



Apple macOS 13 Ventura Assurance Activity Report

Version: 1.1
Date: 2024-01-12
Status: RELEASED
Classification: Public
Filename: VID11347_SER_AAR_Apple_macOS_13_Ventura_v1.1
Product: Apple macOS 13 Ventura
Sponsor: Apple Inc.
Evaluation Facility: atsec information security corporation
Validation ID: 11347
Validation Body: NIAP CCEVS
Author(s): Chao (Alex) Gong, Joachim Vandersmissen, Walker Riley, Stephan Mueller
Quality Assurance: Trang Huynh

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

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

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.

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

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

Classification Note

Public Information (public)

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

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

Revision History

Version	Date	Author(s)	Changes to Previous Revision	Application Notes
1.0	2023-11-22	Chao (Alex) Gong, Joachim Vandersmissen, Walker Riley, Stephan Mueller	First version	
1.1	2024-01-12	Chao (Alex) Gong, Joachim Vandersmissen, Walker Riley, Stephan Mueller	Address ECR comments	

Table of Contents

1	Evaluation Basis and Documents	9
2	Evaluation Results	10
2.1	CAVP Summary	10
2.2	Security Functional Requirements	14
2.2.1	Security audit (FAU)	14
2.2.1.1	Audit Data Generation (Refined) (FAU_GEN.1)	14
	FAU_GEN.1.1	14
	FAU_GEN.1.2	16
2.2.1.2	Audit Data Generation (Bluetooth) (FAU_GEN.1/BT)	17
	TSS Assurance Activities	17
	Guidance Assurance Activities	17
	Test Assurance Activities	17
2.2.2	Cryptographic support (FCS)	17
2.2.2.1	Cryptographic Key Generation (Refined) (FCS_CKM.1)	17
	TSS Assurance Activities	17
	Guidance Assurance Activities	19
	Test Assurance Activities	20
2.2.2.2	Cryptographic Key Establishment (Refined) (FCS_CKM.2)	22
	TSS Assurance Activities	22
	Guidance Assurance Activities	22
	Test Assurance Activities	23
2.2.2.3	Cryptographic Key Destruction (FCS_CKM_EXT.4)	24
	TSS Assurance Activities	24
	Guidance Assurance Activities	25
	Test Assurance Activities	26
2.2.2.4	Bluetooth Key Generation (FCS_CKM_EXT.8)	27
	TSS Assurance Activities	27
	Guidance Assurance Activities	27
	Test Assurance Activities	27
2.2.2.5	Cryptographic Operation - Encryption/Decryption (Refined) (FCS_COP.1(1))	28
	TSS Assurance Activities	28
	Guidance Assurance Activities	28
	Test Assurance Activities	29
2.2.2.6	Cryptographic Operation - Hashing (Refined) (FCS_COP.1(2))	34
	TSS Assurance Activities	34
	Guidance Assurance Activities	34
	Test Assurance Activities	34
2.2.2.7	Cryptographic Operation - Signing (Refined) (FCS_COP.1(3))	35
	TSS Assurance Activities	35
	Guidance Assurance Activities	35
	Test Assurance Activities	35
2.2.2.8	Cryptographic Operation - Keyed-Hash Message Authentication (Refined) (FCS_COP.1(4))	36
	TSS Assurance Activities	36
	Guidance Assurance Activities	36

