt to connect from the TOE to the SSH server and observe that the attempt fails.*
- ***Test 3:*** *The evaluator will configure an SSH server to only allow the hmac-md5 MAC algorithm. The evaluator will attempt to connect from the TOE to the SSH server and observe that the attempt fails.*

Summary

Test 1: The evaluator established a SSH connection using each of the integrity algorithms specified by the requirement (hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512 and AEAD_AES_128_GCM, AEAD_AES_256_GCM). The evaluator observed the successful negotiation of the algorithm e.g.

```
spawn ssh -c aes128-cbc svruser@10.4.148.210 svruser@10.4.148.210's password:

aes128-cbc: ok

spawn ssh -c aes256-cbc svruser@10.4.148.210 svruser@10.4.148.210's password:

aes256-cbc: ok

spawn ssh -c aes128-ctr svruser@10.4.148.210 svruser@10.4.148.210's password:

aes128-ctr: ok

spawn ssh -c aes256-ctr svruser@10.4.148.210 svruser@10.4.148.210's password:

aes256-ctr: ok
```

Test 2: The evaluator configured an SSH server to only allow the "none" MAC algorithm. The evaluator attempted to connect from the TOE to the SSH server and observed that the attempt failed.

Test 3: The evaluator configured an SSH server to only allow the hmac-md5 MAC algorithm in in SSH server's /etc/ssh/sshd_config. The evaluator attempted to connect from the TOE to the SSH server and observed that the attempt failed.

```
ssh atsec@10.4.148.210

no matching mac found: client hmac-sha2-512,hmac-sha2-256,hmac-sha1,hmac-sha1-96
server hmac-md5
```

FCS_SSHC_EXT.1.6

TSS Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.6-SSH-ASE-01

The evaluator will check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

Summary

The evaluator checked section 7.8.1 and verified that it specifies the key exchange algorithms as being Diffie-Hellman (with various groups) and EC Diffie-Hellman (with various curves). The evaluator compared these descriptions with FCS_SSHC_EXT.1.6 and verified that they correspond to the list in this component: diffie-hellman-group14-sha1, ecdh-sha2-nistp256 and ecdh-sha2-nistp384, ecdh-sha2-nistp521.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.6-SSH-AGD-01

The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Summary

FCS_SSHC_EXT.1.6 defines that the SSH client shall ensure that diffie-hellman-group14-sha1, ecdh-sha2-nistp256 and ecdh-sha2-nistp384, ecdh-sha2-nistp521 are the only allowed key exchange methods used for the SSH protocol.

The evaluator checked [CCG] section 2.3.4.3 *Configure SSH client* where the guidance for the SSH client is provided. The evaluator determined the following:

- The SSH client is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH client to use only approved key exchange methods, specify the approved methods in the configuration file `etc/ssh/ssh_config` with the following the option:

  ```
  KexAlgorithms ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,
  diffie-hellman-group14-sha1
  ```

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.6-SSH-ATE-01

- ***Test 1:*** *The evaluator will configure an SSH server to permit all allowed key exchange methods. The evaluator will attempt to connect from the TOE to the SSH server using each allowed key exchange method, and observe that each attempt succeeds.*

Summary

The evaluator configured an SSH server to permit all allowed key exchange methods (diffie-hellman-group14-sha1, ecdh-sha2- nistp256 and ecdh-sha2-nistp384, ecdh-sha2-nistp521). The evaluator attempted to connect from the TOE to the SSH server using each allowed key exchange method, and observed that each attempt succeeded.

FCS_SSHC_EXT.1.7

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.7-SSH-ATE-01

- ***Test 1:*** *[TD0331] The evaluator will configure the TOE to create a log entry when a rekey occurs. The evaluator will connect to the TOE with an SSH client and cause a rekey to occur according to the selection(s) in the ST, and subsequently the evaluator uses available methods and tools to verify that rekeying occurs. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports auditing of rekey events.*

Summary

The evaluator configured an SSH server to create a log entry when a rekey occurs. The evaluator connected to an SSH server with the TOE and caused a rekey to occur according to the selection(s) in the ST, and subsequently review the audit log to ensured that a rekey occurred.

```
debug1: need rekeying

debug1: SSH2_MSG_KEXINIT sent

debug1: rekeying in progress

debug1: SSH2_MSG_KEXINIT received

...

debug1: SSH2_MSG_NEWKEYS sent

debug1: expecting SSH2_MSG_NEWKEYS

debug1: set_newkeys: rekeying

debug1: SSH2_MSG_NEWKEYS received
```

FCS_SSHC_EXT.1.8

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_SSHC_EXT.1.8-SSH-ATE-01

- ***Test 1:*** *The evaluator will delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator will initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the user to accept or deny the key before continuing the connection.*
- ***Test 2:*** *The evaluator will add an entry associating a host name with a public key into the TOE's local database. The evaluator will replace, on the corresponding SSH server, the server's host key with a different host key. The evaluator will initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords).*

Summary

Test 1: The evaluator deleted all entries in the TOE's list of recognized SSH server host keys. The evaluator initiated a connection from the TOE to an SSH server. The evaluator observed that the TOE either rejected the connection or displayed the SSH server's public key and prompted the user to accept or deny the key before continuing the connection.

Test 2: The evaluator added an entry associating a host name with a public key into the TOE's local database. The evaluator replaced, on the corresponding SSH server, the server's host key with a different host key. The evaluator initiated a connection from the TOE to the SSH server using password-based authentication and observed that the TOE rejected the connection. The evaluator verified that the password was not transmitted to the SSH server.

2.1.2.13 SSH Protocol - Server (FCS_SSHS_EXT.1)

FCS_SSHS_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.1-SSH-ASE-01

The evaluator will check to ensure that the TSS contains a description of the public key algorithms that are acceptable for use for authentication, that this list conforms to FCS_SSHS_EXT.1.4, and ensure that password-based authentication methods are also allowed.

Summary

The evaluator checked sections 7.8 and verified that it specifies the public key algorithms that are acceptable for authentication as being RSA and ECDSA, and that password-based authentication is listed as well as one of the methods to authenticate. The evaluator ensured that the lists of the public key algorithms conforms to FCS_SSHS_EXT.1.4: ssh-rsa, ecdsa-sha2-nistp256 and ecdsa-sha2-nistp384.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.1-SSH-ATE-01

- ***Test 1:*** *The evaluator will, for each public key algorithm supported, show that the TOE supports the use of that public key algorithm to authenticate a user connection from an SSH client. Any configuration activities required to support this test shall be performed according to instructions in the guidance documentation.*
- ***Test 2:*** *The evaluator shall choose one public key algorithm supported by the TOE. The evaluator shall generate a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.*
- ***Test 3:*** *Using the guidance documentation, the evaluator will configure the TOE to perform password-based authentication on a client, and demonstrate that a user can be successfully authenticated by the TOE using a password as an authenticator.*
- ***Test 4:*** *The evaluator shall use an SSH client, enter an incorrect password to attempt to authenticate to the TOE, and demonstrate that the authentication fails.*

Summary

Test 1: The evaluator demonstrated that the TOE supports the use of each supported public key algorithm to authenticate a user connection from an SSH client.

Test 2: The evaluator generated a new key pair for that algorithm without configuring the TOE to recognize the public key for authentication. The evaluator used an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication failed.

Test 3: The evaluator configured the server to not allow pubkey authentication. The evaluator attempts to establish a connection. As the server did not allow it, it felt back to password authentication. The evaluator demonstrated that a user can be successfully authenticated by the TOE using a password as an authenticator.

Test 4: The evaluator configured the server to not allow pubkey authentication. The evaluator attempts to establish a connection. As the server did not allow it, it felt back to password authentication. The evaluator entered an incorrect password to attempt to authenticate to the TOE, and demonstrated that the authentication failed.

FCS_SSHS_EXT.1.2

TSS Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.2-SSH-ASE-01

The evaluator will check that the TSS describes how “large packets” in terms of RFC 4253 are detected and handled.

Summary

Packets for the server side of the SSH implementation is handled similarly as the client side. FCS_SSHS_EXT.1.2 and FCS_SSHC_EXT.1.2 specify that packets larger than 262144 will be dropped by the TOE.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.2-SSH-ATE-01

The evaluator will demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

Summary

The evaluator demonstrated that too large packets are rejected. The evaluator verified it through reviewing the SSH code. Specifically two locations were relevant: packet.c which defines the maximum packet size and channel.c which compares the packet size of the current package of the processed communication channel and reacts when the size if not within the limit.

FCS_SSHS_EXT.1.3

TSS Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.3-SSH-ASE-01

The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well. The evaluator will check the TSS to ensure that the encryption algorithms specified are identical to those listed for this component.

Summary

Please refer to AA-FCS_SSHC_EXT.1.3-SSH-ASE-01.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.3-SSH-AGD-01

[The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the encryption algorithms supported are specified as well.]

