Assurance Activity Report

Apple macOS 26 Tahoe

VID 11695

Version 1.1

Evaluated by:

4516 Seton Center Pkwy, Suite 250

Austin, TX 78759

United States of America

Validation Body:

NIAP-CCEVS

TOE Developer and Sponsor:

Apple Inc.

Report Authorized by:

Trang Huynh

This report must not be used to claim product certification, approval, or endorsement by NIAP CCEVS, NVLAP, NIST, or any agency of the Federal Government.

Classification Note

Public Information (public)

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

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

Revision History

Version	Date	Author(s)	Changes to Previous Revision
1.0	2026-07-02	Joachim Vandersmissen	First version
1.1	2026-07-15	Joachim Vandersmissen	Update CVE search date

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 additional assurance activities defined in the following:

- [CFG_GPOS_BT_WLANC_v1.0]: PP-Configuration for General Purpose Operating System, Bluetooth, and Wireless Local Area Network Clients, Version 1.0, dated 2025-05-12
    - [PP_OS_V4.3]: Protection Profile for General Purpose Operating Systems, Version 4.3, dated 2022-09-27
    - [MOD_BT_V1.0]: PP-Module for Bluetooth, Version 1.0, dated 2021-04-15
    - [MOD_WLANC_V1.0]: PP-Module for WLAN Clients, Version 1.0, dated 2022-03-31
- [PKG_TLS_V1.1]: Functional Package for Transport Layer Security (TLS), Version 1.1, dated 2019-03-01

This evaluation claims exact compliance with the above PP-Configuration, PPs, PP-Modules, and Functional Packages.

The following scheme documents and interpretations have been considered:

- [CCEVS-LG]: "CCEVS Labgrams", version as of July 2026.
- [CCEVS-PL]: "CCEVS Scheme Policy Letters", version as of July 2026.
- [CCEVS-PUB]: "CCEVS Scheme Publications", version as of July 2026.
- [CCEVS-TD]: "Technical Decisions", version as of July 2026.

2 Evaluation Results

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

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

2.1 CAVP Summary

The TOE uses the following cryptographic implementations:

- USR: Apple corecrypto Module 26 [Apple silicon, User, Software, SL1]
- KRN: Apple corecrypto Module 26 [Apple silicon, Kernel, Software, SL1]
- SKS: Apple corecrypto Module 26 [Apple silicon, Secure Key Store, Hardware, SL2]
- SE: Secure Enclave hardware onboard Apple silicon
- BC: Broadcom Crypto Hardware Module

The tables below show the cryptographic services used by the TOE and provided by the cryptographic implementations that are included in the TOE, describing cryptographic operations, the algorithm names, and the applicable standard. The table also includes the certificates obtained from the Cryptographic Algorithm Validation Program (CAVP) in the evaluated configuration for each of the cryptographic algorithms.

Table 1: Mapping of SFRs to CAVP certificates (USR)

SFR	Algorithm	Capabilities	Standard	CAVP cert.
FCS_CKM.1	RSA	Modulus: 3072, 4096 bits	[FIPS186-5]	A7643
	ECC	Curves: P-256, P-384, P-521	[FIPS186-5]	A7643
FCS_CKM.2	RSA Key Establishment	Modulus: 3072, 4096 bits	[RFC8017]	CCTL Tested
	KAS-ECC-SSC	Curves: P-256, P-384, P-521	[SP800-56Ar3]	A7641
FCS_COP.1/ENCRYPT	AES-KW	256 bits encrypt, decrypt	[SP800-38F]	A7639
	AES-GCM	256 bits encrypt, decrypt	[SP800-38D]	A7642
FCS_COP.1/HASH	SHA2-256	Byte-oriented mode	[FIPS180-4]	A7644
	SHA2-384, SHA2-512			A7645
FCS_COP.1/SIGN	RSA SigGen	Modulus: 3072, 4096 bits Hash: SHA2-256, SHA2-384, SHA2-512 Padding: PKCS#1 v1.5 and PSS	[FIPS186-5]	A7643
	RSA SigVer	Modulus: 3072, 4096 bits Hash: SHA2-256, SHA2-384, SHA2-512 Padding: PKCS#1 v1.5 and PSS	[FIPS186-5]	A7643
	ECDSA SigGen	Curves: P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A7643
	ECDSA SigVer	Curves: P-384, P-521 Hash: SHA2-256, SHA2-384, SHA2-512	[FIPS186-5]	A7643
FCS_COP.1/KEYHMAC	HMAC-SHA2-256	Byte-oriented mode	[FIPS198-1]	A7644
	HMAC-SHA2-384			A7645
FCS_RBG_EXT.1	CTR_DRBG	AES-256	[SP800-90Ar1]	A7642

Table 2: Mapping of SFRs to CAVP certificates (KRN)

SFR	Algorithm	Capabilities	Standard	CAVP cert.
FCS_COP.1/HASH	SHA2-384	Byte-oriented mode	[FIPS180-4]	A7637
FCS_COP.1/SIGN	RSA SigVer	Modulus: 4096 bits Hash: SHA2-384 Padding: PKCS#1 v1.5	[FIPS186-5]	A7635
FCS_RBG_EXT.1	CTR_DRBG	AES-256	[SP800-90Ar1]	A7634

Table 3: Mapping of SFRs to CAVP certificates (SKS)

SFR	Algorithm	Capabilities	Standard	CAVP cert.
FCS_COP.1/HASH	SHA2-384	Byte-oriented mode	[FIPS180-4]	A7652
FCS_COP.1/SIGN	RSA SigVer	Modulus: 4096 bits Hash: SHA2-384 Padding: PKCS#1 v1.5	[FIPS186-5]	A7652

Table 4: Mapping of SFRs to CAVP certificates (SE)

SFR	Algorithm	Capabilities	Standard	CAVP cert.
FCS_RBG_EXT.1	CTR_DRBG	AES-256	[SP800-90Ar1]	A3490, A6548, A6924

Table 5: Mapping of SFRs to CAVP certificates (BC)

SFR	Algorithm	Capabilities	Standard	CAVP cert.
FCS_COP.1/EN-CRYPT	AES-CTR	256 bits encrypt, decrypt	[SP800-38A]	AES 5926, AES 5927
	AES-CCM	128 bits encrypt, decrypt	[SP800-38C]	AES 5926, AES 5927, A1932
	AES-CCM	256 bits encrypt, decrypt	[SP800-38C]	AES 5952, AES 5953, A1932
	AES-GCM	256 bits encrypt, decrypt	[SP800-38D]	AES 5926, AES 5927, A1932

2.2 Security Functional Requirements

2.2.1 Security audit (FAU)

2.2.1.1 Audit Data Generation (FAU_GEN.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-OSPP-FAU_GEN.1-AGD-01

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

The evaluator will 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 will ensure that the fields contains the information required.

Summary

Section 6.1 of the [CCGUIDE] lists the events that are audited, the format of the audit record XML, and samples for each auditable event. The evaluator verified that this list matches the [ST].

Test Assurance Activities

Assurance Activity AA-OSPP-FAU_GEN.1-ATE-01

[TD1005] The evaluator will 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 will 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 triggered all events listed in the [ST], then used the auditd system to save and view the logs. The evaluator verified that accurate audit logs are generated with content and formatting consistent with those described in the TSS.

2.2.1.2 Audit Data Generation (Bluetooth) (FAU_GEN.1/BT)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FAU_GEN.1-BT-ASE-01

There are additional auditable events that serve to extend the FAU_GEN.1 SFR found in each Base-PP.

This SFR is evaluated in the same manner as defined by the Evaluation Activities for the claimed Base-PP. The only difference is that the evaluator shall also assess the auditable events required for this PP-Module in addition to those defined in the claimed Base-PP.

Summary

This assurance activity is performed in conjunction with the assurance activities for FAU_GEN.1 from the Base-PP. The evaluator verified that all events for Bluetooth are covered.

Sections 6.2 and 6.3 of the [CCGUIDE] list the Bluetooth events that are audited, the format of the Unified Logging audit record, and samples for each auditable event. The evaluator verified that this list matches the [ST].

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.1.3 Audit Data Generation (Wireless LAN) (FAU_GEN.1/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FAU_GEN.1-WLAN-ASE-01

The evaluator shall check the TSS and ensure it provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field.

If "invoke platform-provided functionality" is selected, the evaluator shall examine the TSS to verify it describes (for each supported platform) how this functionality is invoked (it should be noted that this may be through a mechanism that is not implemented by the WLAN Client; however, that mechanism will be identified in the TSS as part of this evaluation activity).

Summary

Section 7.1.1 of the [ST] describes the format for audit records. This section includes each format type and describes each (general) field associated with the audit records.

"Invoke platform-provided functionality" is not selected.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FAU_GEN.1-WLAN-AGD-01

The evaluator shall check the operational guidance and ensure it lists all of the auditable events and 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 check to make sure that every audit event type mandated by the PP-Module is described and that the description of the fields contains the information required in FAU_GEN.1.2/WLAN, and the additional information specified in Table 2 in the main document and Table 5 in the main document.

The evaluator shall in particular ensure that the operational guidance is clear in relation to the contents for failed cryptographic events. In the Auditable Events tables, information detailing the cryptographic mode of operation and a name or identifier for the object being encrypted is required. The evaluator shall ensure that name or identifier is sufficient to allow an administrator reviewing the audit log to determine the context of the cryptographic operation (for example, performed during a key negotiation exchange, performed when encrypting data for transit) as well as the non-TOE endpoint of the connection for cryptographic failures relating to communications with other IT systems.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of this PP-Module. The TOE may contain functionality that is not evaluated in the context of this PP-Module because the functionality is not specified in an SFR. This functionality may have administrative aspects that are described in the operational guidance. Since such administrative actions will not be performed in an evaluated configuration of the TOE, the evaluator shall examine the operational guidance and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the PP-Module, which thus form the set of "all administrative actions". The evaluator may perform this activity as part of the activities associated with ensuring the AGD_OPE guidance satisfies the requirements.

Summary

Sections 6.2 and 6.4 of the [CCGUIDE] list the WLAN events that are audited, the format of the Unified Logging audit record, and samples for each auditable event. The evaluator verified that this list matches the [ST], and that the audit records contained the information required in FAU_GEN.1.2/WLAN including the additional information.

Cryptographic failure events (e.g., failure to establish an EAP-TLS session) are clearly documented in the [CCGUIDE]. The evaluator confirmed that the example records contain sufficient information to determine the context of the failure, the source of the failure, and the endpoints involved.

The evaluator examined the [ST] and the [CCGUIDE], and performed testing and used the TOE. The evaluator determined that the guidance contains all the administrative actions and their associated audit events that are relevant to the claimed PP, PP-Modules, and Functional Packages. These administrative actions are found to be consistent with the actions specified in the [ST].

Test Assurance Activities

Assurance Activity AA-WLANPPM-FAU_GEN.1-WLAN-ATE-01

The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records in accordance with the assurance activities associated with the functional requirements in this PP-Module. 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.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly. For example, testing performed to ensure that the administrative guidance provided is correct verifies that AGD_OPE.1 is satisfied and should address the invocation of the administrative actions that are needed to verify the audit records are generated as expected.

Summary

The evaluator triggered all events listed in the [ST], then used the unified logging system to view the logs. The evaluator verified that accurate audit logs are generated with content and formatting consistent with those described in the TSS.

2.2.2 Cryptographic support (FCS)

2.2.2.1 Cryptographic Key Generation (FCS_CKM.1)

TSS Assurance Activities

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

[TD0712] 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. If "P-256" is selected, the evaluator will examine the TSS to verify that it is only used for Bluetooth functions.

Summary

Section 7.1.2.1 of the [ST] identifies the key sizes and usage for each of the supported schemes (RSA and ECC).

Guidance Assurance Activities

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

[TD0712] 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

Section 1.7 of the [CCGUIDE] states that no configuration is required for the cryptographic engine.

Test Assurance Activities

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

[TD0712, TD0955] Evaluation 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 or FIPS PUB 186-5 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 all be provable primes*
   - *Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes*
   - *Primes p1, p2, q1,q2, p and q shall all be probable primes*

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator 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:

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 or FIPS 186-5 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 or FIPS 186-5 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

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

2.2.2.2 Cryptographic Key Generation (Symmetric Keys for WPA2/WPA3 Connections) (FCS_CKM.1/WPA)

TSS Assurance Activities

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

The evaluator shall verify that the TSS describes how the primitives defined and implemented by this PP-Module are used by the TOE in establishing and maintaining secure connectivity to the wireless clients. The TSS shall also provide a description of the developer's method(s) of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also any third-party testing that is performed.

Summary

Section 7.1.2.1 of the [ST] describes the WLAN protocol. The TOE uses PRF-384 and PRF-704 to generate AES keys, and the AES-CCMP-256 and AES-GCMP-256 cryptographic algorithms to encrypt data traffic.

This section also describes the testing employed by the developer, both internal and third-party, on the cryptographic algorithm and security protocol implementations. The evaluator verified that this detailed description provides sufficient assurance that the TOE implementation conforms to the Wi-Fi standards.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

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

[TD0916] The evaluator shall perform the following tests:

- *Test 1: The evaluator shall configure the access point so the cryptoperiod of the session key is 1 hour. The evaluator shall successfully connect the TOE to the access point and maintain the connection for a length of time that is greater than the configured cryptoperiod. The evaluator shall use a packet capture tool to determine that after the configured cryptoperiod, a re-negotiation is initiated to establish a new session key. Finally, the evaluator shall determine that the renegotiation has been successful and the client continues communication with the access point.*

- *Test 2: The evaluator shall perform the following test using a packet sniffing tool to collect frames between the TOE and a wireless LAN access point:*

  Step 1: The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.

  Step 2: The evaluator shall configure the TOE to communicate with a WLAN access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-f) pre-shared key. The pre-shared key is only used for testing.

  Step 3: The evaluator shall start the sniffing tool, initiate a connection between the TOE and the access point, and allow the TOE to authenticate, associate, and successfully complete the 4-way handshake with the client.

  Step 4: The evaluator shall use the TOE client or access point to generate unicast traffic to the other wireless participant on the network. The evaluator shall disconnect the TOE from the wireless network and stop the sniffer.

  Step 5: The evaluator shall identify the four-way handshake frames (denoted EAPOL-key in Wireshark captures) and capture or derive the PTK from the four-way handshake frames and pre-shared key as specified in IEEE 802.11-2020.

  Step 6: The evaluator shall locate frames after the 4-way handshake with a frame control type of 0x2 (data), a frame control subtype between 0-3 (inclusive), and between 8-11 (inclusive) [i.e. "data" and "QoS data" frames containing a non-null payload], and the frame control flag includes at least bits 0x42 ("Protected frame" flag is set and "from DS" flag is set). In addition, the wireless frame "receiver" station address must be a unicast MAC address. These frames are expected to be protected using the PTK. The evaluator shall use the PTK to decrypt the data portion of the packet as specified in IEEE 802.11-2020 and verify that the decrypted data results in well-defined packets, including the packets that were sent by the evaluator in step 4.

Summary

The evaluator started an access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

Test 1: The evaluator verified that the 4-way EAPOL handshake was automatically triggered after one hour.

Test 2: The evaluator computed the PTK from the four-way handshake frames and verified that unicast frames were successfully decrypted.

2.2.2.3 Cryptographic Key Establishment (FCS_CKM.2)

TSS Assurance Activities

Assurance Activity AA-OSPP-FCS_CKM.2-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

Section 7.1.2.2 of the [ST] identifies the supported key establishment schemes and their usages (RSA-based and elliptic curve-based key establishment). The evaluator verified that these schemes correspond to the key generation schemes identified in FCS_CKM.1.

Guidance Assurance Activities

Assurance Activity AA-OSPP-FCS_CKM.2-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

Section 1.7 of the [CCGUIDE] states that no configuration is required for the cryptographic engine.