Test Assurance Activities	36
2.2.2.9 Random Bit Generation (FCS_RBG_EXT.1)	36
FCS_RBG_EXT.1.1	36
FCS_RBG_EXT.1.2	37
2.2.2.10 Storage of Sensitive Data (FCS_STO_EXT.1)	38
TSS Assurance Activities	38
Guidance Assurance Activities	39
Test Assurance Activities	39
2.2.2.11 TLS Client Protocol (FCS_TLSC_EXT.1)	39
FCS_TLSC_EXT.1.1	39
FCS_TLSC_EXT.1.2	42
FCS_TLSC_EXT.1.3	44
2.2.2.12 TLS Client Protocol (FCS_TLSC_EXT.2)	45
TSS Assurance Activities	45
Guidance Assurance Activities	45
Test Assurance Activities	45
2.2.2.13 TLS Client Protocol (FCS_TLSC_EXT.4)	46
TSS Assurance Activities	46
Guidance Assurance Activities	46
Test Assurance Activities	46
2.2.3 User data protection (FDP)	47
2.2.3.1 Access Controls for Protecting User Data (FDP_ACF_EXT.1)	47
TSS Assurance Activities	47
Guidance Assurance Activities	47
Test Assurance Activities	47
2.2.4 Identification and authentication (FIA)	48
2.2.4.1 Authentication failure handling (Refined) (FIA_AFL.1)	48
FIA_AFL.1.1	48
FIA_AFL.1.2	49
2.2.4.2 Bluetooth User Authorization (FIA_BLT_EXT.1)	49
TSS Assurance Activities	49
Guidance Assurance Activities	49
Test Assurance Activities	50
2.2.4.3 Bluetooth Mutual Authentication (FIA_BLT_EXT.2)	50
TSS Assurance Activities	50
Guidance Assurance Activities	50
Test Assurance Activities	51
2.2.4.4 Rejection of Duplicate Bluetooth Connections (FIA_BLT_EXT.3)	51
TSS Assurance Activities	51
Guidance Assurance Activities	51
Test Assurance Activities	51
2.2.4.5 Secure Simple Pairing (FIA_BLT_EXT.4)	52
TSS Assurance Activities	52
Guidance Assurance Activities	52
Test Assurance Activities	52
2.2.4.6 Trusted Bluetooth Device User Authorization (FIA_BLT_EXT.6)	52
TSS Assurance Activities	52

Guidance Assurance Activities	53
Test Assurance Activities	53
2.2.4.7 Untrusted Bluetooth Device User Authorization (FIA_BLT_EXT.7)	53
TSS Assurance Activities	53
Guidance Assurance Activities	53
Test Assurance Activities	53
2.2.4.8 Multiple Authentication Mechanisms (Refined) (FIA_UAU.5)	54
FIA_UAU.5.1	54
FIA_UAU.5.2	56
2.2.4.9 X.509 Certificate Validation (FIA_X509_EXT.1)	57
FIA_X509_EXT.1.1	57
FIA_X509_EXT.1.2	60
2.2.4.10 X.509 Certificate Authentication (FIA_X509_EXT.2)	61
TSS Assurance Activities	61
Guidance Assurance Activities	61
Test Assurance Activities	61
2.2.5 Security management (FMT)	61
2.2.5.1 Management of security functions behavior (FMT_MOF_EXT.1)	61
TSS Assurance Activities	61
Guidance Assurance Activities	62
Test Assurance Activities	62
2.2.5.2 Management of Security Functions Behavior (FMT_MOF_EXT.1/BT)	62
TSS Assurance Activities	62
Guidance Assurance Activities	63
Test Assurance Activities	63
2.2.5.3 Specification of Management Functions (FMT_SMF_EXT.1)	63
TSS Assurance Activities	63
Guidance Assurance Activities	63
Test Assurance Activities	65
2.2.5.4 Specification of Management Functions (FMT_SMF_EXT.1/BT)	66
TSS Assurance Activities	66
Guidance Assurance Activities	66
Test Assurance Activities	67
2.2.6 Protection of the TSF (FPT)	68
2.2.6.1 Access controls (FPT_ACF_EXT.1)	68
FPT_ACF_EXT.1.1	68
FPT_ACF_EXT.1.2	70
2.2.6.2 Address Space Layout Randomization (FPT_ASLR_EXT.1)	70
TSS Assurance Activities	70
Guidance Assurance Activities	70
Test Assurance Activities	70
2.2.6.3 Stack Buffer Overflow Protection (FPT_SBOP_EXT.1)	71
TSS Assurance Activities	71
Guidance Assurance Activities	72
Test Assurance Activities	72
2.2.6.4 Boot Integrity (FPT_TST_EXT.1)	72
TSS Assurance Activities	72