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

Summary

FCS_SSHS_EXT.1.3 (section 6.1.8.2 of [ST]) defines that the SSH server shall ensure that the SSH transport implementation uses the following encryption algorithms and rejects all other encryption algorithms: aes128-ctr, aes256-ctr, aes128-cbc, aes256-cbc, AEAD_AES_128_GCM, AEAD_AES_256_GCM.

The TSS section 7.1.2 *OpenSSL* specifies the encryption algorithms that OpenSSL supports for the SSH v2.0 protocol which are AES-128 CBC, AES-256 CBC, AES-128 CTR, AES-256 CTR, AES-128-GCM and AES-256-GCM which are identical to those defined in FCS_SSHS_EXT.1.3.

The evaluator checked [CCG] section 2.3.4.2 *Configure SSH server* where the guidance for the SSH server is provided. The evaluator determined the following:

- The SSH server is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH server to use only approved encryption algorithms, specify the approved ciphers in the configuration file `etc/ssh/sshd_config` with the following the option:

  ```
  Ciphers aes256-ctr,aes128-ctr,aes256-gcm@openssh.com,
  aes128-gcm@openssh.com,aes256-cbc,aes128-cbc
  ```

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.3-SSH-ATE-01

- ***Test 1:*** *The evaluator will initiate an SSH connection using each of the encryption algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.*
- ***Test 2:*** *The evaluator will configure an SSH client to only propose the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator will attempt to establish an SSH connection from the client to the TOE server and observe that the connection is rejected.*

Summary

Test 1: The evaluator initiated an SSH connection using each of the encryption algorithms specified in the ST and observed the successful negotiation of the algorithm

Test 2: The evaluator configured an SSH client to only propose the 3des-cbc encryption algorithm and no other encryption algorithms. The evaluator attempted to establish an SSH connection from the client to the TOE server and observed that the connection was rejected:

```
spawn ssh -c 3des-cbc testuser@10.4.148.210

Unable to negotiate with 10.4.148.210 port 22: no matching cipher found. Their
offer:
aes256-ctr,aes128-ctr,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-cbc,aes128-cbc
```

FCS_SSHS_EXT.1.4

TSS Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.4-SSH-ASE-01

The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well. The evaluator will check the TSS to ensure that the public key algorithms specified are identical to those listed for this component.

Summary

Please refer to AA-FCS_SSHC_EXT.1.4-SSH-ASE-01.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.4-SSH-AGD-01

[The evaluator will check the description of the implementation of this protocol in the TSS to ensure that optional characteristics are specified, and the public key algorithms supported are specified as well.]

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

Summary

FCS_SSHS_EXT.1.4 (section 6.1.8.3 of [ST]) defines that the SSH server shall ensure that the SSH transport implementation uses ssh-rsa, ecdsa-sha2-nistp256 and ecdsa-sha2-nistp384 as its public key algorithm(s) and rejects all other public key algorithms.

The TSS section 7.1 *Cryptographic Support* states that OpenSSH supports RSA 2048 through 4096, ECDSA with P-256, P-384 and P-521 using SHA-1 or SHA-2 for authentication.

The TSS section 7.1.2 *OpenSSL* specifies the public key algorithms that OpenSSL/OpenSSH supports for the SSH v2.0 protocol are RSA and ECDSA which the evaluator found to be consistent with those claimed in FCS_SSHC_EXT.1.4.

The TSS section 7.8 *Secure Shell* specifies the public key algorithms that the TOE supports are RSA and ECDSA which the evaluator found to be consistent with those claimed in FCS_SSHS_EXT.1.4.

The evaluator checked [CCG] section 2.3.4.2 *Configure SSH server* where the guidance for the SSH server is provided. The evaluator determined the following:

- The SSH server is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH server to use only approved algorithms for public key, specify the approved ciphers in the configuration file `etc/ssh/sshd_config` with the option specified in section 2.3.4.2 of [CCG].

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.4-SSH-ATE-01

- ***Test 1:*** *Using an appropriately configured client, the evaluator will establish an SSH connection using each of the public key algorithms specified by the requirement to authenticate. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.*
- ***Test 2:*** *The evaluator will configure an SSH client to propose only the ssh-dsa public key algorithm and no other public key algorithms. Using this client, the evaluator will attempt to establish an SSH connection to the TOE and observe that the connection is rejected.*

Summary

Test 1: The evaluator configured clients with supported public key algorithms and attempted to establish an SSH connection. The evaluator observed successful negotiation of the algorithms, e.g.
`spawn ssh -i testid_ecdsa -o PubkeyAcceptedKeyTypes=ecdsa-sha2-nistp384 testuser@10.4.148.210`

```
Enter passphrase for key 'testid_ecdsa':

Last login: Tue Nov 13 08:48:22 2018 from fedora.laptop.cstlab
```

Test 2: The evaluator configured an SSH client to propose only the ssh-dsa public key algorithm. Using this client, the evaluator attempted to establish an SSH connection to the TOE and observed that the connection was rejected.

FCS_SSHS_EXT.1.5

TSS Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.5-SSH-ASE-01

The evaluator will check the TSS to ensure that it lists the supported data integrity algorithms, and that that list corresponds to the list in this component.

Summary

Please refer to AA-FCS_SSHC_EXT.1.5-SSH-ASE-01.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.5-SSH-AGD-01

The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the "none" MAC algorithm is not allowed).

Summary

FCS_SSHS_EXT.1.5 (section 6.1.8.3 of [ST]) defines that the SSH server shall ensure that the SSH transport implementation uses hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512 and AEAD_AES_128_GCM, AEAD_AES_256_GCM as its MAC algorithm(s) and rejects all other MAC algorithm(s).

The evaluator checked [CCG] section 2.3.4.2 *Configure SSH server* where the guidance for the SSH server is provided. The evaluator determined the following:

- The SSH server is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH server to use only approved ciphers for data integrity specify the approved ciphers in the configuration file `etc/ssh/sshd_config` with the following the option:

  ```
  MACs hmac-sha2-512,hmac-sha2-256,hmac-sha1,hmac-sha1-96
  ```

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.5-SSH-ATE-01

- ***Test 1:*** *Using an appropriately configured client, the evaluator will establish a SSH connection using each of the integrity algorithms specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.*
- ***Test 2:*** *The evaluator will configure an SSH client to only propose the "none" MAC algorithm. Using this client, the evaluator will attempt to connect to the TOE and observe that the attempt fails.*
- ***Test 3:*** *The evaluator will configure an SSH client to only propose the hmac-md5 MAC algorithm. Using this client, the evaluator will attempt to connect to the TOE and observe that the attempt fails.*

Summary

Test 1: The evaluator will establish a SSH connection using each of the integrity algorithms specified by the requirement (hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512 and AEAD_AES_128_GCM, AEAD_AES_256_GCM). The evaluator observed the successful negotiation of the algorithm, e.g.

```
spawn ssh -m hmac-sha2-256 svruser@10.4.148.210 svruser@10.4.148.210's password:
```

```
hmac-sha2-256: ok

spawn ssh -m hmac-sha2-512 svruser@10.4.148.210 svruser@10.4.148.210's password:

hmac-sha2-512: ok
```

Test 2: Please note that 'none' MAC algorithm cannot be configured for OpenSSH. An error occurred when restarting the SSH client. The error can be viewed in /var/log/messages:
/etc/ssh/sshd_config line 157: Bad SSH2 mac spec 'none'

Test 3: The evaluator configured an SSH client to only allow the hmac-md5 MAC algorithm. The connection failed.

```
ssh -m hmac-md5 atsec@10.4.148.210

Unable to negotiate with 10.4.148.210 port 22: no matching MAC found. Their offer:
 hmac-sha2-512,hmac-sha2-256,hmac-sha1,hmac-sha1-96
```

FCS_SSHS_EXT.1.6

TSS Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.6-SSH-ASE-01

The evaluator will check the TSS to ensure that it lists the supported key exchange algorithms, and that that list corresponds to the list in this component.

Summary

Please refer to AA-FCS_SSHC_EXT.1.6-SSH-ASE-01.

Guidance Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.6-SSH-AGD-01

The evaluator will also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections to the TOE.

Summary

FCS_SSHS_EXT.1.6 (section 6.1.8.3 of [ST]) defines that the SSH server shall ensure that diffie-hellman-group14-sha1, ecdh-sha2-nistp256 and ecdh-sha2-nistp384, ecdh-sha2-nistp521 are the only allowed key exchange methods used for the SSH protocol.

The evaluator checked [CCG] section 2.3.4.2 *Configure SSH server* where the guidance for the SSH server is provided. The evaluator determined the following:

- The SSH server is only allowed to use approved ciphers, i.e., only those ciphers claimed in [ST]).
- To restrict the SSH server to use only approved key exchange methods, specify the approved methods in the configuration file `etc/ssh/sshd_config` with the following the option:

  ```
  KexAlgorithms ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,
  diffie-hellman-group14-sha1
  ```

The evaluator found that sufficient guidance is provided to set such restriction.