Test Assurance Activities

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

Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Establishment Schemes

The evaluator 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 will 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 will generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FCC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator 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 will 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 Revision 3 key agreement implementation to determine which errors the OS should be able to recognize. The evaluator generates a set of 24 FCC or 30 ECC 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 MACed, 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 will remain unmodified and therefore should result in valid key agreement results (they should pass).

The OS will 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.

RSA-based key establishment

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

Summary

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

The RSAES-PKCS1-v1_5 key establishment algorithm was tested by using a known good implementation (OpenSSL 3) and connecting to the TLS client implemented by the TOE (see FCS_TLSC_EXT.1/WLAN). The successful connection indicates that the algorithm is implemented correctly.

2.2.2.4 Cryptographic Key Distribution (Group Temporal Key for WLAN) (FCS_CKM.2/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FCS_CKM.2-WLAN-ASE-01

The evaluator shall check the TSS to ensure that it describes how the GTK is unwrapped prior to being installed for use on the TOE using the AES implementation specified in this PP-Module.

Summary

Section 7.1.2.3 of the [ST] describes how the GTK is unwrapped using AES Key Wrap.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FCS_CKM.2-WLAN-ATE-01

[TD0916] The evaluator shall perform the following test using a packet sniffing tool to collect frames between the TOE and a wireless access point (which may be performed in conjunction with the assurance activity for FCS_CKM.1.1/WLAN).

Step 1: The evaluator shall configure the access point to an unused channel and configure the WLAN sniffer to sniff only on that channel (i.e., lock the sniffer on the selected channel). The sniffer should also be configured to filter on the MAC address of the TOE and/or access point.

Step 2: The evaluator shall configure the TOE to communicate with the access point using IEEE 802.11-2012 and a 256-bit (64 hex values 0-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.

Step 3: The evaluator shall start the sniffing tool, initiate a connection between the TOE and access point, and allow the TOE to authenticate, associate, and successfully complete the 4-way handshake

with the TOE.

(Step 4 may require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.)

Step 4: The evaluator shall use the TOE client or the access point to generate multicast or broadcast traffic to the other wireless participant on the network. The evaluator will disconnect the TOE from the access point and stop the sniffer.

Step 5: The evaluator shall identify the 4-way handshake frames (denoted EAPOL-key in Wireshark captures) and capture or derive the PTK and GTK from the 4-way handshake frames and pre-shared key as specified in IEEE 802.11-2012.

Step 6: The evaluator shall locate frames after the 4-way handshake with a frame control type of 0x2 (data), a frame control subtype between 0-3 (inclusive), and between 8-11 (inclusive) [i.e. "data" and "QoS data" frames containing a non-null payload], and the frame control flag includes at least bits 0x42 ("Protected frame" flag is set and "from DS" flag is set). In addition, the wireless frame "receiver" station address must be a multicast or broadcast MAC address. These frames are expected to be protected using the GTK. The evaluator shall use the GTK to decrypt the data portion of the selected packet as specified in IEEE 802.11-2020 and verify that the decrypted data results in well-defined packets, including the packets that were sent by the evaluator in step 4.

Summary

The evaluator started an access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

The evaluator computed the GTK from the four-way handshake frames and verified that broadcast frames were successfully decrypted.

2.2.2.5 Key Destruction (FCS_CKM_EXT.4)

TSS Assurance Activities

Assurance Activity AA-OSPP-FCS_CKM_EXT.4-ASE-01

[TD0701] 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 will check to ensure the TSS lists each type of key that is stored in 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).

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 will verify that the pattern does not contain any CSPs.

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

If the selection “destruction of all key encrypting keys (KEKs) protecting the target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived” is included the evaluator will examine the TOE’s keychain in the TSS and identify each instance when a key is destroyed by this method. In each instance the evaluator will verify all keys capable of decrypting the target key are destroyed in accordance with a specified key destruction method in FCS_CKM_EXT.4.1. The evaluator will verify that all of the keys capable of decrypting the target key are not able to be derived to reestablish the keychain after their destruction.

Summary

Section 7.1.2.4 of the [ST] states that the TOE includes the Keychain Access app that allows users the ability to add, remove, and manage certificates, shared secrets, and private keys. The [ST] describes Data Encryption Keys (DEKs) and Key Encryption Keys (KEKs), as well as specifically detailing TLS private keys, Bluetooth SSP private keys, and WLAN keys. The evaluator confirmed that the TSS describes, in sufficient detail, the method of introduction to and management of the keys in volatile memory, as well as how the keys are stored in non-volatile memory, and how the TOE interacts with the platform to manage the keys.

Wrapped keys held in non-volatile storage are cryptographically destroyed by invoking an interface provided by the underlying platform that destroys the abstraction that represents the key. Keys held in volatile memory are destroyed via single overwrite consisting of zeroes.

The [ST] does not make use of the open assignment, overwrite is performed using zeroes.

The [ST] does not specify any configurations or circumstances that may not conform to the key destruction requirements.

The [ST] does not select “destruction of all key encrypting keys (KEKs) protecting the target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived”.

Guidance Assurance Activities

Assurance Activity AA-OSPP-FCS_CKM_EXT.4-AGD-01

There are a variety of concerns that may prevent or delay key destruction in some cases. The evaluator will 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 will 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 1.1 of the [CCGUIDE] states that the TOE is running on Mac computers with custom Apple silicon, including the Secure Enclave. The Secure Enclave Processor operating system (sepOS) is included with macOS and is within the TOE boundary. Section 5 of the [CCGUIDE] explains that the Key Encryption Key used to protect Keychain keys is stored in the Secure Enclave. This ensures that there are no instances where key destruction may be delayed.

Test Assurance Activities

Assurance Activity AA-OSPP-FCS_CKM_EXT.4-ATE-01

- *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 will:*

  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 will 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.

- *Test 3:*

  Tests 3 and 4 do not apply for the selection instructing the underlying platform to destroy the representation of the key as the TOE has no visibility into the inner workings and completely relies on the underlying platform.

  The following tests are used to determine if the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.

  Applied to each key held in non-volatile memory and subject to destruction by overwrite by the TOE. The evaluator will 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 will 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

The evaluator used a tool provided by the vendor to read the checksums of the media key and Class D key, as well as the hash values of the class keys, key encryption keys, and data encryption keys loaded by the Secure Enclave.

The evaluator performed a factory reset of the TOE, which caused the TOE to erase all keys. Then, the evaluator again read the checksums of the media key and the Class D key, and the hash values of the keys loaded by the Secure Enclave. The evaluator found that the checksums and hash values changed, proving that the old keys were erased and new keys were generated by the Secure Enclave.

To verify the zeroization of TLS session keys in volatile memory, the evaluator inspected the memory of a TLS session using LLDB. The evaluator found that session keys were zeroized when no longer needed.

2.2.2.6 Bluetooth Key Generation (FCS_CKM_EXT.8)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FCS_CKM_EXT.8-ASE-01

The evaluator shall ensure that the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs. In particular, the evaluator shall ensure that the implementation does not permit the use of static ECDH key pairs.

Summary

Section 7.1.2.5 of the [ST] states that the TOE generates a new ECDH key pair for every new connection attempt. The TOE does not permit the use of static ECDH key pairs.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-BTPPM-FCS_CKM_EXT.8-ATE-01

The evaluator shall perform the following steps:

Step 1: Pair the TOE to a remote Bluetooth device and record the public key currently in use by the TOE. (This public key can be obtained using a Bluetooth protocol analyzer to inspect packets exchanged during pairing.)

Step 2: Perform necessary actions to generate new ECDH public/private key pairs. (Note that this test step depends on how the TSS describes the criteria used to determine the frequency of generating new ECDH public/private key pairs.)

Step 3: Pair the TOE to a remote Bluetooth device and again record the public key currently in use by the TOE. Step 4: Verify that the public key in Step 1 differs from the public key in Step 3.

Summary

The evaluator paired the TOE twice with a remote Bluetooth device. After both pairings, the evaluator collected the Bluetooth packet logs from the TOE. Using Wireshark, the evaluator recovered the public keys used for Bluetooth and confirmed the public keys were different.

2.2.2.7 Cryptographic Operation - Encryption/Decryption (FCS_COP.1/ENCRYPT)

TSS Assurance Activities

Assurance Activity AA-OSPP-FCS_COP.1-ENCRYPT-ASE-01

[TD0712] If "128-bit" is selected, the evaluator will examine the TSS to verify that 128-bit is only used with AES-CCM for Bluetooth functions.

Summary

Section 7.1.2.6 of the [ST] states that the 128-bit key is only used with AES-CCM for Bluetooth functions. This is also confirmed by reviewing the tables in Section 7.1.2 of the [ST].

Guidance Assurance Activities

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

The evaluator will verify that the AGD documents contains instructions required to configure the OS to use the required modes and key sizes.

Summary

Section 1.7 of the [CCGUIDE] states that no configuration is required for the cryptographic engine.

Test Assurance Activities

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

[TD0712] The evaluator will execute all instructions as specified to configure the OS to the appropriate state. 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 will be 256-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.

- *Test 5: To test the encrypt functionality of AES-CBC, the evaluator will supply a set of 5 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. The plaintext values will 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 5 ciphertext values as input and AES-CBC decryption.*
- *Test 6: To test the encrypt functionality of AES-CBC, the evaluator will supply a set of five 256-keys 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. 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.*
- *Test 7: To test the encrypt functionality of AES-CBC, the evaluator will supply the a 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. Key I will 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 set 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 set of key/ciphertext pairs*

will have 256 256-bit key/ciphertext pairs. Key i in each set will 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 will be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

- *Test 8: To test the encrypt functionality of AES-CBC, the evaluator will supply the set of 256 plaintext values described below and obtain the ciphertext values that result from AES-CBC encryption of the given plaintext using a 256-bit key value of all zeros with an IV of all zeros. Plaintext value i in each set will have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1,256].*

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 shall 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 will 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 will 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 100 plaintext, IV, and key 3-tuples. The keys, plaintext, and IV values are each 256-bits. For each 3-tuple, 1000 iterations will 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 will 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-CTR Test

Known Answer Tests (KATs)

There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, initialization vector (IV), and ciphertext values shall be 256-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- *Test 9: To test the encrypt functionality, the evaluator will supply 5 plaintext values and obtain the ciphertext value that results from encryption of the given plaintext using a 256-bit key value of all zeros and an IV of all zeros. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using the 5 ciphertext values as input.*

- *Test 10: To test the encrypt functionality, the evaluator will supply 5 256-bit key values and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value and an IV of all zeros. To test the decrypt functionality, the evaluator will perform the same test as for encrypt, using an all zero ciphertext value as input.*
- *Test 11: To test the encrypt functionality, the evaluator will supply a set of key values described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values and an IV of all zeros. The set of keys shall have shall have 256 256-bit keys. Keyi shall have the leftmost i bits be ones and the rightmost 256-i bits be zeros, for i in [1, N]. To test the decrypt functionality, the evaluator will supply the set of key and ciphertext value pairs described below and obtain the plaintext value that results from decryption of the given ciphertext using the given key values and an IV of all zeros. The set of key/ciphertext pairs shall have 256 256-bit pairs. Keyi shall have the leftmost i bits be ones and the rightmost 256-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.*
- *Test 12: To test the encrypt functionality, the evaluator will supply the set of 256 plaintext values described below and obtain the two ciphertext values that result from encryption of the given plaintext using a 256 bit key value of all zeros, respectively, and an IV of all zeros. Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 256-i bits be zeros, for i in [1, 256]. To test the decrypt functionality, 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.*

Multi-Block Message Test

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

Monte-Carlo Test

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

The evaluator will test the encrypt functionality using 100 plaintext/key pairs. Each key shall be 256-bit. The plaintext values shall be 256-bit blocks. For each pair, 1000 iterations shall be run as follows:

For AES-ECB mode

 # Input: PT, Key

for i = 1 to 1000:

CT[i] = AES-ECB-Encrypt(Key, PT)

PT = CT[i]

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

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:

- *256 bit key encryption keys (KEKs)*
- *Three plaintext lengths. One of the plaintext lengths will be two semi-blocks (256 bits). One*

of the plaintext lengths will be three semi-blocks (192 bits). The third data unit length will 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.

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 (if selected) and 256 bit keys*
- *Two payload lengths. One payload length will be the shortest supported payload length, greater than or equal to zero bytes. The other payload length will 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 will be 0, if supported. One associated data length will be the shortest supported payload length, greater than or equal to zero bytes. One associated data length will 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 will be tested.*
- *Nonce lengths. The evaluator will test all nonce lengths between 7 and 13 bytes, inclusive, that are supported by the OS.*
- *Tag lengths. The evaluator will test all of the following tag length values that are supported by the OS: 4, 6, 8, 10, 12, 14 and 16 bytes.*

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

- *Test 13: 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 14: 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 15: 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 16: 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 will 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-GCMP Test

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

- *256 bit keys*
- *Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.*
- *Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.*
- *Two IV lengths. If 96 bit IV is supported, 96 bits will 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 will 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 will 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-GCMP Monte Carlo Tests

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

- *256 bit keys*
- *Two plaintext lengths. One of the plaintext lengths will be a non-zero integer multiple of 256 bits, if supported. The other plaintext length will not be an integer multiple of 256 bits, if supported.*
- *Three AAD lengths. One AAD length will be 0, if supported. One AAD length will be a non-zero integer multiple of 256 bits, if supported. One AAD length will not be an integer multiple of 256 bits, if supported.*
- *Two IV lengths. If 96 bit IV is supported, 96 bits will 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 will 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 will 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.

Summary

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

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

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FCS_COP.1-HASH-ATE-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.

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 test MACs. 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 17: 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 will 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 18: 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 will 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 19: 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 will 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 20: 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 will 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 21: 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

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

2.2.2.9 Cryptographic Operation - Signing (FCS_COP.1/SIGN)

TSS Assurance Activities

Assurance Activity AA-OSPP-FCS_COP.1-SIGN-ASE-01

[TD0955] [Conditional: if "2048-bit (for secure boot only) or greater" is selected] The evaluator shall check that the TSS documents that 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

Summary

The [ST] does not select "2048-bit (for secure boot only) or greater".

Guidance Assurance Activities

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

[TD0955] [Conditional: if "2048-bit (for secure boot only) or greater" is selected] The evaluator shall check that the AGD documents any configuration needed to ensure 2048-bit RSA is used only for secure boot and a greater key size is used for any other functions.

Summary

The [ST] does not select "2048-bit (for secure boot only) or greater".

Test Assurance Activities

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

[TD0955]

ECDSA Algorithm Tests

- *Test 22: ECDSA FIPS 186-4, FIPS 186-5, SP 800-186 Section 3 Signature Generation Test. For each supported NIST curve (i.e., 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 23: ECDSA FIPS 186-4, FIPS 186-5, SP 800-186 Section 3 Signature Verification Test. For each supported NIST curve (i.e., 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 24: 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' signature using a known good implementation and the associated public keys to verify the signatures.*

- *Test 25: 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

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

2.2.2.10 Cryptographic Operation - Keyed-Hash Message Authentication (FCS_COP.1/KEYHMAC)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FCS_COP.1-KEYHMAC-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 consists 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 will be compared against the result of generating HMAC tags with the same key using a known-good implementation.

Summary

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

2.2.2.11 Random Bit Generation (FCS_RBG_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

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

The evaluator will perform the following tests:

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 list contains more information on some of the input values to be generated/selected by the evaluator.

- ***Entropy input:** The length of the entropy input value must equal the seed length.*
- ***Nonce:** If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.*
- ***Personalization string:** The length of the personalization string must be less 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.*

Documentation will be produced - and the evaluator will perform the activities - in accordance with Appendix E - Entropy Documentation and Assessment 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 verified that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the DRBG algorithms, and the certificate information is provided in Section 2.1 of this document.