Guidance Assurance Activities	73
Test Assurance Activities	73
2.2.6.5 Trusted Update (FPT_TUD_EXT.1)	73
FPT_TUD_EXT.1.1	73
FPT_TUD_EXT.1.2	74
2.2.6.6 Trusted Update for Application Software (FPT_TUD_EXT.2)	74
FPT_TUD_EXT.2.1	74
FPT_TUD_EXT.2.2	75
2.2.6.7 Write XOR Execute Memory Pages (FPT_W^X_EXT.1)	76
TSS Assurance Activities	76
Guidance Assurance Activities	76
Test Assurance Activities	76
2.2.7 TOE access (FTA)	76
2.2.7.1 Default TOE access banners (FTA_TAB.1)	76
TSS Assurance Activities	76
Guidance Assurance Activities	76
Test Assurance Activities	77
2.2.8 Trusted path/channels (FTP)	77
2.2.8.1 Bluetooth Encryption (FTP_BLT_EXT.1)	77
TSS Assurance Activities	77
Guidance Assurance Activities	77
Test Assurance Activities	77
2.2.8.2 Persistence of Bluetooth Encryption (FTP_BLT_EXT.2)	78
TSS Assurance Activities	78
Guidance Assurance Activities	78
Test Assurance Activities	78
2.2.8.3 Bluetooth Encryption Parameters (BR/EDR) (FTP_BLT_EXT.3/BR)	79
TSS Assurance Activities	79
Guidance Assurance Activities	79
Test Assurance Activities	79
2.2.8.4 Bluetooth Encryption Parameters (LE) (FTP_BLT_EXT.3/LE)	80
TSS Assurance Activities	80
Guidance Assurance Activities	80
Test Assurance Activities	80
2.2.8.5 Trusted channel communication (FTP_ITC_EXT.1)	81
TSS Assurance Activities	81
Guidance Assurance Activities	81
Test Assurance Activities	81
2.2.8.6 Trusted Path (FTP_TRP.1)	81
TSS Assurance Activities	81
Guidance Assurance Activities	82
Test Assurance Activities	82
2.3 Security Assurance Requirements	83
2.3.1 Guidance documents (AGD)	83
2.3.1.1 Operational user guidance (AGD_OPE.1)	83
2.3.1.2 Preparative procedures (AGD_PRE.1)	83
2.3.2 Life-cycle support (ALC)	84

2.3.2.1	Labelling of the TOE (ALC_CMC.1)	84
2.3.2.2	TOE CM coverage (ALC_CMS.1)	84
2.3.2.3	Extension: Timely Security Updates (ALC_TSU_EXT.1)	85
2.3.3	Tests (ATE)	86
2.3.3.1	Independent testing - conformance (ATE_IND.1)	86
2.3.4	Vulnerability assessment (AVA)	87
2.3.4.1	Vulnerability survey (AVA_VAN.1)	87
A	Appendixes	91
A.1	References	91
A.2	Glossary	94

List of Tables

Table 1: Cryptographic algorithm table	10
Table 2: Coverage of CAVP certificates for Intel Processors	13
Table 3: Auditable events (Bluetooth)	17
Table 4: Cryptographic algorithm table	18
Table 5: Cryptographic algorithms for FCS_CKM.1 Cryptographic key generation	19
Table 6: Cryptographic algorithms for FCS_CKM.1 Cryptographic key generation	19
Table 7: Cryptographic algorithms for FCS_CKM.2 Cryptographic key establishment	22
Table 8: Cryptographic algorithms for FCS_CKM.2 Cryptographic key establishment	23
Table 9: Cryptographic algorithms for FCS_COP.1(1) Cryptographic operation - encryption/ decryption	28
Table 10: Storage of sensitive data	38
Table 11: Management functions (Bluetooth)	63
Table 12: Management functions (Operating System)	64

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_V1.0] : PP-Configuration for General Purpose Operating Systems and Bluetooth, Version 1.0, dated 2021-04-15, which consists of the following components:
 - [OSPPv4.2.1] : Protection Profile for General Purpose Operating Systems, Version 4.2.1, dated 2019-04-22.
 - [BT] : PP-Module for Bluetooth, Version 1.0, dated 2021-04-15.

This evaluation claims Exact Compliance with the above PP-Configuration, PP and PP-Modules.

The following scheme documents and interpretations have been considered:

- [CCEVS-LG] : "CCEVS LabGrams", version as of November 2023.
- [CCEVS-PL] : "CCEVS Scheme Policy Letters", version as of November 2023.
- [CCEVS-PUB] : "CCEVS Scheme Publications", version as of November 2023.
- [CCEVS-TD] : "Technical Decisions", version as of November 2023.