Test Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.6-SSH-ATE-01

- ***Test 1:*** *For each of the allowed key exchange methods, the evaluator will configure an SSH client to propose only it and attempt to connect to the TOE and observe that each attempt succeeds.*
- ***Test 2:*** *The evaluator shall configure an SSH client to only allow the diffiehellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the SSH Server and observe that the attempt fails.*

Summary

Test 1: The evaluator configured all allowed key exchange methods on the SSH client and attempted to connect to the TOE and observed that each attempt succeeds.

Test 2: The evaluator shall configure an SSH client to only allow the diffiehellman-group1-sha1 key exchange. The evaluator attempted to connect from the SSH client to the SSH Server and observed that the attempt failed.

```
spawn ssh -i testid_rsa -o PubkeyAcceptedKeyTypes=ssh-rsa -o
KexAlgorithms=diffie-hellman-group1-sha1 testuser@10.4.148.210

Unable to negotiate with 10.4.148.210 port 22: no matching key exchange method
found. Their offer:
ecdh-sha2-nistp521,ecdh-sha2-nistp384,ecdh-sha2-nistp256,diffie-hellman- group14-sha1
```

FCS_SSHS_EXT.1.7

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_SSHS_EXT.1.7-SSH-ATE-01

- ***Test 1:*** *[TD0331] The evaluator will configure the TOE to create a log entry when a rekey occurs. The evaluator will connect to the TOE with an SSH client and cause a rekey to occur according to the selection(s) in the ST, and subsequently the evaluator uses available methods and tools to verify that rekeying occurs. This could be done, e.g., by checking that a corresponding audit event has been generated by the TOE, if the TOE supports auditing of rekey events.*

Summary

The evaluator configured the TOE to create a log entry when a rekey occurs. The evaluator connected to the TOE with an SSH client and caused a rekey to occur and reviewed the audit log to ensure that a rekey occurred.

2.1.3 User data protection (FDP)

2.1.3.1 Acess Controls for Protecting User Data (FDP_ACF_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FDP_ACF_EXT.1-ASE-01

The evaluator will confirm that the TSS comprehensively describes the access control policy enforced by the OS. The description must include the rules by which accesses to particular files and directories are determined for particular users. The evaluator will inspect the TSS to ensure that it describes the access control rules in such detail that given any possible scenario between a user and a file governed by the OS the access control decision is unambiguous.

Summary

The evaluator reviewed sections 7.2 and 7.2.1, and confirmed that the OS implements the standard UNIX access control policy implemented by Linux-based operating systems:

> " *The TOE supports standard UNIX permission bits to provide one form of DAC for file system objects in all supported file systems. There are three sets of three bits that define access for three categories of users: the owning user, users in the owning group, and other users. The three bits in each set indicate the access permissions granted to each user category: one bit for read (r), one for write (w) and one for execute (x).* "

The evaluator reviewed this section and confirmed that the TSS comprehensively describes the access control policy enforced by the OS, and that it describes the access control rules in such detail that given any possible scenario between a user and a file governed by the OS the access control decision is unambiguous.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FDP_ACF_EXT.1-ATE-01

The evaluator will create two new standard user accounts on the system and conduct the following tests:

- ***Test 1:*** *The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to read the file created in the first user's home directory. The evaluator will ensure that the read attempt is denied.*
- ***Test 2:*** *The evaluator will authenticate to the system as the first user and create a file within that user's home directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to modify the file created in the first user's home directory. The evaluator will ensure that the modification is denied.*
- ***Test 3:*** *The evaluator will authenticate to the system as the first user and create a file within that user's user directory. The evaluator will then log off the system and log in as the second user. The evaluator will then attempt to delete the file created in the first user's home directory. The evaluator will ensure that the deletion is denied*
- ***Test 4:*** *The evaluator will authenticate to the system as the first user. The evaluator will attempt to create a file in the second user's home directory. The evaluator will ensure that the creation of the file is denied.*
- ***Test 5:*** *The evaluator will authenticate to the system as the first user and attempt to modify the file created in the first user's home directory. The evaluator will ensure that the modification of the file is accepted.*
- ***Test 6:*** *The evaluator will authenticate to the system as the first user and attempt to delete the file created in the first user's directory. The evaluator will ensure that the deletion of the file is accepted.*

Summary

Test 1: The evaluator authenticated to the system as the first user and create a file within that user's home directory (user1/user1.txt). The evaluator then logged off the system and logged in as the second user. The evaluator attempted to read the file created in the first user's home directory. The evaluator verified that the read attempt is denied (Permission denied).

Test 2: The evaluator authenticated to the system as the first user and created a file within that user's home directory. The evaluator logged off the system and log in as the second user. The evaluator then attempted to modify the file created in the first user's home directory. The evaluator verified that the read attempt is denied (Permission denied).

Test 3: The evaluator authenticated to the system as the first user and created a file within that user's user directory. The evaluator then logged off the system and loged in as the second user. The evaluator attempted to delete the file created in the first user's home directory. The evaluator verified that the read attempt is denied (Permission denied).

Test 4: The evaluator authenticated to the system as the first user. The evaluator attempted to create a file in the second user's home directory. The evaluator verified that the creation of the file is denied (Permission denied).

Test 5: The evaluator authenticated to the system as the first user and attempted to modify the file created in the first user's home directory. The evaluator ensured that the modification of the file was accepted.

Test 6: The evaluator authenticated to the system as the first user and attempted to delete the file created in the first user's directory. The evaluator ensured that the deletion of the file was accepted.

2.1.3.2 Information flow control (FDP_IFC_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FDP_IFC_EXT.1-ASE-01

The evaluator will verify that the TSS section of the ST describes the routing of IP traffic when a VPN client is enabled. The evaluator will ensure that the description indicates which traffic does not go through the VPN and which traffic does, and that a configuration exists for each in which only the traffic identified by the ST author as necessary for establishing the VPN connection (IKE traffic and perhaps HTTPS or DNS traffic) is not encapsulated by the VPN protocol (IPsec).

Summary

Section 7.9 of the TSS explains that

> "*The TOE provides different VPN interface to allow VPN clients to implement a VPN: the TOE provides the XFRM framework with the XFRM netlink interface. In addition, it provides the TUN/TAP interface for supporting user-space VPN clients operating at ISO/OSI level 2 or 3.*"

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.1.4 Identification and authentication (FIA)

2.1.4.1 Authentication Failure Handling (FIA_AFL.1)

FIA_AFL.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_AFL.1.1-ATE-01

The evaluator will set an administrator-configurable threshold for failed attempts, or note the ST-specified assignment. The evaluator will then (per selection) repeatedly attempt to authenticate with an incorrect password, PIN, or certificate until the number of attempts reaches the threshold. Note that the authentication attempts and lockouts must also be logged as specified in FAU_GEN.1.

Summary

The evaluator set an threshold (deny=3 value) for failed attempts and attempted to login via SSH and local console. The evaluator verified that it was not be possible to login when the failure count reached 3. The evaluator also examined audit records to verify that appropriate account lockout audit records were generated.

FIA_AFL.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_AFL.1.2-ATE-01

- ***Test 1:*** *The evaluator will attempt to authenticate repeatedly to the system with a known bad password. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*
- ***Test 2:*** *The evaluator will attempt to authenticate repeatedly to the system with a known bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*

- ***Test 3:*** *The evaluator will attempt to authenticate repeatedly to the system using both a bad password and a bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.*

Summary

Test 1: The evaluator set an threshold (deny=3 value) for failed attempts and attempted to login via SSH and local console. The evaluator verified that it was not be possible to login when the failure count reached 3. The evaluator also examined audit records to verify that appropriate account lockout audit records were generated.

Test 2: This test is not applicable, certificate-based authentication is not supported by the TOE.

2.1.4.2 Multiple Authentication Mechanisms (FIA_UAU.5)

TSS Assurance Activities

Assurance Activity AA-FIA_UAU.5-ASE-01

[TD0246]

The evaluator shall ensure that for all authentication mechanisms specified in FIA_UAU.5.1, the TSS shall describe each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication. Example rules are how the authentication mechanism authenticates the user (e.g. how does the TSF verify that the correct password was entered), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used).

Summary

Section 7.6 of the ST describes the several different authentications methods for the different usages for the TOE. This includes: PAM, SSH key-based authentication, authentication data management and session locking or X.509 certificates validation for TLS. Each section describes in detail how these mechanisms work and what the rules are or can be set to;

- PAM-based identification and authentication mechanisms: The Linux system implements identification and authentication through a set of trusted programs and protected databases. These trusted programs use an authentication infrastructure called the Pluggable Authentication Module (PAM). PAM allows different trusted programs to follow a consistent authentication policy.