Additionally, the evaluator reviewed the vendor's proprietary Entropy Assessment Report and performed the activities in Appendix E.

2.2.2.12 Storage of Sensitive Data (FCS_STO_EXT.1)

TSS Assurance Activities

Assurance Activity AA-OSPP-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. The evaluator will confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1/ENCRYPT.

Summary

Section 7.1.2.11 of the [ST] details each piece of sensitive data stored in the keychain, the purpose for which they are used, and how they are stored. Keychain items are encrypted using two different AES-256-GCM keys, and these cryptographic operations are implemented as specified in FCS_COP.1/ENCRYPT.

Guidance Assurance Activities

Assurance Activity AA-OSPP-FCS_STO_EXT.1-AGD-01

The evaluator will consult the developer documentation to verify that instructions exists on applications should securely store credentials.

Summary

Section 5.2 of the [CCGUIDE] explains how applications can use the Keychain Services API to securely store credentials in a keychain.

Test Assurance Activities

No assurance activities defined.

2.2.2.13 TLS Protocol (FCS_TLS_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLS_EXT.1-AGD-01

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

Summary

The evaluator notes that the [ST] selects “TLS as a client”. Consequently, the evaluator verified that FCS_TLSC_EXT.1 is claimed in [ST] as required.

Test Assurance Activities

No assurance activities defined.

2.2.2.14 TLS Client Protocol (FCS_TLSC_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1-ASE-01

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component. The evaluator shall ensure that the TSS describes the client’s method of establishing all reference identifiers from the application-configured reference identifier, including which types of reference identifiers are supported (e.g. Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the product.

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

Summary

Section 7.1.2.12 of the [ST] describes the TLS protocol and cipher suites supported by the TOE. The evaluator verified that those cipher suites exactly matched those listed for this component. This section also describes the client's method of establishing reference identifiers. The reference identifier used for comparison is the DNS Name or the IP address in the Subject Alternative Name (SAN) extension. Finally, this section confirms that the TSF supports wildcards in the DNS name of the server certificate. Certificate pinning is supported for users of the TLS framework but not used by default by the TOE.

The evaluator verified the override mechanism is discussed in the TSS, where it is detailed that specifically when the TLS framework is used, the didReceiveChallenge method in URLSession can be used to override the default trust handling of TLS server certificates. Override decisions are not stored by the TOE.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1-AGD-01

The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the product so that TLS conforms to the description in the TSS. The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Summary

Section 4 of the [CCGUIDE] explains that no configuration is required for TLS.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1-ATE-01

[TD0499, TD0513] The evaluator shall also perform the following tests:

- *Test FCS_TLSC_EXT.1:1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).*
- *Test FCS_TLSC_EXT.1:2: The goal of the following test is to verify that the TOE accepts only certificates with appropriate values in the extendedKeyUsage extension, and implicitly that the TOE correctly parses the extendedKeyUsage extension as part of X.509v3 server certificate validation.*

The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator shall repeat this test using a different, but otherwise valid and trusted, certificate that lacks the Server Authentication purpose in the extendedKeyUsage extension and ensure that a connection is not established. Ideally, the two certificates should be similar in structure, the types of identifiers used, and the chain of trust.

- *Test FCS_TLSC_EXT.1:3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator shall verify that the product disconnects after receiving the server's Certificate handshake message.*
- *Test FCS_TLSC_EXT.1:4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.*
- *Test FCS_TLSC_EXT.1:5: The evaluator shall perform the following modifications to the*

traffic:

- *Test FCS_TLSC_EXT.1:5.1: Change the TLS version selected by the server in the Server Hello to an undefined TLS version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.*
- *Test FCS_TLSC_EXT.1:5.2: Change the TLS version selected by the server in the Server Hello to the most recent unsupported TLS version (for example 1.1 represented by the two bytes 03 02) and verify that the client rejects the connection.*
- *Test FCS_TLSC_EXT.1:5.3: [conditional] If DHE or ECDHE cipher suites are supported, modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client does not complete the handshake and no application data flows.*
- *Test FCS_TLSC_EXT.1:5.4: Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client does not complete the handshake and no application data flows.*
- *Test FCS_TLSC_EXT.1:5.5: [conditional] If DHE or ECDHE cipher suites are supported, modify the signature block in the server's Key Exchange handshake message, and verify that the client does not complete the handshake and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.*
- *Test FCS_TLSC_EXT.1:5.6: Modify a byte in the Server Finished handshake message, and verify that the client does not complete the handshake and no application data flows.*
- *Test FCS_TLSC_EXT.1:5.7: Send a message consisting of random bytes from the server after the server has issued the Change Cipher Spec message and verify that the client does not complete the handshake and no application data flows. The message must still have a valid 5-byte record header in order to ensure the message will be parsed as TLS.*

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

- *Test FCS_TLSC_EXT.1:6: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails.*

Note that some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

- *Test FCS_TLSC_EXT.1:7: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type.*
- *Test FCS_TLSC_EXT.1:8: [conditional] If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.*
- *Test FCS_TLSC_EXT.1:9: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds.*
- *Test FCS_TLSC_EXT.1:10: The evaluator shall perform the following wildcard tests with each supported type of reference identifier. The support for wildcards is intended to be optional. If wildcards are supported, the first, second, and third tests below shall be executed. If wildcards are not supported, then the fourth test below shall be executed.*

- *Test FCS_TLSC_EXT.1:10.1: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.*
- *Test FCS_TLSC_EXT.1:10.2: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.come) and verify that the connection fails.*
- *Test FCS_TLSC_EXT.1:10.3: [conditional]: If wildcards are supported, the evaluator shall present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.*
- *Test FCS_TLSC_EXT.1:10.4: [conditional]: If wildcards are not supported, the evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection fails.*

- *Test FCS_TLSC_EXT.1:11: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.*
- *Test FCS_TLSC_EXT.1:12: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.*

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

- *Test FCS_TLSC_EXT.1:13:*
  - *Test 1a: The evaluator shall demonstrate that a server using a certifcate with a valid certification path successfully connects.*
  - *Test 1b: The evaluator shall modify the certificate chain used by the server in test 1a to be invalid and demonstrate that a server using a certificate without a valid certification path to a trust store element of the TOE results in an authentication failure.*
  - *Test 1c [conditional]: If the TOE trust store can be managed, the evaluator shall modify the trust store element used in Test 1a to be untrusted and demonstrate that a connection attempt from the same server used in Test 1a results in an authentication failure.*
- *Test FCS_TLSC_EXT.1:14: The evaluator shall demonstrate that a server using a certificate which has been revoked results in an authentication failure.*
- *Test FCS_TLSC_EXT.1:15: The evaluator shall demonstrate that a server using a certificate which has passed its expiration date results in an authentication failure.*
- *Test FCS_TLSC_EXT.1:16: The evaluator shall demonstrate that a server using a certificate which does not have a valid identifier results in an authentication failure.*

Summary

The evaluator started a remote TLS server using the OpenSSL library. All packets were logged using Wireshark. A test application was used to instruct the TOE to perform TLS connections.

The CA certificate used to issue the certificate of the TLS server was trusted by the TOE.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Test FCS_TLSC_EXT.1:8: Not applicable as the TOE mandates the presence of a SAN extension.

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

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

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

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

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

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

Test FCS_TLSC_EXT.1:12: The evaluator created a valid server certificate, though not identical to the pinned certificate, and verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:13a: The evaluator created a valid server certificate and imported the issuer CA certificate into the TOE certificate store, thereby creating a valid certification path. The evaluator verified the connection was successfully established.

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

Test FCS_TLSC_EXT.1:13c: The evaluator created a valid server certificate, identical to the certificate created in Test 1a, but without importing the issuer CA certificate into the TOE certificate store, thereby creating an invalid certification path. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:14: The evaluator started an OCSP server and configured the server certificate to be marked as “revoked”. The evaluator verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:15: The evaluator created an expired server certificate and verified the connection was rejected as expected.

Test FCS_TLSC_EXT.1:16: The evaluator created a server certificate with a correct Common Name but with an incorrect DNS name entry in the SAN extension and verified the connection was rejected as expected.

2.2.2.15 TLS Client Protocol (EAP-TLS for WLAN) (FCS_TLSC_EXT.1/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FCS_TLSC_EXT.1-WLAN-ASE-01

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

Summary

Section 7.1.2.13 of the [ST] describes the EAP-TLS protocol and cipher suites supported by the TOE. The evaluator verified that those cipher suites exactly matched those listed for this component.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FCS_TLSC_EXT.1-WLAN-AGD-01

The evaluator shall check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS (for instance, the set of cipher suites advertised by the TOE may have to be restricted to meet the requirements).

The evaluator shall check that the guidance contains instructions for the administrator to configure the list of Certificate Authorities that are allowed to sign certificates used by the authentication server that will be accepted by the TOE in the EAP-TLS exchange, and instructions on how to specify the algorithm suites that will be proposed and accepted by the TOE during the EAP-TLS exchange.

Summary

Section 4 of the [CCGUIDE] explains that no configuration is required for TLS. Section 5.1 contains instructions for users to manage the CA certificate store using the Keychain Access app.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FCS_TLSC_EXT.1-WLAN-ATE-01

The evaluator shall write, or the TOE developer shall provide, an application for the purposes of testing TLS.

The evaluator shall perform the following tests:

- *Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).*
- *Test 2: The evaluator shall 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 is 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 shall send a server certificate in the TLS connection that does not match the server-selected cipher suite. For example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites. The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.*
- *Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.*
- *Test 5: The evaluator shall perform the following modifications to the traffic:*
  - *Change the TLS version selected by the server in the Server Hello to a unsupported TLS*

version (for example 1.5 represented by the two bytes 03 06) and verify that the client rejects the connection.

- *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.*
- *Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.*
- *[conditional: if the TOE supports at least one cipher suite that uses DHE or ECDHE for key exchange] 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. This test does not apply to cipher suites using RSA key exchange.*
- *Modify a byte in the Server Finished handshake message, and verify that the client sends an Encrypted Message followed by a FIN and ACK message. This is sufficient to deduce that the TOE responded with a Fatal Alert and no further data would be sent.*
- *Send a garbled message from the server after the server has issued the ChangeCipherSpec message and verify that the client denies the connection.*

Summary

The evaluator started an access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

The CA certificate used to issue the certificate of the access point EAP-TLS server was trusted by the TOE.

Finally, the evaluator instructed the TOE to initiate a connection to the access point. Depending on the specific test, the connection succeeded or failed:

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

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

Test 3: The evaluator created an RSA-based server certificate but patched the OpenSSL library used by hostapd to select an ECDSA-based cipher suite, and verified the connection was rejected as expected.

Test 4: The evaluator patched the OpenSSL library used by hostapd to always select the TLS_NULL_WITH_NULL_NULL cipher suite, regardless of the support indicated by the client, and verified the connection was rejected as expected.

Test 5.1: The evaluator patched the OpenSSL library used by hostapd to select the "03 05" TLS version to the client, corresponding to a non-existent TLS 1.4 version, regardless of the support indicated by the client, and verified the connection was rejected as expected.

Test 5.2: The evaluator patched the OpenSSL library used by hostapd to modify the nonce of the server and verified the connection was rejected as expected.

Test 5.3: The evaluator patched the OpenSSL library used by hostapd to modify the selected server cipher suite and verified the connection was rejected as expected.

Test 5.4: The evaluator patched the OpenSSL library used by hostapd to modify the signature in the Server Key Exchange message and verified the connection was rejected as expected.

Test 5.5: The evaluator patched the OpenSSL library used by hostapd to modify the message digest in the Server Finished message and verified the connection was rejected as expected.

Test 5.6: The evaluator patched the OpenSSL library used by hostapd to send another Server Key Exchange message after the Change Cipher Spec message and verified the connection was rejected as expected.

2.2.2.16 TLS Client Support for Mutual Authentication (FCS_TLSC_EXT.2)

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.2-ASE-01

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

Summary

Section 7.1.2.12 of the [ST] describes the mutual authentication supported by the TOE. No factors beyond configuration are necessary in order for the TOE to engage in mutual authentication.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.2-AGD-01

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

Summary

Section 4 of the [CCGUIDE] explains that no configuration is required for TLS. Section 5.1 contains instructions for users to manage client-side certificates using the Keychain Access app.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.2-ATE-01

The evaluator shall also perform the following tests:

- *Test 1: The evaluator shall establish a connection to a server that is not configured for mutual authentication (i.e. does not send Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE did not send Client's Certificate message (type 11) during handshake.*
- *Test 2: The evaluator shall establish a connection to a server with a shared trusted root that is configured for mutual authentication (i.e. it sends Server's Certificate Request (type 13) message). The evaluator observes negotiation of a TLS channel and confirms that the TOE responds with a non-empty Client's Certificate message (type 11) and Certificate Verify (type 15) message.*

Summary

The evaluator started a remote TLS server using the OpenSSL library. All packets were logged using Wireshark. A test application was used to instruct the TOE to perform TLS connections.

The CA certificate used to issue the certificate of the TLS server was trusted by the TOE.

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

Test 1: The evaluator configured the TLS server to not request a client certificate and verified the TOE did not send the Client's Certificate message.

Test 2: The evaluator configured the TLS server to request a client certificate and verified the TOE did send the Client's Certificate and Certificate Verify messages.

2.2.2.17 TLS Client Support for Supported Groups Extension (EAP-TLS for WLAN) (FCS_TLSC_EXT.2/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FCS_TLSC_EXT.2-WLAN-ASE-01

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

Summary

Section 7.1.2.13 of the [ST] describes the Supported Groups extension. The behavior is performed by default.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FCS_TLSC_EXT.2-WLAN-AGD-01

If the TSS indicates that the Supported Groups extension must be configured to meet the requirement, the evaluator shall verify that the operational guidance includes instructions for configuration of this extension.

Summary

Section 7.1.2.13 of the [ST] describes the Supported Groups extension. The behavior is performed by default.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FCS_TLSC_EXT.2-WLAN-ATE-01

The evaluator shall perform the following test:

- *Test 1: The evaluator shall configure a server to perform ECDHE key exchange using each of the TOE's supported curves and shall verify that the TOE successfully connects to the server.*

Summary

The evaluator started an access point using the hostapd application, configured to use a specific group or curve. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

The CA certificate used to issue the certificate of the access point EAP-TLS server was trusted by the TOE.

Finally, the evaluator instructed the TOE to initiate a connection to the access point. The evaluator verified the connection was successfully established using the group / curve.

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

2.2.2.18 TLS Client Support for Renegotiation (FCS_TLSC_EXT.4)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.4-ATE-01

The evaluator shall perform the following tests:

- *Test 1: The evaluator shall use a network packet analyzer/sniffer to capture the traffic between the two TLS endpoints. The evaluator shall verify that either the "renegotiation_info" field or the SCSV cipher suite is included in the ClientHello message during the initial handshake.*
- *Test 2: The evaluator shall verify the Client's handling of ServerHello messages received during the initial handshake that include the "renegotiation_info" extension. The evaluator shall modify the length portion of this field in the ServerHello message to be non-zero and verify that the client sends a failure and terminates the connection. The evaluator shall verify that a properly formatted field results in a successful TLS connection.*
- *Test 3: The evaluator shall verify that ServerHello messages received during secure renegotiation contain the "renegotiation_info" extension. The evaluator shall modify either the "client_verify_data" or "server_verify_data" value and verify that the client terminates the connection.*

Summary

The evaluator started a remote TLS server using the OpenSSL library. All packets were logged using Wireshark. A test application was used to instruct the TOE to perform TLS connections.

The CA certificate used to issue the certificate of the TLS server was trusted by the TOE.

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

Test 1: The evaluator verified the connection succeeded, and the renegotiation_info field is included in the Client Hello message during the initial handshake.