2 Evaluation Results

This Assurance Activity Report covers the Apple macOS 13 Ventura evaluation which claims conformance to the following Protection Profiles:

- [\[CFG_GPOS-BT_V1.0\]](#): PP-Configuration for General Purpose Operating Systems and Bluetooth. Version 1.0 as of 2021-04-15; exact conformance
 - [\[OSPPv4.2.1\]](#): Protection Profile for General Purpose Operating Systems. Version 4.2.1 as of 2019-04-22; exact conformance.
 - [\[BT\]](#): PP-Module for Bluetooth. Version 1.0 as of 2021-04-15; exact conformance.

This report describes the evaluation team's assessment of the assurance activities and their results.

2.1 CAVP Summary

The CAVP certificates contain several different SoCs and micro-architectures in the operational environment (OE). The relationship between the SoCs and micro-architectures used by the hardware platforms claimed in this evaluation are specified in Appendix A.1 of the Security Target.

The following convention is used in the tables of this appendix to identify the cryptographic modules.

KRN

- Apple corecrypto Module v13.0 [Apple ARM, Kernel, Software, SL1], or
- Apple corecrypto Module v13.0 [Intel, Kernel, Software, SL1]

SEP

- SEP Hardware v2.0 in Apple silicon, or
- SEP Hardware v2.0 in Apple T2

SKS

- Apple corecrypto Module v13.0 [Apple silicon, Secure Key Store, Hardware, SL2], or
- Apple corecrypto Module v13.0 [Apple ARM, Secure Key Store, Hardware, SL2]

USR

- Apple corecrypto Module v13.0 [Apple ARM, User, Software, SL1], or
- Apple corecrypto Module v13.0 [Intel, User, Software, SL1]

BC

The Broadcom core hardware module implementation names are:

- Crypto Hardware Module aes_core_gcm.vhd for Broadcom chip models 4378, 4387 and 4388.
- Cryptographic Hardware Module for Broadcom chip models 4355, 4364 and 4377.

The **T2** is marketed as Apple ARM technology and runs T2OS 13.

Table 1: Cryptographic algorithm table

SFR	Algorithm	Capabilities	Mod	Implementatio	CAVP
FCS_CKM.1	RSA KeyGen	2048, 3072, 4096	USR	vng_ltc	A3488
	[FIPS186-4]	(Apple silicon)			
		2048, 3072, 4096	USR	c_avx2	A3506
		(Intel)			
	ECDSA KeyGen	P-256, P-384, P-521	USR	vng_ltc	A3488

SFR	Algorithm	Capabilities	Mod	Implementatio	CAVP	
	[FIPS186-4]	(Apple silicon)				
		P-256, P-384, P-521 (Intel)	USR	c_avx2	A3506	
FCS_CKM.2	RSA Key Establishment [RFC8017]	2048, 3072, 4096 (Apple silicon)	USR	c_ltc	Tested by the CCTL following the Assurance Activity for FCS_CKM.2.	
		2048, 3072, 4096 (Intel)	USR	c_ltc		
	ECC Key Establishment (KAS-ECC-SSC Sp800-56Ar3) [SP800-56A-Rev3]	P-256, P-384, P-521 (Apple silicon)	USR	c_ltc	A3486	
		P-256, P-384, P-521 (Intel)	USR	c_ltc	A3509	
FCS_COP.1-1	AES-CBC [SP800-38A]	128-bit, 256-bit (Apple silicon)	USR	asm_arm	A3483	
		128-bit, 256-bit (Intel)	USR	asm_aesni	A3501	
	AES-GCM [SP800-38D]	128-bit, 256-bit (Apple silicon)	USR	vng_asm	A3487	
		128-bit, 256-bit (Intel)	USR	vng_asm	A3510	
	AES-CCM [SP800-38C]	128-bit		BC	4388	AES 5926
					4387	AES 5926
					4378	AES 5926
					4377	AES 4791
					4364	AES 3678
	4355	AES 3678				
FCS_COP.1-2	SHS Byte-oriented mode [FIPS186-4]	SHA-1, SHA-256, SHA-384, SHA-512 (Apple silicon)	USR	vng_ltc	A3488	
		SHA-256 (Apple silicon)	KRN	vng_ltc	A3521	
			SKS	vng_ltc	A4259	
		SHA-1, SHA-256, SHA-384, SHA-512	USR	c_avx2	A3506	