  Linux uses a suite of libraries called the "Pluggable Authentication Modules" (PAM) that allow an administrative user to choose how PAM-aware applications authenticate users. The TOE provides PAM modules that implement all the security functionality to:
  - Provides login control and establishing all UIDs, GIDs and login ID for a subject
  - Ensure the quality of passwords
  - Enforce limits for accounts (such as the number of maximum concurrent sessions allowed for a user)
  - Enforce the change of passwords after a configured time including the password quality enforcement
  - Enforcement of locking of accounts after failed login attempts.
  - Restriction of the use of the root account to certain terminals

- Restriction of the use of the su and sudo commands

- Authentication Data Management: Each TOE instance maintains its own set of users with their passwords and attributes. Although the same human user may have accounts on different servers interconnected by a network and running an instantiation of the TOE, those accounts and their parameter are not synchronized on different TOE instances. As a result the same user may have different user names, different user Ids, different passwords and different attributes on different machines within the networked environment. Each TOE instance within the network maintains its own administrative database by making all administrative changes on the local TOE instance. The file /etc/passwd contains for each user the user's name, the id of the user, an indicator whether the password of the user is valid, the principal group id of the user and other (not security relevant) information. The file /etc/shadow contains for each user a hash of the user's password, the userid, the time the password was last changed, the expiration time as well as the validity period of the password and some other information that are not subject to the security functions as defined in the Security Target.
- SSH Key-Based Authentication: In addition to the PAM-based authentication outlined above, the OpenSSH server is able to perform a key-based authentication. To establish a key-based authentication, a user first has to generate an RSA, or ECDSA key pair. The private part of the key pair remains on the client side. The public part is copied to the server into the file .ssh/authorized_keys. When the login operation is performed the SSHv2 protocol tries to perform the "publickey" authentication using the private key on the client side and the public key found on the server side. The operations performed during the publickey authentication is defined in RFC 4252 chapter 7.
- Session Locking: The TOE uses the screen(1) application which locks the current session of the user either after an administrator-specified time of inactivity or upon the user's request. To unlock the session, the user must supply his password. Screen uses PAM to validate the password and allows the user to access his session after a successful validation.
- X.509 Certificate Validation: X.509 certificate validation is implemented by OpenSSL supporting the TLS protocol. RFC 5280 defines the X.509 certificate and validation mechanism.

Guidance Assurance Activities

Assurance Activity AA-FIA_UAU.5-AGD-01

[TD0246]

The evaluator shall ensure that for all authentication mechanisms specified in FIA_UAU.5.1, the TSS shall describe each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication. Example rules are how the authentication mechanism authenticates the user (e.g. how does the TSF verify that the correct password was entered), the result of a successful authentication (i.e. is the user input used to derive or unlock a key) and which authentication mechanism can be used at which authentication factor interfaces (i.e. if there are times, for example, after a reboot, that only specific authentication mechanisms can be used).

Summary

[ST] section 6.1.6.2 *FIA_UAU.5 Multiple Authentication Mechanisms* specifies FIA_UAU.5. It lists the following authentication mechanisms implemented by the TOE:

- authentication based on user name and password
- SSH public key-based authentication as specified by the Extended Package for Secure Shell

It also defines the following rules for authentication:

- authentication on the local console is based on user name and password
- authentication via the SSHv2 protocol first performs the certificate-based authentication which is followed by the user name and password authentication if the certificate-based authentication was unsuccessful

[CCG] section 5.2 *Authentication* provides related guidance for user authentication. It states that the user connecting to the TOE system via SSH must be authenticated with a user account that consists of a user name and password. The user account is created by the administrator who assigns a default password. The user is required to change this default password (using the `passwd(1)` command) the first time he/she connects the TOE system. It also states that the user may use SSH user keys for authentication as described in the man page of `ssh(1)`. When using key-based authentication, the SSH server would first attemp to authenticate the user with the provided key. If this authentication attempt fails, the SSH server then falls back to password-based authentication by asking the user for his/her password.

Additional guidance for SSH key-based authentication is provided in [CCG] section 5.4 *SSH key-based Authentication* which provides instructions for the user to generate and install the necessary cryptographic keys.

Test Assurance Activities

See assurance activities for SFR elements below.

FIA_UAU.5.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_UAU.5.1-ATE-01

If user name and password authentication is selected, the evaluator will configure the OS with a known user name and password and conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the OS using the known user name and password. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will attempt to authenticate to the OS using the known user name but an incorrect password. The evaluator will ensure that the authentication attempt is unsuccessful.*

Summary

Test 1: The evaluator performed multiple attempts to authenticate to the OS using the known user name and password. The evaluator ensured that the authentication attempts were successful.

Test 2: The evaluator performed multiple attempts to authenticate to the OS using the known user name but an incorrect password. The evaluator ensured that the authentication attempt were unsuccessful.

Assurance Activity AA-FIA_UAU.5.1-ATE-02

If user name and PIN that releases an asymmetric key is selected, the evaluator will examine the TSS for guidance on supported protected storage and will then configure the TOE or OE to establish a PIN which enables release of the asymmetric key from the protected storage (such as a TPM, a hardware token, or isolated execution environment) with which the OS can interface. The evaluator will then conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the OS using the known user name and PIN. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will attempt to authenticate to the OS using the known user name but an incorrect PIN. The evaluator will ensure that the authentication attempt is unsuccessful.*

Summary

Test 1: The evaluator attempted to authenticate to the OS using the known user name and PIN (password) as well as SSH public key based authentication with correct credentials. The evaluator ensured that the authentication attempt is successful.

Test 2: The evaluator attempted to authenticate to the OS using the known user name and incorrect PIN (password) as well as SSH public key based authentication with incorrect credentials. The evaluator ensured that the authentication attempt is unsuccessful.

Assurance Activity AA-FIA_UAU.5.1-ATE-03

If X.509 certificate authentication is selected, the evaluator will generate an X.509v3 certificate for a user with the Client Authentication Enhanced Key Usage field set. The evaluator will provision the OS for authentication with the X.509v3 certificate. The evaluator will ensure that the certificates are validated by the OS as per FIA_X509_EXT.1.1 and then conduct the following tests:

- ***Test 1:*** *The evaluator will attempt to authenticate to the OS using the X.509v3 certificate. The evaluator will ensure that the authentication attempt is successful.*
- ***Test 2:*** *The evaluator will generate a second certificate identical to the first except for the public key and any values derived from the public key. The evaluator will attempt to authenticate to the OS with this certificate. The evaluator will ensure that the authentication attempt is unsuccessful.*

Summary

This test is not applicable, the TOE does not implemented such functionality.

FIA_UAU.5.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_UAU.5.2-ATE-01

[TD0246]

- ***Test:*** *For each authentication mechanism rule, the evaluator shall ensure that the authentication mechanism(s) behave as documented in the TSS.*

Summary

The evaluator performed multiple authentication mechanism related tests. The evaluator attempted to authenticate to the OS using the known user name and PIN (password) as well as SSH public key based authentication with correct and incorrect credentials.

2.1.4.3 X.509 Certificate Validation (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FIA_X509_EXT.1.1-ASE-01

The evaluator will 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 reviewed section 7.6.5 *X.509 Certificate Validation* which describes how the TOE verifies X.509 certificates. It is specified that X.509 certificate validation is implemented by NSS supporting the TLS protocol.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_X509_EXT.1.1-ATE-01

The tests described must be performed in conjunction with the other certificate services assurance 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. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- ***Test 1:*** *The evaluator will demonstrate that validating a certificate without a valid certification path results in the function failing. The evaluator will then load a certificate or certificates as trusted CAs needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.*
- ***Test 2:*** *The evaluator will demonstrate that validating an expired certificate results in the function failing.*
- ***Test 3:*** *The evaluator will test that the OS can properly handle revoked certificates--conditional on whether CRL, OCSP, or OCSP stapling is selected; if multiple methods are selected, then a test shall be performed for each method. The evaluator will test revocation of the node certificate and revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). The evaluator will 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 either OCSP option is selected, the evaluator will configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator will configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set, and verify that validation of the CRL fails.*
- ***Test 5:*** *The evaluator will modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate should fail to parse correctly.)*
- ***Test 6:*** *The evaluator will modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate should not validate.)*
- ***Test 7:*** *The evaluator will modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate should not validate.)*

Summary

Test 1: The evaluator demonstrated that validating a certificate without a valid certification path results in the function failing. The evaluator loaded a certificate or certificates as trusted CAs needed to validate the certificate to be used in the function, and demonstrated that the function succeeded. The evaluator deleted one of the certificates, and showed that the function fails..

Test 2: The evaluator demonstrated that validating an expired certificate resulted in the function failing.

Test 3: The evaluator tested that the OS could properly handle revoked CRL certificates. The evaluator tested revocation of the node certificate and revocation of the first intermediate CA certificate. The evaluator ensured that a valid certificate was used, and that the validation function succeeds. The evaluator attempted to test with a certificate that had been revoked (CLR and first intermediate CA) to ensure when the certificate is no longer valid that the validation function fails.

Test 4: The evaluator configured the CA to sign a CRL with a certificate that does not have the cRLsign key usage set, and verified that validation of the CRL fails.

Test 5: The evaluator modified any byte in the first eight bytes of the certificate and verified that the certificate failed to validate.

Test 6: The evaluator modified any byte in the last byte of the certificate (the signature) and verified that the certificate failed to validate.

Test 7: The evaluator modified any byte in the public key of the certificate and verified that the certificate fails to validate.