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

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

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

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.5-ASE-01

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

Summary

Section 7.1.2.12 of the [ST] describes the Supported Groups extension.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.5-ATE-01

The evaluator shall also perform the following test:

- *Test 1: The evaluator shall configure a server to perform key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.*

Summary

The evaluator started a remote TLS server using the OpenSSL library, configured to use a specific group or curve. All packets were logged using Wireshark. A test application was used to instruct the TOE to perform TLS connections.

The CA certificate used to issue the certificate of the TLS server was trusted by the TOE.

Finally, the evaluator instructed the TOE to initiate a TLS request to the TLS server. The evaluator verified the connection was successfully established using the group / curve.

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

2.2.2.20 Supported WPA Versions (FCS_WPA_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FCS_WPA_EXT.1-AGD-01

[TD0710] The evaluator shall ensure that the AGD contains guidance on how to configure the WLAN client to connect to networks supporting WPA3 and, if selected, WPA2.

Summary

Section 3.11 of the [CCGUIDE] contains instructions for the user to enable Wi-Fi and join networks. There is no specific configuration required for WPA2 or WPA3.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FCS_WPA_EXT.1-ATE-01

[TD0710] The evaluator shall configure a Wi-Fi network that utilizes WPA3 and verify that the client can connect. The same test shall be repeated for WPA2 if it is selected.

Summary

The evaluator started a WPA3 access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

The CA certificate used to issue the certificate of the access point EAP-TLS server was trusted by the TOE.

Finally, the evaluator instructed the TOE to initiate a connection to the access point. The evaluator verified the connection was successfully established.

The steps above were repeated for WPA2.

2.2.3 User data protection (FDP)

2.2.3.1 Access Controls for Protecting User Data (FDP_ACF_EXT.1)

TSS Assurance Activities

Assurance Activity AA-OSPP-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

Section 7.1.3 of the [ST] describes the access control policy enforced by the OS, rules to determine access rights, and the order by which the rules are processed when determining those rights. The evaluator confirmed that the description in the TSS contains sufficient 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-OSPP-FDP_ACF_EXT.1-ATE-01

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

- *Test 26: 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 27: 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 28: 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 29: 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 30: 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 31: 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

The evaluator created two standard user accounts: testuser1 and testuser2. Then, the evaluator authenticated as testuser1 and created a file in testuser1's home directory: /Users/testuser1/file_testuser1.

Test 26: The evaluator authenticated to macOS as testuser2 and attempted to read /Users/testuser1/file_testuser1. The evaluator verified that the read attempt was denied.

Test 27: The evaluator authenticated to macOS as testuser2 and attempted to modify /Users/testuser1/file_testuser1. The evaluator verified that the modification attempt was denied.

Test 28: The evaluator authenticated to macOS as testuser2 and attempted to delete /Users/testuser1/file_testuser1. The evaluator verified that the deletion attempt was denied.

Test 29: The evaluator authenticated to macOS as testuser1 and attempted to create a file in /Users/testuser2. The evaluator verified that the file creation was denied.

Test 30: The evaluator authenticated to macOS as testuser1 and attempted to modify /Users/testuser1/file_testuser1. The evaluator verified that the file was successfully modified.

Test 31: The evaluator authenticated to macOS as testuser1 and attempted to delete /Users/testuser1/file_testuser1. The evaluator verified that the file was successfully deleted.

2.2.4 Identification and authentication (FIA)

2.2.4.1 Authentication Failure Handling (FIA_AFL.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FIA_AFL.1-ATE-01

[TD0691] 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.

- *Test 53 [conditional, to be performed if "authentication based on user name and password" is selected in FIA_AFL.1 and FIA_UAU.5]: 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 54 [conditional, to be performed if "authentication based on user name and a PIN that releases an asymmetric key stored in OE-protected storage" is selected in FIA_AFL.1 and FIA_UAU.5]: The evaluator will attempt to authenticate repeatedly to the system with a known bad PIN. 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 55 [conditional, to be performed if "authentication based on X.509 certificates" is selected in FIA_AFL.1 and FIA_UAU]: The evaluator will attempt to authenticate repeatedly to the system using 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.*

Summary

Test 53: The evaluator used pwpolicy to configure the threshold for failed attempts to four. Then, the evaluator attempted to authenticate with a bad password four times followed by one attempt with the correct password. The evaluator verified that the attempt to authenticate with the correct password was denied and that the event was logged.

Test 54: The evaluator used pwpolicy to configure the threshold for failed attempts to four. Then, the evaluator inserted a paired YubiKey and attempted to authenticate with a bad PIN four times followed by one attempt with the correct PIN. The evaluator verified that the attempt to authenticate with the correct PIN was denied and that the event was logged.

Test 55: Not applicable as the [ST] does not select “authentication based on X.509 certificates”.

2.2.4.2 Bluetooth User Authorization (FIA_BLT_EXT.1)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.1-ASE-01

The evaluator shall examine the TSS to ensure that it contains a description of when user permission is required for Bluetooth pairing; and that this description mandates explicit user authorization via manual input for all Bluetooth pairing; including application use of the Bluetooth trusted channel and situations where temporary (non-bonded) connections are formed.

Summary

Section 7.1.4.2 of the [ST] states that user authorization is always required for Bluetooth pairing. The user can match and accept a six-digit PIN, enter the PIN manually, or explicitly confirm the pairing with a peripheral that does not support PIN entry.

Guidance Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.1-AGD-01

The evaluator shall examine the guidance to verify that these user authorization screens are clearly identified and instructions are given for authorizing Bluetooth pairings.

Summary

Section 3.12.2 of the [CCGUIDE] provides instructions for authorizing Bluetooth pairings.

Test Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.1-ATE-01

The evaluator shall perform the following steps:

Step 1: Initiate pairing with the TOE from a remote Bluetooth device that requests no man-in-the-middle protection; no bonding; and claims to have NoInput/NoOutput (IO) capability. Such a device will attempt to evoke behavior from the TOE that represents the minimal level of user interaction

that the TOE supports during pairing.

Step 2: Verify that the TOE does not permit any Bluetooth pairing without explicit authorization from the user (e.g. the user must have to minimally answer "yes" or "allow" in a prompt).

Summary

The evaluator configured a Linux system to present itself as a NoInputNoOutput device and disabled bonding. All packets were logged using btmon. Then, the evaluator attempted to pair the Linux system with the TOE, found that the TOE requires the user to authorize the pairing by pressing “pair”. From the packet log, the evaluator confirmed that the pairing was not completed until the user authorized the pairing.

2.2.4.3 Bluetooth Mutual Authentication (FIA_BLT_EXT.2)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.2-ASE-01

The evaluator shall ensure that the TSS describes how data transfer of any type is prevented before the Bluetooth pairing is completed. The TSS shall specifically call out any supported RFCOMM and L2CAP data transfer mechanisms. The evaluator shall ensure that the data transfers are only completed after the Bluetooth devices are paired and mutually authenticated.

Summary

Section 7.1.4.3 of the [ST] states that the TOE will only transfer data to a remote Bluetooth device if it paired with that device. The evaluator confirmed that this section also calls out all supported RFCOMM and L2CAP data transfer mechanisms.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.2-ATE-01

The evaluator shall use a Bluetooth tool to attempt to access TOE files using the OBEX Object Push service (OBEX Push) and verify that pairing and mutual authentication are required by the TOE before allowing access. If the OBEX Object Push service is unsupported on the TOE; a different service that transfers data over Bluetooth L2CAP and/or RFCOMM may be used in this test.

Summary

The evaluator configured a Linux system to send a file via OBEX Push upon successful pairing. All packets were logged using btmon. The evaluator found that the TOE requires the user to authorize the pairing and inspected the packet log to verify that no data was transferred until the user authorized the pairing.

2.2.4.4 Rejection of Duplicate Bluetooth Connections (FIA_BLT_EXT.3)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.3-ASE-01

The evaluator shall ensure that the TSS describes how Bluetooth sessions are maintained such that at least two devices with the same Bluetooth device address are not simultaneously connected and such that the initial session is not superseded by any following session initialization attempts.

Summary

Section 7.1.4.4 of the [ST] states that the TOE will reject Bluetooth pairing attempts from a BD_ADDR for which an established connection exists. It describes in detail how duplicates are handled using a connection list, which is checked before a Bluetooth connection is accepted.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.3-ATE-01

The evaluator shall perform the following steps:

Step 1: Pair the TOE with a remote Bluetooth device (DEV1) with a known address BD_ADDR. Establish an active session between the TOE and DEV1 with the known address BD_ADDR.

Step 2: Attempt to pair a second remote Bluetooth device (DEV2) claiming to have a Bluetooth device address matching DEV1 BD_ADDR to the TOE. Using a Bluetooth protocol analyzer, verify that the pairing attempt by DEV2 is not completed by the TOE and that the active session to DEV1 is unaffected.

Step 3: Attempt to initialize a session to the TOE from DEV2 containing address DEV1 BD_ADDR. Using a Bluetooth protocol analyzer, verify that the session initialization attempt by DEV2 is ignored by the TOE and that the initial session to DEV1 is unaffected.

Summary

The evaluator paired the TOE with a Linux system (DEV1) and configured a separate Bluetooth dongle (DEV2) to use the same Bluetooth address as DEV1. Then, the evaluator attempted to pair DEV2 with the TOE. All packets were logged using btmon. The connection failed and the original connection between the TOE and DEV1 remained intact. The evaluator inspected the packet log to verify that the pairing failed.

2.2.4.5 Secure Simple Pairing (FIA_BLT_EXT.4)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.4-ASE-01

The evaluator shall verify that the TSS describes the secure simple pairing process.

Summary

Section 7.1.4.5 of the [ST] describes the Secure Simple Pairing process.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.4-ATE-01

The evaluator shall perform the following steps:

Step 1: Initiate pairing with the TOE from a remote Bluetooth device that supports Secure Simple Pairing.

Step 2: During the pairing process; observe the packets in a Bluetooth protocol analyzer and verify that the TOE claims support for both "Secure Simple Pairing (Host Support)" and "Secure Simple Pairing (Controller Support)" during the LMP Features Exchange.

Step 3: Verify that Secure Simple Pairing is used during the pairing process.

Summary

The evaluator used a Linux system that supports Secure Simple Pairing to pair with the TOE. All packets were logged using btmon. The evaluator inspected the packet log and found the required packets to verify Secure Simple Pairing was used.

2.2.4.6 Trusted Bluetooth Device User Authorization (FIA_BLT_EXT.6)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.6-ASE-01

The evaluator shall verify that the TSS describes all Bluetooth profiles and associated services for which explicit user authorization is required before a remote device can gain access. The evaluator shall also verify that the TSS describes any difference in behavior based on whether or not the device has a trusted relationship with the TOE for that service (i.e. whether there are any services that require explicit user authorization for untrusted devices that do not require such authorization for trusted devices). The evaluator shall also verify that the TSS describes the method by which a device can become 'trusted'.

Summary

Section 7.1.4.6 of the [ST] describes all Bluetooth profiles supported by the TOE. This section also explains that a trust relationship is automatically established when the TOE is paired with a remote device. At that point, authorization for all Bluetooth services supported by the TOE is granted to the trusted device. No further “protected” services are claimed.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.6-ATE-01

The evaluator shall perform the following tests:

- *Test 1: While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a "protected" Bluetooth service (as specified in the assignment in FIA_BLT_EXT.6.1) from a "trusted" remote device. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.*
- *Test 2: The evaluator shall repeat Test 1, this time allowing the authorization and verifying that the remote device successfully accesses the service.*

Summary

The [ST] does not claim any services for which additional user authorization is required. Please refer to the test assurance activities for FIA_BLT_EXT.7.

2.2.4.7 Untrusted Bluetooth Device User Authorization (FIA_BLT_EXT.7)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.7-ASE-01

The TSS evaluation activities for this component are addressed by FIA_BLT_EXT.6.

Summary

Please refer to the TSS assurance activities for FIA_BLT_EXT.6.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-BTPPM-FIA_BLT_EXT.7-ATE-01

The evaluator shall perform the following tests if the TSF differentiates between "trusted" and "untrusted" devices for the purpose of granting access to services. If it does not, then the test evaluation activities for FIA_BLT_EXT.6 are sufficient to satisfy this component.

- *Test 1: While the service is in active use by an application on the TOE, the evaluator shall attempt to gain access to a "protected" Bluetooth service (as specified in the assignment in FIA_BLT_EXT.7.1) from an "untrusted" remote device. The evaluator shall verify that the user is explicitly asked for authorization by the TOE to allow access to the service for the particular remote device. The evaluator shall deny the authorization on the TOE and verify that the remote attempt to access the service fails due to lack of authorization.*
- *Test 2: The evaluator shall repeat Test 1, this time allowing the authorization and verifying that the remote device successfully accesses the service.*
- *Test 3: (conditional): If there exist any services that require explicit user authorization for access by untrusted devices but not by trusted devices (i.e. a service that is listed in FIA_BLT_EXT.7.1 but not FIA_BLT_EXT.6.1), the evaluator shall repeat Test 1 for these services and observe that the results are identical. That is, the evaluator shall use these results to verify that explicit user approval is required for an untrusted device to access these services, and failure to grant this approval will result in the device being unable to access them.*
- *Test 4: (conditional): If test 3 applies, the evaluator shall repeat Test 2 using any services chosen in Test 3 and observe that the results are identical. That is, the evaluator shall use these results to verify that explicit user approval is required for an untrusted device to access these services, and granting this approval will result in the device being able to access them.*
- *Test 5: (conditional): If test 3 applies, the evaluator shall repeat Test 3 except this time designating the device as "trusted" prior to attempting to access the service. The evaluator shall verify that access to the service is granted without explicit user authorization (because the device is now trusted and therefore FIA_BLT_EXT.7.1 no longer applies to it). That is, the evaluator shall use these results to demonstrate that the TSF will grant a device access to different services depending on whether or not the device is trusted.*

Summary

Test 1: The evaluator attempted to connect a Linux system to the TOE to access the RFCOMM Handsfree Audio Gateway service, but denied the pairing on the TOE. The Linux system could not access the service.

Test 2: The evaluator successfully connected a Linux system to the TOE to access the RFCOMM Handsfree Audio Gateway service after authorizing the pairing on the TOE. The Linux system could access the service.

Test 3: As explained in the [ST], a trust relationship is established when the TOE is paired with a remote device. At that point, authorization for all Bluetooth services supported by the TOE is granted to the trusted device. Therefore, this test is identical to Test 1.

Test 4: As explained in the [ST], a trust relationship is established when the TOE is paired with a remote device. At that point, authorization for all Bluetooth services supported by the TOE is granted to the trusted device. Therefore, this test is identical to Test 2.

Test 5: The evaluator paired a Linux system to the TOE, then disconnected both devices while keeping the pairing information intact (the Linux system is trusted). Then, the

evaluator successfully connected a Linux system to the TOE to access the RFCOMM Handsfree Audio Gateway service, without any explicit authorization required.

2.2.4.8 Port Access Entity Authentication (FIA_PAE_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FIA_PAE_EXT.1-ATE-01

The evaluator shall perform the following tests:

- *Test 1: The evaluator shall demonstrate that the TOE has no access to the test network. After successfully authenticating with an authentication server through a wireless access system, the evaluator shall demonstrate that the TOE does have access to the test network.*
- *Test 2: The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid client certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.*
- *Test 3: The evaluator shall demonstrate that the TOE has no access to the test network. The evaluator shall attempt to authenticate using an invalid authentication server certificate, such that the EAP-TLS negotiation fails. This should result in the TOE still being unable to access the test network.*

Summary

The evaluator started an access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

The CA certificate used to issue the certificate of the access point EAP-TLS server was trusted by the TOE.

Finally, the evaluator verified the TOE does not have access to the test network, then instructed the TOE to initiate a connection to the access point. Depending on the specific test, the connection succeeded or failed:

Test 1: The evaluator generated a valid EAP-TLS client certificate and a valid EAP-TLS server certificate, and verified the connection was successfully established.

Test 2: The evaluator generated an expired EAP-TLS client certificate and verified the connection was rejected as expected.

Test 3: The evaluator generated an expired EAP-TLS server certificate and verified the connection was rejected as expected.

2.2.4.9 Multiple Authentication Mechanisms (FIA_UAU.5)

TSS Assurance Activities

Assurance Activity AA-OSPP-FIA_UAU.5-ASE-01

The evaluator will ensure that the TSS describes the rules as to how each authentication mechanism specified in FIA_UAU.5.1 is implemented and used. Example rules are how the authentication mechanism authenticates the user (i.e. how does the TSF verify that the correct password or authentication factor is used), 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). Rules regarding how the authentication factors interact in terms of unsuccessful authentication are covered in FIA_AFL.1.

Summary

Section 7.1.4.8 of the [ST] describes the different authentication mechanisms supported by the TOE: password or smart card. For each of these authentication mechanisms, the aforementioned section describes the rules as to how it is used.

Guidance Assurance Activities

Assurance Activity AA-OSPP-FIA_UAU.5-AGD-01

The evaluator will verify that configuration guidance for each authentication mechanism is addressed in the AGD guidance.

Summary

Section 3.3 and Section 3.4 of the [CCGUIDE] provide guidance for the smart card and password authentication mechanisms, respectively.

Test Assurance Activities

Assurance Activity AA-OSPP-FIA_UAU.5-ATE-01

- *Test 56: 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 57: 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*

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 58: 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 59: 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.*

Authentication mechanisms related to authentication based on X.509 certificates are tested under FIA_X509_EXT.1 and SSH public key-based authentication are tested in the Functional Package for Secure Shell (SSH), version 1.0.

For each authentication mechanism rule, the evaluator will ensure that the authentication mechanism(s) behave as documented in the TSS.

Summary

Test 56: The evaluator successfully authenticated with a known username and password.

Test 57: The evaluator verified that attempting to authenticate with a known username, but incorrect password is unsuccessful.

Test 58: Using a YubiKey, the evaluator successfully authenticated with a known username and PIN.

Test 59: Using a YubiKey, the evaluator verified that attempting to authenticate with a known username, but incorrect PIN is unsuccessful.

2.2.4.10 X.509 Certificate Validation (FIA_X509_EXT.1)

TSS Assurance Activities

Assurance Activity AA-OSPP-FIA_X509_EXT.1-ASE-01

[TD0773] 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.

If there are exceptional use cases where the OS cannot perform revocation checking in accordance with at least one of the revocation methods, the evaluator will ensure the TSS describes each revocation checking exception use case and, for each exception, the alternate functionality the TOE implements to determine the status of the certificate and disable functionality dependent on the validity of the certificate.

Summary

Section 7.1.4.9 of the [ST] states that a certificate is checked for validity when imported into the keychain, during session establishment with a TLS server, and prior to presenting a certificate to the server during TLS mutual authentications. It also states that certificate path validation is performed according to sections 6.1.4.10 and 6.1.4.11 of the [ST], which provide a full description of the algorithm. There are no exceptional use cases where the OS cannot perform revocation checking.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FIA_X509_EXT.1-ATE-01

[TD0773] The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The 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 64: The evaluator will demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn: by establishing a certificate path in which one of the issuing certificates is not a CA certificate, by omitting the basicConstraints field in one of the issuing certificates, by setting the basicConstraints field in an issuing certificate to have CA=False, by omitting the CA signing bit of the key usage field in an issuing certificate, and by setting the path length field of a valid CA field to a value strictly less than the certificate path. The evaluator will then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator will then remove trust in one of the CA certificates, and show that the function fails.*
- *Test 65: The evaluator will demonstrate that validating an expired certificate results in the function failing.*
- *Test 66: [Conditional, to be performed for use cases identified in exceptions that cannot be configured to allow revocation checking] The evaluator will test that the OS can properly handle revoked certificates - conditional on whether CRL, OCSP, OCSP stapling, or OCSP multi-stapling is selected; if multiple methods are selected, then a test will 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). If OCSP stapling per RFC 6066 is the only supported revocation method, testing revocation of the intermediate CA certificate is omitted. 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. If the exceptions are configurable, the evaluator shall attempt to configure the exceptions to*

allow revocation checking for each function indicated in FIA_X509_EXT.2.

- *Test 67: If any 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 68: The evaluator will modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)*
- *Test 69: The evaluator will modify any byte in the last eight bytes of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)*
- *Test 70: The evaluator will modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature of the certificate will not validate.)*
- *Test 71[conditional, to be performed if*
  - *ECDSA schemes is selected from FCS_COP.1.1/SIGN*
  - *6187 is selected from FCS_SSH_EXT.1.1 from Functional Package for Secure Shell (SSH), version 1.0*

  ]:
  - *Test 71.1: The evaluator will establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator will confirm that the TOE validates the certificate chain.*
  - *Test 71.2: The evaluator will replace the intermediate certificate in the certificate chain for Test 71.1 with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 71.1, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator will confirm the TOE treats the certificate as invalid.*
- *Test 72[conditional, to be performed if*
  - *exceptional use cases is selected from FIA_X509_EXT.1.1*

  ]: For each exceptional use case for revocation checking described in the ST, the evaluator shall attempt to establish the conditions of the use case, designate the certificate as invalid and perform the function relying on the certificate. The evaluator shall observe that the alternate revocation checking mechanism successfully prevents performance of the function.

[Conditional, to be performed if "authentication based on X.509 certificates" is selected in FIA_UAU.5]: The evaluator will generate an X.509v3 certificate for a user with the Client Authentication Extended 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 two tests:

- *Test 73: 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 74: 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.*

The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The 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 75: 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 76: 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 77: 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

The evaluator started a remote TLS server using the OpenSSL library. All packets were logged using Wireshark. A test application was used to instruct the TOE to perform TLS connections.

The CA certificate, used to issue a certificate chain of four certificates (one root CA, two intermediate CAs, and one server node certificate), was trusted by the TOE.

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

Test 64a: The evaluator created an (otherwise valid) intermediate CA certificate with a basicConstraints extension containing the “CA=false” entry and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 64b: The evaluator created an (otherwise valid) intermediate CA certificate without a basicConstraints extension and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 64c: The evaluator created an (otherwise valid) intermediate CA certificate with a basicConstraints extension containing the “CA=false” entry and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 64d: The evaluator created an (otherwise valid) intermediate CA certificate without the certificate signing purpose set in the keyUsage extension and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 64e: The evaluator created an (otherwise valid) intermediate CA certificate with a path length set to 0 and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 64f: The evaluator created a valid certificate chain and verified the connection was successfully established, indicating that the certificate chain validation succeeded. The evaluator then removed an intermediate certificate from the chain and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 65: The evaluator created an (otherwise valid) expired intermediate CA certificate and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 66: The evaluator created a valid certificate chain. Then, the evaluator started an OCSP server and configured the server node certificate to be marked as “revoked”. The evaluator verified the connection was rejected as expected. This test was repeated using an intermediate certificate. To test the validation function, the evaluator started an OCSP server and configured the server node certificate to be marked as “valid”. The evaluator verified the connection was accepted as expected. Finally, the tester configured the server

node certificate to be marked as “revoked” and verified the connection was rejected as expected.

Test 67: The evaluator created a valid certificate chain. Then, the evaluator started an OCSP server, configured the server node certificate to be marked as “valid”, but did not add the OCSP signing purpose to the certificate used to sign the OCSP response. The evaluator verified the connection was rejected as expected.

Test 68: The evaluator patched the OpenSSL TLS server to flip a bit in the first byte of the server node certificate, immediately before sending it to the client, and verified the connection was rejected as expected, indicating that the certificate failed to parse correctly.

Test 69: The evaluator patched the OpenSSL TLS server to flip a bit in the last byte of the server node certificate signature, immediately before sending it to the client, and verified the connection was rejected as expected, indicating that the certificate failed to validate.

Test 70: The evaluator patched the OpenSSL TLS server to flip a bit in the first byte of the server node certificate public key, immediately before sending it to the client, and verified the connection was rejected as expected, indicating that the certificate failed to validate.

Test 71.1: The evaluator created a valid certificate chain, using named elliptic curve parameters for each certificate, and verified the connection was successfully established, indicating that the certificate chain validation succeeded. These steps were repeated for each of the supported curves.

Test 71.2: The evaluator created a valid certificate chain, using explicit elliptic curve parameters for an intermediate CA certificate, and verified the connection was rejected as expected, indicating that the certificate chain validation failed. These steps were repeated for each of the supported curves.

Test 72: Not applicable as [ST] does not select “exceptional use cases”.

Test 73: Not applicable as the [ST] does not select “authentication based on X.509 certificates” in FIA_UAU.5.

Test 74: Not applicable as the [ST] does not select “authentication based on X.509 certificates” in FIA_UAU.5.

Test 75: The evaluator created an (otherwise valid) intermediate CA certificate without a basicConstraints extension and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 76: The evaluator created an (otherwise valid) intermediate CA certificate with a basicConstraints extension containing the “CA=false” entry and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 77: The evaluator created a valid intermediate CA certificate with a basicConstraints extension containing the “CA=true” entry and verified the connection was successfully established, indicating that the certificate chain validation succeeded.

2.2.4.11 X.509 Certificate Validation (FIA_X509_EXT.1/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FIA_X509_EXT.1-WLAN-ASE-01

The evaluator shall ensure the TSS describes where the check of validity of the EAP-TLS certificates takes place. The evaluator shall also ensure the TSS also provides a description of the certificate path validation algorithm.

Summary

Section 7.1.4.9 of the [ST] states that a certificate is checked for validity when imported into the keychain, during session establishment with an EAP-TLS server, and prior to presenting a certificate to the server during EAP-TLS mutual authentications. It also states that certificate path validation is performed according to Sections 6.1.4.10 and 6.1.4.11 of the [ST], which provide a full description of the algorithm.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FIA_X509_EXT.1-WLAN-ATE-01

The tests described must be performed in conjunction with the other Certificate Services assurance activities. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. The evaluator shall create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- *Test 1: The evaluator shall then load a certificate or certificates to the Trust Anchor Database needed to validate the certificate to be used in the function (e.g. application validation), and demonstrate that the function succeeds. The evaluator then shall delete one of the certificates, and show that the function fails.*
- *Test 2: The evaluator shall demonstrate that validating an expired certificate results in the function failing*
- *Test 3: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate does not contain the basicConstraints extension. The validation of the certificate path fails.*
- *Test 4: The evaluator shall construct a certificate path, such that the certificate of the CA issuing the TOE's certificate has the cA flag in the basicConstraints extension not set. The validation of the certificate path fails*
- *Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate (the certificate will fail to parse correctly).*
- *Test 6: The evaluator shall modify any bit in the last byte of the signature algorithm of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).*
- *Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate (the signature on the certificate will not validate).*

Summary

The evaluator started an access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

The CA certificate, used to issue a certificate chain of four certificates (one root CA, two intermediate CAs, and one server node certificate), was trusted by the TOE.

Finally, the evaluator instructed the TOE to initiate a connection to the access point. Depending on the specific test, the connection succeeded or failed:

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

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

Test 3: The evaluator created an (otherwise valid) intermediate CA certificate without a basicConstraints extension and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

Test 4: The evaluator created an (otherwise valid) intermediate CA certificate with a basicConstraints extension containing the “CA=false” entry and verified the connection was rejected as expected, indicating that the certificate chain validation failed.

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

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

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

2.2.4.12 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-OSPP-FIA_X509_EXT.2-ATE-01

[TD0789] The evaluator will acquire or develop an application that uses the selected OS 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 all selections listed.

Summary

The evaluator started a remote webserver using the OpenSSL library. All packets were logged using Wireshark. A test application was used to instruct the TOE to perform HTTPS connections.

The CA certificate, used to issue a certificate chain of four certificates (one root CA, two intermediate CAs, and one server node certificate), was trusted by the TOE.

Then, the evaluator instructed the TOE to initiate an HTTPS request to the webserver. A TLS connection is established as part of the HTTPS request. The evaluator observed that the connection succeeded, the traffic is identified as TLS, and the generated server certificate is used to authenticate the connection.

2.2.4.13 X.509 Certificate Authentication (EAP-TLS for WLAN) (FIA_X509_EXT.2/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FIA_X509_EXT.2-WLAN-ASE-01

[TD0703] The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operational environment so that the TOE can use the certificates.

Summary

Section 7.1.4.10 of the [ST] describes the use of X.509v3 certificates for performing mutual authentication for TLS connections. The evaluator confirmed that the TSS describes, in sufficient detail, the mechanism by which the TOE determines the appropriate certificate to use, as well as how the configuration of the certificate trust policy dictates whether or not the certificate is trusted and its intended purpose.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FIA_X509_EXT.2-WLAN-AGD-01

[TD0703] If not already present in the TSS, the evaluator shall check the administrative guidance to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions for configuring the operating environment so that the TOE can use the certificates.

Summary

Section 7.1.4.10 of the [ST] describes how the TOE chooses which certificates to use. Section 5.1 of the [CCGUIDE] provides instructions for loading, removing, and changing trust settings of certificates.

Test Assurance Activities

No assurance activities defined.

2.2.4.14 X.509 Certificate Storage and Management (FIA_X509_EXT.6)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FIA_X509_EXT.6-ASE-01

The evaluator shall examine the TSS to determine that it describes all certificate stores implemented that contain certificates used to meet the requirements of this PP-Module. This description shall contain information pertaining to how certificates are loaded into the store, and how the store is protected from unauthorized access.

If the TOE relies on a platform mechanism for certificate loading and storage, the evaluator shall verify that the TSS identifies this mechanism and describes how use of this mechanism is protected against unauthorized access.

Summary

Section 7.1.4.11 of the [ST] describes all certificate stores implemented by the TOE. The TOE stores certificates in the keychain. This section also describes how the administrators can load certificates in the keychain and how the TOE protects the keychain against unauthorized access.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FIA_X509_EXT.6-AGD-01

The evaluator shall check the administrative guidance to ensure that it describes how to load X.509 certificates into the TOE's certificate store, regardless of whether the TSF provides this mechanism itself or the TOE relies on a platform-provided mechanism for this.

Summary

Section 5.1 of the [CCGUIDE] contains instructions for users to manage the CA certificate store using the Keychain Access app, while applications can use the API functions described in Section 5.2.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FIA_X509_EXT.6-ATE-01

The evaluator shall perform the following test for each TOE function that requires the use of certificates:

- *Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. The evaluator shall then load any certificates needed to validate the certificate to be used in the function and demonstrate that the function succeeds. The evaluator shall then delete one of these dependent certificates and show that the function fails.*
- *Test 2: The evaluator shall demonstrate that the mechanism used to load or configure X.509 certificates cannot be accessed without appropriate authorization.*

Summary

Test 1: The evaluator started an access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections. The evaluator then followed the steps in the test procedure, loading and removing the CA certificate as needed, and verified the TOE did not automatically trust the server certificate when the CA certificate was removed.

Test 2: The evaluator attempted to import a CA certificate using the Keychain Access app as an unprivileged user and verified the TOE did not import the CA Certificate.

2.2.5 Security management (FMT)

2.2.5.1 Management of Security Functions Behavior (FMT_MOF_EXT.1)

TSS Assurance Activities

Assurance Activity AA-OSPP-FMT_MOF_EXT.1-ASE-01

The evaluator will 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

Section 7.1.5 of the [ST] specifies that functions requiring Administrator access require the user to enter the correct Administrator password before allowing the user to modify the function.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FMT_MOF_EXT.1-ATE-01

[TD1005] The evaluator will also perform the following test.