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-FIA_X509_EXT.1.2-ATE-01

The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- ***Test 1:*** *The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.*
- ***Test 2:*** *The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension not set. The validation of the certificate path fails.*
- ***Test 3:*** *The evaluator will construct a certificate path, such that the certificate of the CA issuing the OS's certificate has the CA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeds.*

Summary

Test 1: The evaluator constructed a certificate path, such that the certificate of the CA issuing the OS's certificate did not contain the basicConstraints extension. The validation of the certificate path failed.

Test 2: The evaluator constructed a certificate path, such that the certificate of the CA issuing the OS's certificate had the CA flag in the basicConstraints extension not set. The validation of the certificate path failed.

Test 3: The evaluator constructed a certificate path, such that the certificate of the CA issuing the OS's certificate had the CA flag in the basicConstraints extension set to TRUE. The validation of the certificate path succeeded.

2.1.4.4 X.509 Certificate Authentication (FIA_X509_EXT.2)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_X509_EXT.2-ATE-01

The evaluator will acquire or develop an application that uses the OS TLS mechanism with an X.509v3 certificate. The evaluator will then run the application and ensure that the provided certificate is used to authenticate the connection. The evaluator will repeat the activity for any other selections listed.

Summary

The evaluator performed multiple X.509v3 certificate related tests. Please refer to FIA_X509 test described above.

2.1.5 Security management (FMT)

2.1.5.1 Management of security functions and behavior (FMT_MOF_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FMT_MOF_EXT.1-ASE-01

[TD0104] The evaluator shall verify that the TSS describes those management functions that are restricted to Administrators, including how the user is prevented from performing those functions, or not able to use any interfaces that allow access to that function.

Summary

The evaluator reviewed section 7.4 and confirmed that the TSS describe how the security management functions are restricted to the administrator:

> "*these files are protected using the DAC mechanisms against unauthorized access where usually the root user only is allowed to write to the files.*"

Section 7.4 specifies that all management functions are restricted to the root user, or users who are assigned to the "wheel" group and are therefore eligible to switch to the root user. Management functions.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FMT_MOF_EXT.1-ATE-01

[TD0104]

- ***Test 1:*** *For each function that is indicated as restricted to the administrator, the evaluation shall perform the function as an administrator, as specified in the Operational Guidance, and determine that it has the expected effect as outlined by the Operational Guidance and the SFR. The evaluator shall then perform the function (or otherwise attempt to access the function) as a non-administrator and observe that they are unable to invoke that functionality.*

Summary

For each function that is indicated as restricted to the administrator, the evaluator performed the function as an administrator and determined that it had the expected effect. The evaluator then performed the function as a non-administrator and observed that the evaluator was unable to invoke that functionality. The evaluators tested the various ways to manage the TOE. In case of configuration files, the evaluator verified that they existed, in which case management operation could be executed by simply a modification of that file.

2.1.5.2 Extended: Specification of Management Functions (FMT_SMF_EXT.1)

No assurance activities defined for this SFR.

2.1.6 Protection of the TSF (FPT)

2.1.6.1 Access Controls (FPT_ACF_EXT.1)

FPT_ACF_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FPT_ACF_EXT.1.1-ASE-01

The evaluator will confirm that the TSS specifies the locations of kernel drivers/modules, security audit logs, shared libraries, system executables, and system configuration files. Every file does not need to be individually identified, but the system's conventions for storing and protecting such files must be specified.

Summary

Section 7.3 specifies the location of kernel drivers and device drivers (/lib/modules), security audit logs (/var), shared libraries (/lib, /lib64, /usr/lib and /usr/lib64), system executables (/bin, /usr/bin, /sbin, /usr/sbin), system configuration files (/etc), and some additional information not mandated by this assurance activity.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_ACF_EXT.1.1-ATE-01

The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- ***Test 1:*** *The evaluator will attempt to modify all kernel drivers and modules.*
- ***Test 2:*** *The evaluator will attempt to modify all security audit logs generated by the logging subsystem.*
- ***Test 3:*** *The evaluator will attempt to modify all shared libraries that are used throughout the system.*
- ***Test 4:*** *The evaluator will attempt to modify all system executables.*
- ***Test 5:*** *The evaluator will attempt to modify all system configuration files.*
- ***Test 6:*** *The evaluator will attempt to modify any additional components selected.*

Summary

Test 1: The evaluator attempted to modify all kernel drivers and modules (/lib/*). The OS denied permission to complete the action.

Test 2: The evaluator attempted to modify all security audit logs generated by the logging subsystem (e.g. /var/log/audit/*). The OS denied permission to complete the action.

Test 3: The evaluator attempted to modify all shared libraries that were used throughout the system (e.g. /lib/* , /lib64/*, /usr/lib/*, /usr/lib64/*). The OS denied permission to complete the action.

Test 4: The evaluator attempted to modify all system executables (e.g. /bin/*, /usr/bin/*). The OS denied permission to complete the action.

Test 5:The evaluator attempted to modify all system configuration files (e.g. /etc/*,). The OS denied permission to complete the action.

Test 6: The evaluator attempted to modify any additional components selected (e.g. /boot/*). The OS denied permission to complete the action.

FPT_ACF_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_ACF_EXT.1.2-ATE-01

The evaluator will create an unprivileged user account. Using this account, the evaluator will ensure that the following tests result in a negative outcome (i.e., the action results in the OS denying the evaluator permission to complete the action):

- ***Test 1:*** *The evaluator will attempt to read security audit logs generated by the auditing subsystem*
- ***Test 2:*** *The evaluator will attempt to read system-wide credential repositories*
- ***Test 3:*** *The evaluator will attempt to read any other object specified in the assignment*

Summary

Test 1: Test 1: The evaluator attempted to read security audit logs generated by the auditing subsystem (/var/log/audit/). The OS denied permission to complete the action.

Test 2: The evaluator attempted to read system-wide credential repositories (/etc/shadow, /etc/security/opasswd). The OS denied permission to complete the action.

Test 3: The evaluator attempted to read any other object specified in the assignment. The OS denied permission to complete the action.

2.1.6.2 Address Space Layout Randomization (FPT_ASLR_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_ASLR_EXT.1-ATE-01

The evaluator will select 3 executables included with the TSF. These must include any web browser or mail client included with the TSF. For each of these apps, the evaluator will launch the same executables on two separate instances of the OS on identical hardware and compare all memory mapping locations. The evaluator will ensure that no memory mappings are placed in the same location. If the rare chance occurs that two mappings are the same for a single executable and not the same for the other two, the evaluator will repeat the test with that executable to verify that in the second test the mappings are different.

Summary

The evaluator verified that whether no memory mappings are placed in the same location for different applications. The evaluator tested: the /sbin/arping twice and compare its memory mappings, the sshd daemon twice and compare its memory mappings as well as /sbin/wpa_cli twice and compare its memory mappings.

2.1.6.3 Stack Buffer Overflow Protection (FPT_SBOP_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FPT_SBOP_EXT.1-ASE-01

The evaluator will determine that the TSS contains a description of stack-based buffer overflow protections used by the OS. Example implementations may be activated through compiler options such as "-fstack-protector-all", "-fstackprotector", and "/GS" flags. These are referred to by a variety of terms, such as stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner.

Summary

Section 7.3.1 specifies that

> "*All binaries (i.e. the kernel, kernel modules, executables, shared libraries) are compiled with the option “stack-protectorstrong”.*"

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_SBOP_EXT.1-ATE-01

> ***Test 1:*** *The evaluator will inventory the kernel, libraries, and application binaries to determine those that do not implement stack-based buffer overflow protections. This list should match up with the list provided in the TSS.*

Summary

The evaluator analyzed the kernel, libraries, and application binaries to determine those that did not implement stack-based buffer overflow protections. The evaluator checked the binary flags to identify whether binaries had protection. All setuid programs had the stack protection enabled. All programs not having the stack protection were mentioned in the ST.

2.1.6.4 Boot Integrity (FPT_TST_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FPT_TST_EXT.1-ASE-01

> *The evaluator will verify that the TSS section of the ST includes a comprehensive description of the boot procedures, including a description of the entire bootchain, for the TSF.*

Summary

The evaluator verified that section 7.3.2 includes a comprehensive description of the Linux boot process, which includes a description of the entire bootchain for the TSF:

" *The boot process consists of the following steps when the CPU is powered on or reset:*

1. *The firmware performs any hardware initialization steps.*
2. *The BIOS searches for the boot loader to boot in an order predefined by the firmware setting. Once a valid device is found, the firmware copies the contents of its first sector containing the boot loader into RAM, and starts executing the code just copied.*

Every boot loader performs the following general steps to initialize Linux:

1. *Loading the kernel image it is configured to load (the actual way of configuring the boot loader is different for each boot loader implementation). The loading process ensures that the kernel image is loaded to a well-defined memory location.*
2. *Loading the initramfs image it is configured to load. Again, this image is loaded to a well-defined memory location.*
3. *The kernel is compiled such that the setup function will always be loaded into a well-known memory location. This allows the boot loader to jump to the setup function to transfer control to the kernel.*

"

Assurance Activity AA-FPT_TST_EXT.1-ASE-02

> *The evaluator will ensure that the OS cryptographically verifies each piece of software it loads in the bootchain to include bootloaders and the kernel. Software loaded for execution directly by the platform (e.g. first-stage bootloaders) is out of scope.*
>
> *For each additional category of executable code verified before execution, the evaluator will verify that the description in the TSS describes how that software is cryptographically verified.*

The evaluator will verify that the TSS contains a description of the protection afforded to the mechanism performing the cryptographic verification.

Summary

Section 7.3.3 of the ST specifies that Secure Boot Support. The Unified Extensible Firmware Interface (UEFI) Secure Boot technology ensures that the bootloader's signature is verified. Then, the GRUB 2 bootloader and Oracle kernel's signatures are verified at boot time. The kernel also contains a public key to verify the signature of drivers and kernel modules. GRUB 2 module loading is disabled because there is no infrastructure to sign and verify GRUB 2 modules. Oracle provides a signed GRUB 2 binary which encompasses all modules supported by OL 7.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TST_EXT.1-ATE-01

The evaluator will perform the following tests:

- ***Test 1:*** *The evaluator will perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the OS properly boots.*
- ***Test 2:*** *The evaluator will modify a TSF executable that is part of the bootchain verified by the TSF (i.e. Not the first-stage bootloader) and attempt to boot. The evaluator will ensure that an integrity violation is triggered and the OS does not boot (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that in such a way to invalidate the structure of the module.).*
- ***Test 3:*** *If the ST author indicates that the integrity verification is performed using a public key, the evaluator will verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1. The evaluator will digitally sign the TSF executable with a certificate that does not have the Code Signing purpose in the extendedKeyUsage field and verify that an integrity violation is triggered. The evaluator shall repeat the test using a certificate that contains the Code Signing purpose and verify that the integrity verification succeeds. Ideally, the two certificates should be identical except for the extendedKeyUsage field.*

Summary

Test 1: The evaluator performed actions to cause TSF software to load and observe that the integrity mechanism did not flag any executables as containing integrity errors and that the OS properly boots.

Test 2: The evaluator modified a TSF executable that is part of the bootchain verified by the TSF (i.e. Not the first-stage bootloader) and attempted to boot. The evaluator observed that the integrity violation is triggered and the OS does not boot.

Test 3: This test is not applicable for the TOE.

2.1.6.5 Trusted Update (FPT_TUD_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.1-ASE-01

All supported origins for the update must be indicated in the TSS and evaluated.

Summary

The evaluator verified that section 7.3.4 includes a description of the trusted installation and update process for the TOE. The evaluator examined this section and noticed that the TOE uses the common Red Hat Packages (RPMs) method for software installation and updates. Every RPM package is signed using Oracle's certificate which is provided as part of the installation media for the OS. The administrator has to first verify the installation media to ensure integrity and authenticity by using signatures and hashes found on Oracle's download server.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

See assurance activities for SFR elements below.

FPT_TUD_EXT.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.1.1-ATE-01

> *The evaluator will check for an update using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require installing and temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update. (The evaluator is also to ensure that this query occurs over a trusted channel as described in FTP_ITC_EXT.1.)*

Summary

The evaluator verified the TOE trusted update operation with successful and unsuccessful output. The TOE trusted updated was tested with having the correct GPG key installed and without having the correct GPG key installed.

FPT_TUD_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.1.2-ATE-01

For the following tests, the evaluator will initiate the download of an update and capture the update prior to installation. The download could originate from the vendor's website, an enterprise-hosted update repository, or another system (e.g. network peer). All supported origins for the update must be indicated in the TSS and evaluated.

- ***Test 1:*** *The evaluator will ensure that the update has a digital signature belonging to the vendor prior to its installation. The evaluator will modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator will ensure that theFPT_TUD_EXT.2.1 FPT_TUD_EXT.2.2 OS does not install the modified update.*
- ***Test 2:*** *The evaluator will ensure that the update has a digital signature belonging to the vendor. The evaluator will then attempt to install the update (or permit installation to continue). The evaluator will ensure that the OS successfully installs the update.*

Summary

Test 1: The evaluator ensured that the update had a digital signature belonging to the vendor prior to its installation. The evaluator removed the correct GPG key (RPM-GPG-KEY-oracle-ol7.gpg). The evaluator. The evaluator attempted to install the update. The evaluator confirmed that OS does not install the modified update.

Test 2: The evaluator installed the correct GPG key on the TOE via ‘rpm --import RPM-GPG-KEY-oracle-ol7.gpg’). The evaluator attempted to install the update and ensured that the OS successfully installs the update.

2.1.6.6 Trusted Update for Application Software (FPT_TUD_EXT.2)

TSS Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.2-ASE-01

All origins supported by the OS must be indicated in the TSS and evaluated. However, this only includes those mechanisms for which the OS is providing a trusted installation and update functionality. It does not include user or administrator-driven download and installation of arbitrary files.

Summary

The evaluator verified that section 7.3.4 specifies that all RPMs are listed on Oracle's download server and that the OS either downloads the latest RPM, or notifies the user that a newer RPM is available. To avoid rollback attacks, these lists of RPMs are fetched from Oracle servers by the TOE.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

See assurance activities for SFR elements below.

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-FPT_TUD_EXT.2.1-ATE-01

The evaluator will check for updates to application software using procedures described in the documentation and verify that the OS provides a list of available updates. Testing this capability may require temporarily placing the system into a configuration in conflict with secure configuration guidance which specifies automatic update. (The evaluator is also to ensure that this query occurs over a trusted channel as described in FTP_ITC_EXT.1.)

Summary

The evaluator configured yum for updates as described in the guidance documentation and verified that the OS provides a list of available updates. Please note that the test that was performed for FPT_TUD_EXT.1 applies here as well, because the update mechanism are the same.

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-FPT_TUD_EXT.2.2-ATE-01

The evaluator will initiate an update to an application. This may vary depending on the application, but it could be through the application vendor's website, a commercial app store, or another system. All origins supported by the OS must be indicated in the TSS and evaluated. However, this only includes those mechanisms for which the OS is providing a trusted installation and update functionality. It does not include user or administrator-driven download and installation of arbitrary files.

- ***Test 1:*** *The evaluator will ensure that the update has a digital signature which chains to the OS vendor or another trusted root managed through the OS. The evaluator will modify the downloaded update in such a way that the digital signature is no longer valid. The evaluator will then attempt to install the modified update. The evaluator will ensure that the OS does not install the modified update.*
- ***Test 2:*** *The evaluator will ensure that the update has a digital signature belonging to the OS vendor or another trusted root managed through the OS. The evaluator will then attempt to install the update. The evaluator will ensure that the OS successfully installs the update.*

Summary

Test 1: The evaluator configured yum for updates as described in the guidance documentation and verified that the OS provides a list of available updates. The evaluator tested attempted to installed updates when the digital signature was no longer valid and ensured that the OS did not installed the modified update. Please note that the test that was performed for FPT_TUD_EXT.1 applies here as well, because the update mechanism are the same.

Test 2: The evaluator configured yum for updates as described in the guidance documentation and verified that the OS provides a list of available updates. The evaluator ensure that the update had a digital signature belonging to the OS vendor and verified that the OS successfully installed the update.

2.1.7 Trusted path/channels (FTP)

2.1.7.1 Trusted channel communication (FTP_ITC_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FTP_ITC_EXT.1-ATE-01

The evaluator will configure the OS to communicate with another trusted IT product as identified in the second selection. The evaluator will monitor network traffic while the OS performs communication with each of the servers identified in the second selection. The evaluator will ensure that for each session a trusted channel was established in conformance with the protocols identified in the first selection.

Summary

The evaluator configure the OS to communicate with management server using TLS and SSH. Please see extensive testing description of TLS and SSH testing performed above.

2.1.7.2 Trusted Path (FTP_TRP.1)

TSS Assurance Activities

Assurance Activity AA-FTP_TRP.1-ASE-01

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

Summary

The evaluator reviewed section 7./ *Trusted Path / Channel* which covers the trusted paths that the TOE supports. The TOE supports three different trusted paths: TLS and SSH. Remote users, including administrators, can use the SSH protocol implemented within OpenSSH to log in remotely to the machine and administer the system. The evaluator verified that this protocol (SSH), is consistent with the requirement in the ST from the SSH-EP.

Guidance Assurance Activities

Assurance Activity AA-FTP_TRP.1-AGD-01

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

Summary

Administrators can login to the TOE via the local console or SSH. The evaluator examined [CCG] section 3.7.3 *SSH key-based Authentication* for related guidance to FTP_TRP.1. It states the following:

- Key-based authentication is configured on a per-user basis by configuring the file `.ssh/authorized_keys` in the home directory of the user.
- Keys required for authentication is generated using the tool `ssh-keygen(8)`. Key generation algorithm/scheme and supported key sizes can be specified via this command.