- *Test 32: For each function that is indicated as restricted to only the administrator, the evaluation will 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 will 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

The evaluator performed each restricted function as an administrator in FMT_SMF_EXT.1 and FMT_SMF.1/WLAN. Then, the evaluator attempted to perform each function as a non-administrator as described below.

Function 1 & 2: The evaluator verified that screen lock settings can be set by the administrator by installing a configuration profile, and an unprivileged user is not able to modify the settings.

Function 4: The evaluator attempted to change the audit storage capacity by modifying the /etc/security/audit_control file and verified that the operation was denied.

Function 5: The evaluator attempted to use pwpolicy to configure the minimum password length and verified that the operation was denied.

Function 6: The evaluator attempted to use pwpolicy to configure the minimum number of special characters and verified that the operation was denied.

Function 7: The evaluator attempted to use pwpolicy to configure the minimum number of numeric characters and verified that the operation was denied.

Function 8: The evaluator attempted to use pwpolicy to configure the minimum number of uppercase characters and verified that the operation was denied.

Function 9: The evaluator attempted to use pwpolicy to configure the minimum number of lowercase characters and verified that the operation was denied.

Function 10: The evaluator attempted to use pwpolicy to configure the lockout policy and verified that the operation was denied.

Function 11: The evaluator attempted to change the firewall settings and verified that administrator credentials are required.

Function 15: The evaluator attempted to change the audit rules by modifying the /etc/security/audit_control file and verified that the operation was denied.

Function 16: The evaluator attempted to change the network time server and verified that administrator credentials are required.

Function 17: The evaluator attempted to change the automatic software update settings and verified that administrator credentials are required.

Function WL-1: The security policy for wireless networks is configured by installing configuration profiles. The evaluator verified that unprivileged users cannot install configuration profiles.

Function WL-2: The evaluator configured the TOE such that administrator authorization is required to change wireless networks, then added a known wireless network set to “Auto-Join”. After this configuration, the evaluator verified that unprivileged users can only connect to the allowed wireless network.

Function WL-3: The evaluator attempted to enable Internet Sharing (i.e., a hotspot WLAN network) and verified that administrator credentials are required.

Function WL-8: The evaluator attempted to import CA certificates using the Keychain Access app and verified that administrator credentials are required.

Function WL-9: The evaluator attempted to delete CA certificates using the Keychain Access app and verified that administrator credentials are required.

2.2.5.2 Management of Security Functions Behavior (FMT_MOF_EXT.1/BT)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FMT_MOF_EXT.1-BT-ASE-01

The evaluator shall examine the TSS to ensure that it identifies the Bluetooth-related management functions that are supported by the TOE and the roles that are authorized to perform each function.

Summary

The [ST] does not restrict any Bluetooth management functions to the administrator.

Guidance Assurance Activities

Assurance Activity AA-BTPPM-FMT_MOF_EXT.1-BT-AGD-01

The evaluator shall examine the operational guidance to ensure that it provides sufficient guidance on each supported Bluetooth manaagement function to describe how the function is performed and any role restrictions on the subjects that are authorized to perform the function.

Summary

Guidance to perform each Bluetooth management function is provided throughout [CCGUIDE], notably in Section 3.12. As indicated in the [ST], the Bluetooth management functions are not restricted to the administrator.

Test Assurance Activities

Assurance Activity AA-BTPPM-FMT_MOF_EXT.1-BT-ATE-01

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 will 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

The [ST] does not restrict any Bluetooth management functions to the administrator. Therefore, these tests are not applicable.

2.2.5.3 Specification of Management Functions (FMT_SMF_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-OSPP-FMT_SMF_EXT.1-AGD-01

The evaluator will verify that every management function captured in the ST is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

Summary

Function 1 & 2: Section 3.5 of the [CCGUIDE] describes how to enable/disable screen lock and configure screen lock inactivity timeout.

Function 3: Section 5 of the [CCGUIDE] describes how to import keys, certificates, and WLAN passwords into the keychain.

Function 4: Section 3.8.1 of the [CCGUIDE] describes how to configure the audit storage capacity.

Function 5: Section 3.4 of the [CCGUIDE] describes how to set the minimum password length using pwpolicy.

Function 6-9: Section 3.4 of the [CCGUIDE] describes how to set the minimum number of special, numeric, uppercase, or lowercase characters in a password using pwpolicy.

Function 10: Section 3.4 of the [CCGUIDE] describes how to set the maximum number of failed authentication attempts using pwpolicy.

Function 11: Section 3.7 of the [CCGUIDE] describes how to configure the firewall.

Function 15: Section 3.8.1 of the [CCGUIDE] describes how to configure the audit rules.

Function 16: Section 3.9 of the [CCGUIDE] describes how to configure the name or address of the network time server.

Function 17: Section 3.10 of the [CCGUIDE] describes how to enable/disable automatic downloading and installing of software updates.

Function 18: Section 3.11 of the [CCGUIDE] describes how to configure the Wi-Fi interface.

Function 19: Section 3.12.1 of the [CCGUIDE] describes how to enable/disable Bluetooth.

For each management function, the evaluator verified that the corresponding section contains sufficient information to perform the corresponding management duties.

Test Assurance Activities

Assurance Activity AA-OSPP-FMT_SMF_EXT.1-ATE-01

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

Summary

Function 1: The evaluator verified that the administrator can enable and disable screen lock by installing a configuration profile.

Function 2: The evaluator verified that the administrator can change the screen lock activity timeout by installing a configuration profile.

Function 3: The evaluator verified that any user can import secrets into secure key storage by importing a certificate into the Keychain Access app.

Function 4: The evaluator verified that an administrator can configure the audit storage capacity by modifying the /etc/security/audit_control file.

Function 5: The evaluator verified that an administrator can use pwpolicy to configure the minimum password length of a user. The evaluator then verified the user’s password could only be changed to a password of at least the required length.

Function 6-9: The evaluator verified that an administrator can use pwpolicy to configure the minimum number of special, numeric, uppercase, and lowercase characters in a password of a user. The evaluator then verified the user’s password could only be changed to a password

containing at least the required number of special, numeric, uppercase, and lowercase characters.

Function 10: The evaluator verified that an administrator can use pwpolicy to configure the maximum number of failed authentication attempts allowed by the lockout policy.

Function 11: The evaluator verified that an administrator can enable/disable the firewall and modify the settings.

Function 15: The evaluator verified that an administrator can configure the audit rules by modifying the /etc/security/audit_control file.

Function 16: The evaluator verified that an administrator can configure the network time server by changing the time server address and toggling whether the time and date is set automatically by the server.

Function 17: The evaluator verified that an administrator can enable/disable automatic software updates.

Function 18: The evaluator verified that any user can enable/disable Wi-Fi and join a wireless network.

Function 19: The evaluator verified that any user can enable/disable Bluetooth in Test 1 of FMT_SMF_EXT.1/BT.

2.2.5.4 Specification of Management Functions (FMT_SMF_EXT.1/BT)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FMT_SMF_EXT.1-BT-ASE-01

> *The evaluator shall ensure that the TSS includes a description of the Bluetooth profiles and services supported and the Bluetooth security modes and levels supported by the TOE.*
>
> *If function BT-4, "Allow/disallow additional wireless technologies to be used with Bluetooth," is selected, the evaluator shall verify that the TSS describes any additional wireless technologies that may be used with Bluetooth, which may include Wi-Fi with Bluetooth High Speed and/or NFC as an Out of Band pairing mechanism.*
>
> *If function BT-5, "Configure allowable methods of Out of Band pairing (for BR/EDR and LE)," is selected, the evaluator shall verify that the TSS describes when Out of Band pairing methods are allowed and which ones are configurable.*
>
> *If function BT-8, "Disable/enable the Bluetooth services and/or profiles available on the OS (for BR/EDR and LE)," is selected, the evaluator shall verify that all supported Bluetooth services are listed in the TSS as manageable and, if the TOE allows disabling by application rather than by service name, that a list of services for each application is also listed.*
>
> *If function BT-9, "Specify minimum level of security for each pairing (for BR/EDR and LE)," is selected, the evaluator shall verify that the TSS describes the method by which the level of security for pairings are managed, including whether the setting is performed for each pairing or is a global setting.*

Summary

Section 7.1.4.6 of the [ST] describes the Bluetooth profiles and services supported by the TOE. Section 7.1.5 of the [ST] describes the Bluetooth security modes and levels supported by the TOE.

The [ST] does not select function BT-4, BT-5, BT-8, or BT-9.

Guidance Assurance Activities

Assurance Activity AA-BTPPM-FMT_SMF_EXT.1-BT-AGD-01

The evaluator shall ensure that the management functions defined in the PP-Module are described in the guidance to the same extent required for the Base-PP management functions.

Summary

Guidance to perform each Bluetooth management function is provided throughout [CCGUIDE], notably in Section 3.12. The evaluator verified the functions are described to the same extent as required for the other management functions.

Test Assurance Activities

Assurance Activity AA-BTPPM-FMT_SMF_EXT.1-BT-ATE-01

The evaluator shall use a Bluetooth-specific protocol analyzer to perform the following tests:

- *Test 1: The evaluator shall disable the Discoverable mode and shall verify that other Bluetooth BR/EDR devices cannot detect the TOE. The evaluator shall use the protocol analyzer to verify that the TOE does not respond to inquiries from other devices searching for Bluetooth devices. The evaluator shall enable Discoverable mode and verify that other devices can detect the TOE and that the TOE sends response packets to inquiries from searching devices.*

The following tests are conditional on if the corresponding function is included in the ST:

- *Test 2: (conditional): The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name, change the Bluetooth device name, and verify that the Bluetooth traffic from the TOE lists the new name. The evaluator shall examine Bluetooth traffic from the TOE to determine the current Bluetooth device name for BR/EDR and LE. The evaluator shall change the Bluetooth device name for LE independently of the device name for BR/EDR. The evaluator shall verify that the Bluetooth traffic from the TOE lists the new name.*
- *Test 3: (conditional): The evaluator shall disable Bluetooth BR/EDR and enable Bluetooth LE. The evaluator shall examine Bluetooth traffic from the TOE to confirm that only Bluetooth LE traffic is present. The evaluator shall repeat the test with Bluetooth BR/EDR enabled and Bluetooth LE disabled, confirming that only Bluetooth BR/EDR is present.*
- *Test 4: (conditional): For each additional wireless technology that can be used with Bluetooth as claimed in the ST, the evaluator shall revoke Bluetooth permissions from that technology. If the set of supported wireless technologies includes Wi-Fi, the evaluator shall verify that Bluetooth High Speed is not able to send Bluetooth traffic over Wi-Fi when disabled. If the set of supported wireless technologies includes NFC, the evaluator shall verify that NFC cannot be used for pairing when disabled. For any other supported wireless technology, the evaluator shall verify that it cannot be used with Bluetooth in the specified manner when disabled. The evaluator shall then re-enable all supported wireless technologies and verify that all functionality that was previously unavailable has been restored.*
- *Test 5: (conditional): The evaluator shall attempt to pair using each of the Out of Band pairing methods, verify that the pairing method works, iteratively disable each pairing method, and verify that the pairing method fails.*
- *Test 6: (conditional): The evaluator shall enable Advertising for Bluetooth LE, verify that the advertisements are captured by the protocol analyzer, disable Advertising, and verify that no advertisements from the device are captured by the protocol analyzer.*
- *Test 7: (conditional): The evaluator shall enable Connectable mode and verify that other Bluetooth devices may pair with the TOE and (if the devices were bonded) re-connect after pairing and disconnection. For BR/EDR devices: The evaluator shall use the protocol analyzer to verify that the TOE responds to pages from the other devices and permits pairing and re-connection. The evaluator shall disable Connectable mode and verify that the TOE does not respond to pages from remote Bluetooth devices, thereby not permitting pairing or re-connection. For LE: The evaluator shall use the protocol analyzer to verify that the TOE sends connectable advertising events and responds to connection requests. The evaluator*

shall disable Connectable mode and verify that the TOE stops sending connectable advertising events and stops responding to connection requests from remote Bluetooth devices.

- *Test 8: (conditional): For each supported Bluetooth service and/or profile listed in the TSS, the evaluator shall verify that the service or profile is manageable. If this is configurable by application rather than by service and/or profile name, the evaluator shall verify that a list of services and/or profiles for each application is also listed.*
- *Test 9: (conditional): The evaluator shall allow low security modes/levels on the TOE and shall initiate pairing with the TOE from a remote device that allows only something other than Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR), or Security Mode 1/Level 3 (for LE). (For example, a remote BR/EDR device may claim Input/Output capability "NoInputNoOutput" and state that man-in-the-middle (MiTM) protection is not required. A remote LE device may not support encryption.) The evaluator shall verify that this pairing attempt succeeds due to the TOE falling back to the low security mode/level. The evaluator shall then remove the pairing of the two devices, prohibit the use of low security modes/levels on the TOE, then attempt the connection again. The evaluator shall verify that the pairing attempt fails. With the low security modes/levels disabled, the evaluator shall initiate pairing from the TOE to a remote device that supports Security Mode 4/Level 3 or Security Mode 4/Level 4 (for BR/EDR) or Security Mode 1/Level 3 (for LE). The evaluator shall verify that this pairing is successful and uses the high security mode/level.*

Summary

Test 1: The evaluator disabled Bluetooth on the TOE. All packets were logged using btmon. Then, the evaluator attempted to pair a Linux system with the TOE. From the packet log, the evaluator confirmed that the TOE did not respond to inquiries and the pairing did not complete.

Test 2-9: Not applicable as the [ST] does not select this function.

2.2.5.5 Specification of Management Functions (WLAN Client) (FMT_SMF.1/WLAN)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FMT_SMF.1-WLAN-AGD-01

The evaluator shall check the operational guidance to verify that every management function claimed by the TOE is described there. The evaluator shall also verify that these descriptions include the information required to perform the management duties associated with the function.

Summary

Guidance to perform each WLAN management function is provided throughout [CCGUIDE], notably in Section 3.11 and Section 5.1. The guidance was consulted and has been assessed by the evaluator while testing each management function, and the evaluator confirmed the descriptions included the information required to perform the management duties for the function.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FMT_SMF.1-WLAN-ATE-01

The evaluator shall test the TOE's ability to provide the management functions by configuring the TOE and performing the management activities associated with each function claimed in the SFR.

Note that this may be accomplished in conjunction with the testing of other requirements, such as

FCS_TLSC_EXT.1/WLAN and FTA_WSE_EXT.1.

Summary

Function WL-1: The security policy for wireless networks is configured by installing configuration profiles. The evaluator verified that the administrator can install configuration profiles containing wireless network settings and the TOE connects to the configured network.

Function WL-2: The evaluator configured the TOE such that administrator authorization is required to change wireless networks, then added a known wireless network set to “Auto-Join”. After this configuration, the evaluator verified that unprivileged users can only connect to the allowed wireless network.

Function WL-3: The evaluator verified that the administrator can enable and disable Internet Sharing (i.e., a hotspot WLAN network) in the Settings.

Function WL-5: The evaluator verified that any user can enable and disable AirDrop in the Settings.

Function WL-8: The evaluator verified that the administrator can import CA certificates using the Keychain Access app.

Function WL-9: The evaluator verified that the administrator can delete CA certificates using the Keychain Access app.

2.2.6 Protection of the TSF (FPT)

2.2.6.1 Access Controls (FPT_ACF_EXT.1)

TSS Assurance Activities

Assurance Activity AA-OSPP-FPT_ACF_EXT.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 7.1.6.1 of the [ST] specifies that System Integrity Protection protects the kernel drivers/modules, shared libraries, and system executables, while standard file permissions are used to protect security audit logs and system configuration files. The location of each is specified in this section as well.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FPT_ACF_EXT.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 33: The evaluator will attempt to modify all kernel drivers and modules.*
- *Test 34: The evaluator will attempt to modify all security audit logs generated by the logging subsystem.*
- *Test 35: The evaluator will attempt to modify all shared libraries that are used throughout*

the system.

- *Test 36: The evaluator will attempt to modify all system executables.*
- *Test 37: The evaluator will attempt to modify all system configuration files.*
- *Test 38: The evaluator will attempt to modify any additional components selected.*

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 39: The evaluator will attempt to read security audit logs generated by the auditing subsystem*
- *Test 40: The evaluator will attempt to read system-wide credential repositories*
- *Test 41: The evaluator will attempt to read any other object specified in the assignment*

Summary

The evaluator created an unprivileged user account and performed the following tests as that user:

Test 33: The evaluator attempted to modify all kernel extensions in /System/Library. The evaluator verified that the modifications were denied.

Test 34: The evaluator attempted to modify all security audit logs in /var/audit and /var/db/diagnostics. The evaluator verified that the modifications were denied.

Test 35: The evaluator attempted to modify all shared libraries in /usr/lib, /Library, and /System/Library. The evaluator verified that the modifications were denied.

Test 36: The evaluator attempted to modify all system executables in /usr/bin. The evaluator verified that the modifications were denied.

Test 37: The evaluator attempted to modify all system configuration files in /private/etc and /Library/Preferences. The evaluator verified that the modifications were denied.

Test 38: The evaluator attempted to modify all TSF data in /var/folders/zz and all applications in /Applications, /System/Applications, and /System/Library/CoreServices. The evaluator verified that the modifications were denied.

Test 39: The evaluator attempted to read all security logs in /var/audit and /var/db/diagnostics. The evaluator verified that the read attempts were denied.

Test 40: The evaluator attempted to read all system-wide credential repositories in /Library/Keychains and /var/db/SystemKey. The evaluator verified that the read attempts were denied.

Test 41: Not applicable as the [ST] selects “no other objects”.

2.2.6.2 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-OSPP-FPT_ASLR_EXT.1-ATE-01

The evaluator will select 3 executables included with the TSF. If the TSF includes a web browser it

must be selected. If the TSF includes a mail client it must be selected. 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. This test can also be completed on the same hardware and rebooting between application launches.

Summary

The evaluator launched the Safari web browser, Mail mail client, and Calculator app and used vmmap to dump the memory mappings to a file. This was repeated after rebooting the machine. Finally, the evaluator examined the mappings and found that they were different.

2.2.6.3 Stack Buffer Overflow Protection (FPT_SBOP_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FPT_SBOP_EXT.1-ATE-01

[TD0906] For stack-based OSes, the evaluator will determine that the TSS contains a description of stack-based buffer overflow protections used by the OS. These are referred to by a variety of terms. These include, but are not limited to, ASLR, tagging, stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner. The evaluator will also preform the following test:

- *Test 42 (Conditional: stack-based overflow protection can be determined by inventorying): 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.*

For OSes that store parameters/variables separately from control flow values, the evaluator will verify that the TSS describes what data structures control values, parameters, and variables are stored. The evaluator will also ensure that the TSS includes a description of the safeguards that ensure parameters and variables do not intermix with control flow values.

Summary

The evaluator scanned all binaries on the TOE, except those listed as exceptions in the TSS. For each of those, the evaluator used nm to confirm the presence of the functions "___stack_chk_fail" and "___stack_chk_guard". This shows that stack canaries were used.

2.2.6.4 Boot Integrity (FPT_TST_EXT.1)

TSS Assurance Activities

Assurance Activity AA-OSPP-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. 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 7.1.6.4 of the [ST] describes the boot procedures, including the entire bootchain, of both sepOS and the Apple silicon devices, including self-tests that are performed at start-up. The evaluator verified the descriptions are sufficiently detailed.

The Boot ROM code contains the Apple Root CA public key, which is used to verify the digital signatures of the bootchain. As the Boot ROM is immutable, the public key is sufficiently protected against modification.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FPT_TST_EXT.1-ATE-01

The evaluator will also perform the following tests:

- *Test 43: 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 44: 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 45[conditional, to be performed if*
  - *a digital signature using an X509 certificate with hardware-based protection is selected from FPT_TST_EXT.1.1*

  ]: If the ST author indicates that the integrity verification is performed using public key in an X509 certificate, the evaluator will verify that the boot integrity mechanism includes a certificate validation according to FIA_X509_EXT.1 for all certificates in the chain from the certificate used for boot integrity to a certificate in the trust store that are not themselves in the trust store. This means that, for each X509 certificate in this chain that is not a trust store element, the evaluator must ensure that revocation information is available to the TOE during the bootstrap mechanism (before the TOE becomes fully operational).

Summary

Test 43: The integrity check is performed during each boot. The evaluator rebooted the TOE and verified that the reboot was successful.

Test 44: The evaluator modified iBoot.img4 by flipping a single bit of a digest stored in the file. The evaluator rebooted the TOE and verified that the TOE did not boot and entered recovery mode.

Test 45: Not applicable as the [ST] does not select “a digital signature using an X509 certificate with hardware-based protection”.

2.2.6.5 TSF Cryptographic Functionality Testing (WLAN Client) (FPT_TST_EXT.3/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FPT_TST_EXT.3-WLAN-ASE-01

The evaluator shall examine the TSS to ensure that it details the self tests that are run by the TSF on start-up; this description should include an outline of what the tests are actually doing (e.g., rather than saying "memory is tested", a description similar to "memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written" shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

The evaluator shall examine the TSS to ensure that it describes how to verify the integrity of stored TSF executable code when it is loaded for execution. The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the integrity of stored TSF executable code has not been compromised. The evaluator also ensures that the TSS (or the operational guidance) describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases.

Summary

Section 7.1.6.4 of the [ST] describes the self-tests that are performed at start-up. The evaluator verified the descriptions are sufficiently detailed. If the self-tests fail, then the TSF will not operate, requiring the user to reboot the device. The TSS indicates that the self-tests are executed automatically on power-up.

The static reference hash is stored in the secure bootchain, cannot be modified by unauthorized means, and can be relied upon. This mechanism ensures that any unauthorized modification to the stored code will be detected prior to execution, thereby demonstrating that the integrity of the TSF executable code has not been compromised.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FPT_TST_EXT.3-WLAN-AGD-01

The evaluator shall ensure that the operational guidance describes the actions that take place for successful (e.g. hash verified) and unsuccessful (e.g., hash not verified) cases.

Summary

The actions are present in the TSS as listed above. Section 2.3 of the [CCGUIDE] also explains that macOS will boot into the Recovery partition if a system integrity error occurs.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FPT_TST_EXT.3-WLAN-ATE-01

The evaluator shall perform the following tests:

- *Test 1: The evaluator shall perform the integrity check on a known good TSF executable and verify that the check is successful.*
- *Test 2: The evaluator shall modify the TSF executable, perform the integrity check on the modified TSF executable, and verify that the check fails.*

Summary

Test 1: This test is covered by Test 43 of FPT_TST_EXT.1, as the successful boot implies all integrity checks were successful.

Test 2: This test is covered by Test 44 of FPT_TST_EXT.1, as the failure to boot implies at least one integrity check failed.

Please refer to the test assurance activities for FPT_TST_EXT.1.

2.2.6.6 Trusted Update (FPT_TUD_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FPT_TUD_EXT.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 the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SIGN. The digital signature verification may be performed as part of a network protocol occurs over a trusted channel as described in FTP_ITC_EXT.1.) If the signature verification is not performed as part of a trusted channel, the evaluator will send a query response with a bad signature and verify that the signature verification fails. The evaluator will then send a query response with a good signature and verify that the signature verification is successful.

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 46: 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 the OS does not install the modified update.*
- *Test 47: 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

The evaluator verified macOS provides a list of available updates. The TOE communicates with the Apple Update Server over TLS as tested in FTP_ITC_EXT.1.

Test 46: The evaluator initiated an update to macOS, modified the update by flipping a bit in payloadv2.bom.signature, and verified that the update did not install due to an invalid signature.

Test 47: The evaluator initiated an update to macOS and verified that the update installed successfully.

2.2.6.7 Trusted Update for Application Software (FPT_TUD_EXT.2)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FPT_TUD_EXT.2-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 the response to this query is authentic by using a digital signature scheme specified in FCS_COP.1/SIGN. The digital signature verification may be performed as part of a network protocol as described in FTP_ITC_EXT.1. If the signature verification is not performed as part of a trusted channel, the evaluator will send a query response with a bad signature and verify that the signature verification fails. The evaluator will then send a query response with a good signature and verify that the signature verification is successful.

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 48: 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 49: 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

The evaluator verified macOS provides a list of available application updates via the App Store. The TOE communicates with the Apple Update Server over TLS as tested in FTP_ITC_EXT.1. The evaluator performed both Test 48 and Test 49 for two applications: Microsoft Excel and Xcode.

Test 48: The evaluator initiated an update to an application using the App Store, ensured the digital signature chained to a trusted Apple certificate, modified the update to invalidate the signature, and verified that the update did not install due to an invalid signature.

Test 49: The evaluator initiated an update to an application using the App Store, ensured the digital signature chained to a trusted Apple certificate, and verified that the update installed successfully.

2.2.7 TOE access (FTA)

2.2.7.1 Default TOE Access Banners (FTA_TAB.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-OSPP-FTA_TAB.1-ATE-01

The evaluator will configure the OS, per instructions in the OS manual, to display the advisory warning message "TEST TEST Warning Message TEST TEST". The evaluator will then log out and confirm that the advisory message is displayed before logging in can occur.

Summary

The evaluator added the advisory message to /Library/Security/PolicyBanner.txt, rebooted, and verified that the warning message is displayed before the log in screen.

2.2.7.2 Wireless Network Access (FTA_WSE_EXT.1)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FTA_WSE_EXT.1-ASE-01

The evaluator shall examine the TSS to determine that it defines SSIDs as the attribute to specify acceptable networks.

Summary

Section 7.1.7.2 of the [ST] states a list of known wireless networks (identified by SSID) can be used to restrict the TOE's access.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FTA_WSE_EXT.1-AGD-01

The evaluator shall examine the operational guidance to determine that it contains guidance for configuring the list of SSIDs that the WLAN Client is able to connect to.

Summary

Section 3.11.3 of the [CCGUIDE] describes how an administrator can prevent unprivileged users from configuring the wireless interface. This technique is used to restrict the list of known wireless networks (identified by SSID) that the TOE can connect to.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FTA_WSE_EXT.1-ATE-01

The evaluator shall perform the following tests for each attribute:

- *Test 1: The evaluator configures the TOE to allow a connection to a wireless network with a specific SSID. The evaluator configures the test environment such that the allowed SSID and an SSID that is not allowed are both "visible" to the TOE. The evaluator shall demonstrate that they can successfully establish a connection with the allowed SSID. The evaluator shall then attempt to establish a session with the disallowed SSID and observe that the attempt fails.*

Summary

The evaluator configured the TOE such that administrator authorization is required to change wireless networks, then added a known wireless network set to "Auto-Join". After this configuration, the evaluator verified that unprivileged users can only connect to the allowed wireless network.

2.2.8 Trusted path/channels (FTP)

2.2.8.1 Bluetooth Encryption (FTP_BLT_EXT.1)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.1-ASE-01

The evaluator shall verify that the TSS describes the use of encryption, the specific Bluetooth protocol(s) it applies to, and whether it is enabled by default.

The evaluator shall verify that the TSS includes the protocol used for encryption of the transmitted data and the key generation mechanism used.

Summary

Section 7.1.8.1 of the [ST] states that the TOE uses ECDH key exchange and AES for data encryption, applied to BR/EDR and LE, and this encryption is enabled and enforced by default.

Guidance Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.1-AGD-01

The evaluator shall verify that the operational guidance includes instructions on how to configure the TOE to require the use of encryption during data transmission (unless this behavior is enforced by default).

Summary

Section 4 of the [CCGUIDE] explains that the TOE always enforces encryption when transmitting data over Bluetooth, and this is not configurable.

Test Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.1-ATE-01

There are no test EAs for this component. Testing for this SFR is addressed through the evaluation of FTP_BLT_EXT.3/BR and, if claimed, FTP_BLT_EXT.3/LE.

Summary

Please refer to FTP_BLT_EXT.3/BR and FTP_BLT_EXT.3/LE.

2.2.8.2 Persistence of Bluetooth Encryption (FTP_BLT_EXT.2)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.2-ASE-01

The evaluator shall verify that the TSS describes the TSF's behavior if a remote device stops encryption while connected to the TOE.

Summary

Section 7.1.8.2 of the [ST] states that the TOE terminates the connection if the remote device stops encryption while connected to the TOE.

Guidance Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.2-AGD-01

The evaluator shall verify that the operational guidance describes how to enable/disable encryption (if configurable).

Summary

Section 4 of the [CCGUIDE] explains that the TOE always enforces encryption when transmitting data over Bluetooth, and this is not configurable.

Test Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.2-ATE-01

The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:

Step 1: Initiate pairing with the TOE from a remote Bluetooth device that has been configured to

have a minimum encryption key size that is equal to or greater than that of the TOE.

Step 2: After pairing has successfully finished and while a connection exists between the TOE and the remote device; turn off encryption on the remote device. This can be done using commercially-available tools.

Step 3: Verify that the TOE either restarts encryption with the remote device or terminates the connection with the remote device.

Summary

The evaluator configured a Bluetooth dongle to use a minimum encryption key size of 128 bits. Then, the evaluator successfully paired a Linux system to the TOE and confirmed that the encryption key size being used was 128 bits. Using hcitool, the evaluator disabled encryption on the Linux system and verified that the TOE terminated the connection.

2.2.8.3 Bluetooth Encryption Parameters (BR/EDR) (FTP_BLT_EXT.3/BR)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.3-BR-ASE-01

The evaluator shall examine the TSS and verify that it specifies the minimum key size for BR/EDR encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.

Summary

Section 7.1.8.3 of the [ST] states that the TOE always uses a 128-bit key size for encryption and this value is not configurable.

Guidance Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.3-BR-AGD-01

The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for BR/EDR encryption, if configurable.

Summary

Section 4 of the [CCGUIDE] explains that the TOE always uses 128-bit AES, and the encryption key size is not configurable.

Test Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.3-BR-ATE-01

The evaluator shall perform the following tests:

- ***Test 1:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*

  Step 1: Initiate BR/EDR pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

  Step 2: Use a Bluetooth packet sniffer to verify that the encryption key size negotiated for the connection is at least as large as the minimum encryption key size defined for the TOE.

- ***Test 2:*** *(conditional): If the encryption key size is configurable, configure the TOE to support a different minimum key size, then repeat Test 1 and verify that the negotiated key size is at least as large as the new minimum value.*

- ***Test 3:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*

  Step 1: Initiate BR/EDR pairing with the TOE from a remote Bluetooth device that has been configured to have a maximum encryption key size of 1 byte. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

  Step 2: Verify that the encryption key size suggested by the remote device is not accepted by the TOE and that the connection is not completed.

Summary

Test 1: The evaluator configured a Bluetooth dongle to use BR/EDR with a minimum encryption key size of 128 bits. Then, the evaluator successfully BR/EDR paired a Linux system to the TOE and confirmed that the encryption key size being used was 128 bits.

Test 2: Not applicable as the encryption size is not configurable.

Test 3: The evaluator used a modified Linux kernel and configured a Bluetooth dongle to use BR/EDR with a minimum encryption key size of 8 bits. The evaluator initiated BR/EDR pairing from a Linux system and verified that the TOE rejected the connection.

2.2.8.4 Bluetooth Encryption Parameters (LE) (FTP_BLT_EXT.3/LE)

TSS Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.3-LE-ASE-01

The evaluator shall examine the TSS and verify that it specifies the minimum key size for LE encryption, whether this value is configurable, and the mechanism by which the TOE will not negotiate keys sizes smaller than the minimum.