Test Assurance Activities

Assurance Activity AA-FTP_TRP.1-ATE-01

The evaluator will also perform the following tests:

- ***Test 1:*** *The evaluator will ensure that communications using each remote administration method is tested during the course of the evaluation, setting up the connections as described in the operational guidance and ensuring that communication is successful.*
- ***Test 2:*** *For each method of remote administration supported, the evaluator will follow the operational guidance to ensure that there is no available interface that can be used by a remote user to establish a remote administrative sessions without invoking the trusted path.*
- ***Test 3:*** *The evaluator will ensure, for each method of remote administration, the channel data is not sent in plaintext.*
- ***Test 4:*** *The evaluator will ensure, for each method of remote administration, modification of the channel data is detected by the OS.*

Summary

Test 1: Remote OS administration is offered via SSH. The evaluator tested communications using SSH remote administration method for remote users. Please refer to test cases that describe SSH.

Test 2: Remote OS administration is offered via SSH. The evaluator followed the operational guidance to ensure that there were no available interface that could be used by a remote user to establish a remote administrative sessions without invoking SSH.

Test 3: Remote OS administration is offered via SSH. The evaluator ensured that SSH data was not sent in plaintext.

Test 4: Remote OS administration is offered via SSH. The evaluator ensured that modification of the channel (SSH) data was detected by the OS.

2.2 Security Assurance Requirements

2.2.1 Life-cycle support (ALC)

2.2.1.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activity AA-ALC_CMC.1-ALC-01

The evaluator will 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 will check the AGD guidance and OS 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 OS, the evaluator will examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Summary

The evaluator checked section 1.2 of the ST and checked that the name and version of the TOE Oracle Linux 7.3. The evaluator checked the guidance documents and also checked on the OS received for testing (using $cat /etc/os-release) that the version of the TOE was consistent across the documents. The evaluator checked the website maintained by the developer (http://edelivery.oracle.com/linux) and verified that the version of the TOE is also consistent.

2.2.1.2 TOE CM coverage (ALC_CMS.1)

Assurance Activity AA-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 OS is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the assurance 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 will 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 and linker flags). The evaluator will ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator will 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 checked in section 7.3.1 of the ST that all system binaries are compiled with the option “stack-protector-strong” to add a stack canary and associated verification code during the entry and exit of function frames. The entire TSF version is identified by the TOE version Oracle Linux 7.3, which is different with respect to other product developed by Oracle.

2.2.1.3 Extension: Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activity AA-ALC_TSU_EXT.1-ALC-01

The evaluator will 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 will verify that this description addresses the entire application. The evaluator will also verify that, in addition to the OS developer's process, any third-party processes are also addressed in the description. The evaluator will 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 will 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 OS patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator will verify that this time is expressed in a number or range of days. The evaluator will verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the OS. 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 reviewed section 7.3.4 which describes the trusted installation and update for the TOE. The evaluator verified that section 7.3.4 includes a description of the trusted installation and update process for the TOE. The evaluator examined this section and noticed that the TOE uses the common Red Hat Packages (RPMs) method for software installation and updates. Every RPM package is signed using Oracle's certificate which is provided as part of the installation media for the OS. The administrator has to first verify the installation media to ensure integrity and authenticity by using signatures and hashes found on Oracle's download server. The evaluator verified that section 7.3.4 specifies that all RPMs are listed on Oracle's download server and that the OS either downloads the latest RPM, or notifies the user that a newer RPM is available. To avoid rollback attacks, these lists of RPMs are fetched from Oracle servers by the TOE.

2.2.2 Security Target evaluation (ASE)

2.2.3 Guidance documents (AGD)

2.2.3.1 Operational user guidance (AGD_OPE.1)

Assurance Activity AA-AGD_OPE.1-AGD-01

If cryptographic functions are provided by the OS, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the OS. 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 OS. The documentation must describe the process for verifying updates to the OS by verifying a digital signature – this may be done by the OS or the underlying platform. The evaluator will 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 OS (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 hash/digital signature. The OS 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

As the evaluator noted in previous assurance activities that the TOE's OpenSSL is a cryptographic library that requires an application in order to configure it, thus, it cannot be configured in and by itself. Also, [ST] makes no explicit statements about which applications are covered by the evaluation as that is considered outside the scope of the TOE. The evaluator noted that the FIPS

mode/FIPS 140-2 compliant mode as described in section 2.3.4.5 *Enable FIPS Mode* of [CCG] is used to enforce that only FIPS-approved algorithms/functions and by extension those claimed in [ST] are supported in the evaluated configuration.

[CCG] section 3.15.2 *TLS Configuration* contains the following warning with regard to the use of other cryptographic engines in the evaluated configuration:

> *Please note that only the cipher implementations part of the OpenSSL libraries were subject to cryptographic validation. None of the OpenSSL engines were validated.*

[CCG] section 3.16 *Trusted Updates* provides related guidance for trusted updates. It states the following:

- The vendor Oracle provides regular updates to the TOE.
- The update mechanism is fully configured to obtain updates.
- The update process is performed via the command `yum update` which fetches the current list of available updates for the current installation base lists any applicable updates and asks the administrator whether to install the available updates.
- The administrator installs the updates via the command `yum install`. This command ensures the most current version of the software components is installed.
- If the update fails, the `yum` command will list the reason for failure and prevent the update operation.

The evaluator referred to the man pages for `yum` which includes options for making the update accessible to the OS (e.g., placement in a specific directory); instructions for initiating the update process; as well as discerning whether the process was successful or unsuccessful.

As a standard procedure for Linux distributions, the RPM packages are digitally signed and verified/enforced by default (via `rpm`) unless explicitly specified with `--nosignature` which omits verification of the RPM package or header signatures when reading the RPM package. Additionally, the signatures of each installed RPM package can be viewed with `rpm -qi` which shows the key that was used for signing the package as well as the signature ciphers.

The evaluator verified the guidance documentation, in particular [CCG] against the security functionality claimed in [ST] and determined that the guidance covers them all. Thus, it was clear to the evaluator which security functionality are covered by the evaluation activities.

2.2.3.2 Preparative procedures (AGD_PRE.1)

Assurance Activity AA-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

According to [ST] section 1.4.3 *Required Hardware and Software* and [CCG] section 1.3.1 *Hardware Requirements*, the evaluation supports the following platform:

- x86 64bit Intel Xeon processors:
  - Oracle Server X7-2

The TOE is expected to operate on the single platform listed. No other platforms are covered by the evaluation and thus no guidance is provided for any other platforms.

2.2.4 Tests (ATE)

2.2.4.1 Independent testing - conformance (ATE_IND.1)

Assurance Activity AA-ATE_IND.1-ATE-01

The evaluator will 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 Assurance Activities*

While it is not necessary to have one test case per test listed in an Assurance 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 affect; 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 OS 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 (IPsec, TLS). 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 test plan [ETP] describes the configuration of the tests system in section 2, which has been prepared as part of AGD. In section 3, all tests are specified, or references to test scripts are given. The test information has an introductory paragraph that refers to the ST SFR under test and also contains a copy of the test requirement of each applicable SFR component. The test plan also contains a separate section for each test, where the tester denotes the test result.

2.2.5 Vulnerability assessment (AVA)

2.2.5.1 Vulnerability survey (AVA_VAN.1)

Assurance Activity AA-AVA_VAN.1-AVA-01

The evaluator will 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.

Summary

The evaluator used the following public source of information for the search for vulnerabilities:

- CVE: https://cve.mitre.org/cve and https://nvd.nist.gov/vuln. These has been used for searching more product and version-specific weaknesses
- Google: the standard search engine has been used for general weaknesses of the involved technologies, but also for followup searches wheron CVE entries, where the CVE entry itself did not provide enough information to come to an conclusion on the applicability and exploitability of a vulnerability
- Redhat vulnerability response page: https://access.redhat.com/security/vulnerabilities lists vulnerability that have a considerable large scale impact and there therefore usually also relevant for low assurance evaluations. Note that the Oracle TOE code base stems from the RedHat Linux.
- Oracle Linux information on relevant CVEs (https://linux.oracle.com/pls/apex/f?p=130:21)
- Oracle Documentation for information on potentially additional interfaces or features that, although no security functions themselves, could have security-impact to the TOE

The evaluator used the following search terms on the CVE page:

- Oracle (Enterprise) Linux
- Oracle UEK
- names and versions of the rpm packages of the installed system ([RPMLIST]). Most notably:
  - kernel-3.10.0-862.3.3.0.1
  - kernel-uek-4.1.12-124.16.4
  - openssl-1:1.0.1e-60
  - openssh-6.6.1p1-31

No exploitable vulnerabilities have been identified.