Summary

Section 7.1.8.3 of the [ST] states that the TOE always uses a 128-bit key size for encryption and this value is not configurable.

Guidance Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.3-LE-AGD-01

The evaluator shall verify that the guidance includes instructions on how to configure the minimum encryption key size for LE encryption, if configurable.

Summary

Section 4 of the [CCGUIDE] explains that the TOE always uses 128-bit AES, and the encryption key size is not configurable.

Test Assurance Activities

Assurance Activity AA-BTPPM-FTP_BLT_EXT.3-LE-ATE-01

The evaluator shall perform the following tests:

- ***Test 1:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*

  Step 1: Initiate LE pairing with the TOE from a remote Bluetooth device that has been configured to have a minimum encryption key size that is equal to or greater than that of the TOE. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

  Step 2: Use a Bluetooth packet sniffer to verify that the encryption key size negotiated for

the connection is at least as large as the minimum encryption key size defined for the TOE.

- ***Test 2:*** *(conditional): If the encryption key size is configurable, configure the TOE to support a different minimum key size, then repeat Test 1 and verify that the negotiated key size is at least as large as the new minimum value.*
- ***Test 3:*** *The evaluator shall perform the following steps using a Bluetooth protocol analyzer to observe packets pertaining to the encryption key size:*

  Step 1: Initiate LE pairing with the TOE from a remote Bluetooth device that has been configured to have a maximum encryption key size of 1 byte. This can be done using certain commercially-available tools that can send the appropriate command to certain commercially-available Bluetooth controllers.

  Step 2: Verify that the encryption key size suggested by the remote device is not accepted by the TOE and that the connection is not completed.

Summary

Test 1: The evaluator configured a Linux system to use LE with a minimum encryption key size of 128 bits. Then, the evaluator successfully LE paired the TOE to the Linux system and confirmed that the encryption key size being used was 128 bits.

Test 2: Not applicable as the encryption size is not configurable.

Test 3: The evaluator used a modified Linux kernel with a minimum encryption key size of 8 bits. The evaluator initiated LE pairing from a Linux system and verified that the TOE rejected the connection.

2.2.8.5 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-OSPP-FTP_ITC_EXT.1-ATE-01

[TD0789] The evaluator shall configure the OS to communicate with another trusted IT product as identified in the third selection. The evaluator shall monitor network traffic while the OS performs communication with each of the servers identified in the third selection. The evaluator shall ensure that for each session a trusted channel was established in conformance with the selected protocols.

Summary

The [ST] identifies two authorized IT entities: “Apple Update Server” and “application initiated TLS”.

The evaluator initiated an update to macOS and collected packet logs using Wireshark. The evaluator confirmed from the packet log that macOS established a TLS connection with the Apple Update Server.

“Application initiated TLS” was tested as part of FCS_TLSC_EXT.1. Please refer to the test assurance activities for FCS_TLSC_EXT.1.

2.2.8.6 Trusted Channel Communication (Wireless LAN) (FTP_ITC.1/WLAN)

TSS Assurance Activities

Assurance Activity AA-WLANPPM-FTP_ITC.1-WLAN-ASE-01

The evaluator shall examine the TSS to determine that it describes the details of the TOE connecting to an access point in terms of the cryptographic protocols specified in the requirement, along with TOE-specific options or procedures that might not be reflected in the specification. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

Summary

Section 7.1.8.4 of the [ST] describes the details of the 802.11-2012 and 802.1X protocols used by the TOE. Section 7.1.2.13 of the [ST] describes the details of the EAP-TLS protocol. The evaluator verified all protocols listed in the TSS are specified and included in the requirements in the ST.

Guidance Assurance Activities

Assurance Activity AA-WLANPPM-FTP_ITC.1-WLAN-AGD-01

The evaluator shall confirm that the operational guidance includes instructions for establishing the connection to the access point and that it includes recovery instructions should a connection be unintentionally broken.

Summary

Section 3.11.2 of the [CCGUIDE] describes how to join a Wi-Fi network. If a connection is broken, macOS can be configured to automatically reconnect, or the user can reconnect manually using the same steps.

Test Assurance Activities

Assurance Activity AA-WLANPPM-FTP_ITC.1-WLAN-ATE-01

The evaluator shall perform the following tests:

- *Test 1: The evaluator shall ensure that the TOE is able to initiate communications with an access point using the protocols specified in the requirement by setting up the connections as described in the operational guidance and ensuring that communications are successful.*
- *Test 2: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.*
- *Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, modification of the channel data is detected by the TOE.*
- *Test 4: The evaluators shall physically interrupt the connection from the TOE to the access point (e.g., moving the TOE host out of range of the access point, turning the access point off). The evaluators shall ensure that subsequent communications are appropriately protected, at a minimum in the case of any attempts to automatically resume the connection or connect to a new access point.*

Summary

The evaluator started an access point using the hostapd application. All packets were logged using Wireshark. A configuration profile was installed to instruct the TOE to set up WLAN connections.

The CA certificate used to issue the certificate of the access point EAP-TLS server was trusted by the TOE.

Test 1: The evaluator instructed the TOE to connect to the test access point and verified the connection was successful.

Test 2: The evaluator inspected the packet log and found that all channel data was protected using EAP-TLS, CCMP, and GCMP, ensuring the confidentiality of the data.

Test 3: The evaluator inspected the packet log and found that all channel data was protected using EAP-TLS, CCMP, and GCMP, ensuring the integrity of the data.

Test 4: The evaluator terminated the test access point and verified the TOE disconnected. The evaluator then started the test access point again, verified the TOE automatically connected, and all channel data was again protected using EAP-TLS, CCMP, and GCMP.

2.2.8.7 Trusted Path (FTP_TRP.1)

TSS Assurance Activities

Assurance Activity AA-OSPP-FTP_TRP.1-ASE-01

[TD0839] The evaluator will examine the TSS to determine that the methods of remote or local OS administration are indicated, along with how those communications are protected. (Conditional: if "remote" is selected in FTP_TRP1.1) 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

Section 7.1.8.5 of the [ST] specifies the means of local OS administration through a trusted path that is restricted to physical access and is initiated by the local user protected by the user's authentication credentials. There is no remote administration.

Guidance Assurance Activities

Assurance Activity AA-OSPP-FTP_TRP.1-AGD-01

[TD0839] The evaluator will confirm that the operational guidance contains instructions for establishing the remote administrative sessions or initial user authentication for each supported method.

Summary

Section 3.2.4 of the [CCGUIDE] describes initial user authentication for local users. There is no remote administration.

Test Assurance Activities

Assurance Activity AA-OSPP-FTP_TRP.1-ATE-01

[TD0839] The evaluator will also perform the following tests:

- *Test 78: The evaluator will ensure that communications using each remote or local administration method is tested during the course of the evaluation, setting up the connections or initial user authentication as described in the operational guidance and ensuring that communication is successful.*
- *Test 79 (Conditional: if "remote" is selected in FTP_TRP1.1): 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 80(Conditional: if "remote" is selected in FTP_TRP1.1): The evaluator will ensure, for*

each method of remote administration, the channel data is not sent in plaintext.

- *Test 81(Conditional: if "remote" is selected in FTP_TRP1.1): The evaluator will ensure, for each method of remote administration, modification of the channel data is detected by the OS.*

Summary

Test 78: Initial user authentication is covered by Test 56 and Test 58 of FIA_UAU.5. Please refer to the test assurance activities for FIA_UAU.5.

Test 79: Not applicable as the [ST] does not select "remote".

Test 80: Not applicable as the [ST] does not select "remote".

Test 81: Not applicable as the [ST] does not select "remote".

2.3 Security Assurance Requirements

2.3.1 Development (ADV)

2.3.1.1 Basic Functional Specification (ADV_FSP.1)

There are no specific evaluation activities associated with these SARs, except ensuring the information is provided. The functional specification documentation is provided to support the evaluation activities described in Section 5.1 Security Functional Requirements, and other activities described for AGD, ATE, and AVA SARs. The requirements on the content of the functional specification information is implicitly assessed by virtue of the other evaluation activities being performed; if the evaluator is unable to perform an activity because there is insufficient interface information, then an adequate functional specification has not been provided.

2.3.2 Guidance documents (AGD)

2.3.2.1 Operational User Guidance (AGD_OPE.1)

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

Some of the contents of the operational guidance are verified by the evaluation activities in Section 5.1 Security Functional Requirements and evaluation of the OS according to the [CEM]. The following additional information is also required. If cryptographic functions are provided by the OS, the operational guidance ill contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the OS. It will 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 will make it clear to an administrator which security functionality is covered by the evaluation activities.

Summary

Section 1.7 of the [CCGUIDE] states that the Apple corecrypto cryptographic libraries and the Broadcom chip are the only cryptographic providers associated with the TOE. No configuration is required, and other cryptographic engines were not evaluated or tested and should not be used.

Section 2 of the [CCGUIDE] provides instructions for obtaining, initiating, and installing OS updates. This section states that signature verification of the update is performed by the TOE before the update is installed. If the update is installed successfully, macOS will boot normally, otherwise the TOE will boot into macOS Recovery.

Sections 1.2, 1.3, and 3.1 of the [CCGUIDE] describe the scope of the evaluation and explicitly list the functionalities of VPN and Siri as not covered by the evaluation activities.

2.3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activity AA-OSPP-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 OS functional requirements. The evaluator will check to ensure that the guidance provided for the OS adequately

addresses all platforms claimed for the OS in the ST.

Summary

While performing other assurance activities, the evaluator determined that sufficient guidance was provided for the TOE to address all claimed platforms. The guidance provided for the TOE applies to all platforms claimed in the [ST].

2.3.3 Life-cycle support (ALC)

2.3.3.1 Labeling of the TOE (ALC_CMC.1)

Assurance Activity AA-OSPP-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

Section 1.4 of the [ST] identifies the TOE as “Apple macOS 26 Tahoe” with the specific tested version of the TOE identified as “macOS 26.3.” Section 1.1 of the [CCGUIDE] identifies the TOE version identically as macOS 26.3. The evaluator was able to install macOS on the devices covered by this evaluation and found that it matches the version in the [ST] and the [CCGUIDE].

2.3.3.2 TOE CM Coverage (ALC_CMS.1)

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

The evaluator 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 will 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 examined the lifecycle evidence provided by the vendor, which contains the following information:

Xcode IDE is used for building apps for Mac, iPhone, and iPad. Xcode IDE is tightly integrated with the Cocoa framework. See https://developer.apple.com/xcode/ for more details on Xcode IDE.

The vendor provided “Apple Platform Security”, which contains the section *App security in macOS*. This section details the runtime security features employed by macOS, by default, to prevent vulnerabilities by ensuring that only trusted software runs on a user’s system. Additionally, the section *Overview of additional macOS system security capabilities* describes more security features employed by macOS, such as system integrity protection, signed system volume security, protection for peripherals, and DMA protections.

Finally, the evaluator verified the TSF is uniquely identified as part of the assurance activities for ALC_CMC.1.

2.3.3.3 Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activity AA-OSPP-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 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 will 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

Section 7.2 of the [ST] contains a description of the security update process. The evaluator verified that this section covers the entire application and describes the developer’s process in detail. There are no third-party processes as the security updates are solely developed by Apple.

As explained in this section, security updates are made available via the Apple Update Server. Since Apple does not disclose security issues until updates are available, the [ST] asserts that there are 0 days between public disclosure and availability of the update. The entire update deployment process is controlled by Apple, so there are no third-party delays.

Finally, Section 7.2 contains a website and an email address where potential vulnerabilities can be reported. The website is protected by a trusted channel (HTTPS), whereas the email can be encrypted using the Apple Product Security PGP key.

2.3.4 Tests (ATE)

2.3.4.1 Independent Testing – Conformance (ATE_IND.1)

Assurance Activity AA-OSPP-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 will determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's evaluation activities.

While it is not necessary to have one test case per test listed in an evaluation activity, the evaluator

must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no 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 will be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

Summary

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

The test environment was set up according to a setup strategy that followed the evaluated configuration requirements specified in the guidance [CCGUIDE] and Security Target [ST], supplemented by configurations required to perform testing. The testing was performed on a subset of the platforms claimed in the [ST]: only one model corresponding to each processor series was fully tested. The tested models are Mac14,2; Mac15,8; Mac16,8; and Mac17,2. Additional protocol-level Bluetooth and WLAN testing was performed on Mac14,7. The DTR contains a detailed rationale for each equivalency argument considered during testing.

The following tools were used for testing:

- Lenovo ThinkPad T560, running Arch Linux
  - nftables for enabling masquerading in the Linux kernel version 6.17.8
  - GNU netcat, version 0.7.1
  - dnsmasq, version 2.91
  - hostapd, version 2.11
  - iw, version 6.17
  - netstat (net-tools, version 2.10)
  - OpenSSL 3.2.0 or newer
  - Wireshark, version 4.6.0 with libpcap 1.10.5
  - Bluez 5.84 and tools

- MacBook, running macOS 26.3 Tahoe (the TOE)
  - OpenSSL 3.2.0 or newer
  - Wireshark, version 4.6.2 (including tshark)
  - YubiKey Manager, version 7.3.0
  - LLDB

The machines are connected to the internal network using a wired connection.

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

2.3.5 Vulnerability assessment (AVA)

2.3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activity AA-OSPP-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

As required by NIAP Policy 17 Addendum – vulnerability mitigation guidance, the following search terms were used during the vulnerability search:

- macOS 26
- XNU
- curl
- libarchive
- libexpat
- libxml2
- libxslt
- sudo
- WebKit
- Safari
- Apple Mail
- AirDrop
- AirPlay
- Secure Enclave Processor

- BCM4388
- BCM4387
- BCM4378
- corecrypto
- IEEE 802.11
- EAP-TLS
- TLS
- Bluetooth

The evaluator searched for publicly known vulnerabilities using the following sources:

- MITRE Common Vulnerabilities and Exposures (CVE) List:
  - https://cve.mitre.org/cve/
- National Vulnerability Database (NVD):
  - https://nvd.nist.gov/
- CISA Known Exploited Vulnerabilities (KEV) Catalog:
  - https://www.cisa.gov/known-exploited-vulnerabilities-catalog
- Apple security releases
  - https://support.apple.com/en-us/100100
- Background Security Improvements by date
  - https://support.apple.com/en-us/111333
- curl vulnerabilities
  - https://curl.se/docs/security.html

The vulnerability search was repeated on the following dates, throughout the evaluation process (last search date: 2026-07-14):

- 2026-06-26
- 2026-06-29
- 2026-07-07
- 2026-07-14

All “crucial” vulnerabilities (as defined by the NIAP Policy 17 Addendum) found during the vulnerability search were mitigated. Any residual vulnerabilities are not considered crucial.

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

A References

CC **Common Criteria for Information Technology Security Evaluation**
Version CC:3.1 R5
Date April 2017
Location https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf
Location https://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf
Location https://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf

CCEVS-LG **CCEVS LabGrams**
Author(s) NIAP
Date July 2026
Location https://www.niap-ccevs.org/labgrams

CCEVS-PL **CCEVS Scheme Policy Letters**
Author(s) NIAP
Date July 2026
Location https://www.niap-ccevs.org/policies

CCEVS-PUB **CCEVS Scheme Publications**
Author(s) NIAP
Date July 2026
Location https://www.niap-ccevs.org/publications

CCEVS-TD **Technical Decisions**
Author(s) NIAP
Date July 2026
Location https://www.niap-ccevs.org/technical-decisions

CCGUIDE **Apple macOS 26 Tahoe Common Criteria Guide**
Date 2026-07-02
File name st-vid11695-agd.pdf

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

ST **Apple macOS 26 Tahoe Security Target**
Date 2026-07-02
File name st-vid11695-st.pdf

B 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.