A Appendixes

A.1 References

ADM	**Administrator's Guide**	
	Date received	2018-04-17
	File name	agd/manuals/AdminGuide-E54669-20170417.pdf
ALC-EVD	**Oracle Linux Life Cycle Document**	
	Version	1.7
	Date	2019-01-29
	File name	alc/Oracle Linux6_7_lifecycle_v1.7.pdf
AUDISPCONF	**Audit event dispatcher configuration**	
	Date	2018-01-31
	File name	agd/audisp-remote.conf.5.html
AUDITCONF	**Audit configuration**	
	Date	2018-01-31
	File name	agd/auditd.conf.5.html
AUDITD	**Audit daemon**	
	Date	2018-01-31
	File name	agd/auditd.8.html
AUDITRULES	**Audit rule maintenance**	
	Date	2018-01-31
	File name	agd/auditctl.8.html
AUSEARCH	**Audit search man page**	
	Date	2018-01-31
	File name	agd/ausearch.8.html
Can7566	**Ubuntu security portal**	
	Date	2018-06-25
	File name	ava/CVE-2018-7566_Ubuntu.html
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

CCDB-2017-05-17 **CC and CEM addenda - Exact Conformance, Selection-Based SFRs, Optional SFRs**
Version 0.5
Date 2017-05-17
Location https://www.commoncriteriaportal.org/files/ccfiles/CCDB-2017-05-17-CCaddenda-Exact_Conformance.pdf

CCEVS-TD0104 **FMT_SMF and FMT_MOF in OS PP**
Date 2016-09-16
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=107

CCEVS-TD0107 **FCS_CKM - ANSI X9.31-1998, Section 4.1.for Cryptographic Key Generation**
Date 2016-09-14
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=110

CCEVS-TD0163 **Update to FCS_TLSC_EXT.1.1 Test 5.4 and FCS_TLSS_EXT.1.1 Test**
Date 2017-04-05
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=167

CCEVS-TD0208 **Remote Users in OSPP**
Date 2017-06-09
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=212

CCEVS-TD0239 **Cryptographic Key Destruction in OS PP**
Date 2017-09-22
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=245

CCEVS-TD0240 **FCS_COP.1.1(1) Platform provided crypto for encryption/decryption**
Date 2017-11-27
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=246

CCEVS-TD0243 **SSH Key-Based Authentication**
Date 2017-10-03
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=249

CCEVS-TD0244 **FCS_TLSC_EXT - TLS Client Curves Allowed**
Date 2017-11-16
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=250

CCEVS-TD0246 **Assurance Activity for FIA_UAU.5.2**
Date 2017-10-31
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=252

CCEVS-TD0304 **Update to FCS_TLSC_EXT.1.2**
Date 2018-04-04
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=310

CCEVS-TD0305 **Handling of TLS connections with and without mutual authentication**
Date 2018-04-04
Location https://www.niap-ccevs.org/Documents_and_Guidance/view_td.cfm?td_id=311

CCG **Common Criteria Guide for Oracle Linux 7.3**
Version 1.0
Date 2018-12-20
File name agd/CCGuide-OL73-v1.0.pdf

CEM **Common Methodology for Information Technology Security Evaluation**
Version 3.1R5
Date April 2017
Location http://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R5.pdf

CEM-OSPPv4.1 **CEM extension for the Protection Profile for General Purpose Operating Systems (OSPP) v4.1**
Version 4.1
Date 2016-03-09

CEM-SSHEPv1.0 **CEM extension for the Extended Package for Secure Shell (SSH) v1.0**
Version 1.0
Date 2016-02-19

CHRONY **chrony configuration file**
Date 2018-02-05
File name agd/chrony.conf.5.html

CIPHERS **OpenSSL ciphers man page**
Date 2018-01-31
File name agd/ciphers.1ssl.gz

CRYPTSETUP **Cryptsetup man page**
Date 2018-01-31
File name agd/cryptsetup.8.html

CVEDISC **CVE discussions with developer**
Date 2018-06-15
File name ava/OL7-CVE-discussion-20180615 Oracle response.pdf

DBUSTUT **DBUS tutorial**
Date 2017-12-05
File name adv/dbus-tutorial.html

Deb6412 **Debian Security Portal**
Date 2018-06-25
File name ava/CVE-2018-6412_Debian.html

ECDSA	**ECDSA OpenSSL man page**	
	Date	2018-01-31
	File name	agd/ecdsa.3ssl.gz
ECPARAM	**EC parameter man page**	
	Date	2018-01-31
	File name	agd/ecparam.1ssl.gz
ELSA-2018-1453	**dhcp security update**	
	Date	2018-05-15
	Location	https://linux.oracle.com/errata/ELSA-2018-1453.html
ETP	**Evaluator testplan**	
	Version	1.7
	Date	2018-12-05
	File name	ate/OL7-ETP-v1.7.pdf
FSP	**Functional Specification**	
	Date	2018-10-29
	File name	adv/FSPmap-OL73_20181029.xls
GCRYPT	**The Libgcrypt Reference Manual**	
	Date	2013-07-25
	File name	adv/gcrypt.pdf
GENPKEY	**GENPKEY man page**	
	Date	2018-01-31
	File name	agd/genpkey.1ssl.gz
KERNAPI	**Kernel Crypto API**	
	Date	2018-01-31
	File name	adv/crypto-API.tar.xz
LDURL	**Logind DBUS info**	
	Date	2018-02-02
	Location	https://www.freedesktop.org/wiki/Software/systemd/logind/
MELTDOWN	**Meltdown paper**	
	Date	2017
	File name	ava/meltdown.pdf
NETLINK	**Netlink man page**	
	Date	2018-01-31
	File name	agd/netlink.7.html
NMCLI	**Network Manager Cmd**	
	Date	2018-01-31
	File name	agd/nmcli.1.html
NSS	**NSS cryptographic library**	
	Date	2018-10-29
	File name	adv/nss-docs.tar.gz

ONLINEDOC	**Oracle Linux 7 Documentation** Date 2018-02-14 Location https://docs.oracle.com/cd/E52668_01/index.html
OR-2018-14634	**Oracle patch information on integer overflow in kernel's create_elf_tables() vulnerability** Date 2018-09-25 Location https://linux.oracle.com/cve/CVE-2018-14634.html
OSCP_OSSL	**OCSP support OpenSSL** Date 2018-01-31 File name agd/ocsp.1ssl.gz
OSPP	**Operating System Protection Profile** Date 2016-03-09 Location https://www.niap-ccevs.org/pp/pp_os_v4.1.pdf
OSSL	**OpenSSL man pages** Date 2018-01-31 File name agd/openssl-man-pages.tar.gz
OSSLHMAC	**OpenSSL HMAC** Date 2018-01-31 File name agd/HMAC.3ssl.gz
PABOLDIN	**Meltdown PoC** Date 2018-01-09 File name ava/meltdown-exploit-master.zip
PAM_UNIX	**pam_unix plugin manpage** Date 2018-01-31 File name agd/pam_unix.8.html
PAMTALLY2	**PAM tally2 plugin** Date 2018-01-31 File name agd/pam_tally2.8.html
RFKILL	**rfkill man page** Date 2018-01-31 File name agd/rfkill.8.html
RH-2018-14634	**Redhat advisory on integer overflow in kernel's create_elf_tables() vulnerability** Date 2018-09-25 File name ava/CVE-2018-14634 - Red Hat Customer Portal.html
RH7738	**CVE-2018-7738 Customer Portal Information** Date 2018-06-25 File name ava/CVE-2018-7738 - Red Hat Customer Portal.html
RPM	**Package Manager** Date 2018-01-31 File name agd/rpm.8.html

RPMLIST	**RPM list of installed system**	
	Date	2018-06-06
	File name	ava/installed_rpms_20180606.txt
SCREEN	**Screen man page**	
	Date	2018-01-31
	File name	agd/screen.1.html
SCRUB	**scrub man apage**	
	Date	2018-01-31
	File name	agd/scrub.1.html
SSH	**SSH man page**	
	Date	2018-01-31
	File name	agd/ssh.1.html
SSHD	**SSH daemon man page**	
	Date	2018-01-31
	File name	agd/sshd.8.html
SSHD_CONFIG	**SSH config file**	
	Date	2018-01-31
	File name	agd/sshd_config.5.html
SSH-EP	**Extended Package for Secure Shell (SSH)**	
	Date	2016-02-19
	Location	https://www.niap-ccevs.org/pp/pp_ssh_ep_v1.0.pdf
ST	**Oracle Linux 7.3 Security Target**	
	Version	1.3
	Date	2019-01-29
	File name	ase/ST-OL73-v1.3.pdf
STUNNEL	**stunnel man page**	
	Date	2018-01-31
	File name	agd/stunnel.8.html
Sudo1000367	**Red Hat Security Portal**	
	Date	2018-06-25
	File name	ava/CVE-2017-1000367 - Red Hat Customer Portal.html
TESTFILES	**Test scripts and programs**	
	Date received	2018-11-30
	File name	ate/testfiles-2018-11-30.zip
TESTLOGS	**Test log files for some test cases**	
	Date	2018-11-30
	File name	ate/testlogs-2018-11-30.zip
TLSTEST	**TLS test description and result logs**	
	Date	2018-10-29
	File name	ate/TLS-test.zip

X509_OSSL	**x509 - Certificate display and signing utility**	
	Date	2018-01-31
	File name	agd/x509.1ssl.gz
X509IP	**X509 check functions**	
	Date	2018-01-31
	File name	agd/X509 check host.html
X509TEST	**X.509 test description and resuls**	
	Date	2018-10-29
	File name	ate/X509-test.zip

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.