SFR	Algorithm	Capabilities	Mod	Implementatio	CAVP
		(Intel)			
		SHA-256	KRN	c_avx2	A3623
		(Intel)			
		SHA-256	SKS	vng_neon	A4110
		(T2)			
FCS_COP.1-3	RSA SigVer [FIPS186-4]	Modulo: 2048, 3072, 4096 with: SHA-1, SHA-256, SHA-384, SHA-512 (Apple silicon)	USR	vng_ltc	A3488
		Modulo: 2048, 3072, 4096 with: SHA-1, SHA-256, SHA-384, SHA-512 (Intel)	USR	c_avx2	A3506
		Modulo: 4096 with: SHA-256 (Intel)	KRN	c_avx2	A3623
		Modulo: 4096 with: SHA-256 (T2)	SKS	vng_ltc	A4109
	ECDSA SigVer [FIPS186-4]	P-256, P-384, P-521 with: SHA-1, SHA-256, SHA-384, SHA-512 (Apple silicon)	USR	vng_ltc	A3488
		P-521 with: SHA-512 (Apple silicon)	KRN	vng_ltc	A3521
			SKS	vng_ltc	A4259
		P-256, P-384, P-521 with: SHA-1, SHA-256, SHA-384, SHA-512 (Intel)	USR	vng_ltc	A3506
FCS_COP.1-4	HMAC Byte-oriented mode [FIPS198-1]	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (Apple silicon)	USR	vng_ltc	A3488
		HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	USR	c_avx2	A3506

SFR	Algorithm	Capabilities	Mod	Implementatio	CAVP
		(Intel)			
FCS_RBG_EXT.1	CTR_DRBG [SP800-90A-Rev1] 📄	AES-256	USR	vng_asm	A3487
		(Apple silicon)	SEP	M2 Max (skg) M2 Pro (skg) M2 (skg) M1 Ultra (skg) M1 Max (skg) M1 Pro (skg)	A3490
				M1 (skg)	A1362
		AES-256	USR	vng_asm	A3510
		(Intel)			
		AES-256	SEP	T2 (skg)	DRBG 2029
		(T2)			

The following table shows the coverage of CAVP tests for the Intel processors used in the hardware platforms covered by this evaluation and specified in Appendix A.1 of the Security Target. For those processor models not tested, the last column indicates the equivalent processor on which the CAVP tests were performed. The equivalence argument for these processors is that the reference testing is performed on a processor of the same Intel Micro Architecture and Intel processor Generation.

Table 2: Coverage of CAVP certificates for Intel Processors

Intel Processor	Gen	Micro Architecture	Cryptographic Modules		Equivalent processor
			USR	KRN	
Xeon W-2140B	W	Skylake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Xeon W-2150B	W	Skylake			Xeon W-2140B
Xeon W-2170B	W	Skylake			Xeon W-2140B
Xeon W-2190B	W	Skylake			Xeon W-2140B
Xeon W-3223	W	Cascade Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Xeon W-3235	W	Cascade Lake			Xeon W-3223
Xeon W-3245	W	Cascade Lake			Xeon W-3223
Xeon W-3265	W	Cascade Lake			Xeon W-3223
Xeon W-3265M	W	Cascade Lake			Xeon W-3223
Xeon W-3275M	W	Cascade Lake			Xeon W-3223
Core i5-8210Y	8 th	Amber Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Core i5-8257U	8 th	Coffee Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Core i5-8259U	8 th	Coffee Lake			Core i5-8257U

Intel Processor	Gen	Micro Architecture	Cryptographic Modules		Equivalent processor
			USR	KRN	
Core i5-8279U	8 th	Coffee Lake			Core i5-8257U
Core i7-8557U	8 th	Coffee Lake			Core i5-8257U
Core i7-8559U	8 th	Coffee Lake			Core i5-8257U
Core i7-8569U	8 th	Coffee Lake			Core i5-8257U
Core i5-8500B	8 th	Coffee Lake			Core i7-8700B
Core i7-8700B	8 th	Coffee Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Core i7-8750H	8 th	Coffee Lake			Core i7-8700B
Core i7-8850H	8 th	Coffee Lake			Core i7-8700B
Core i9-8950HK	8 th	Coffee Lake			Core i7-8700B
Core i7-9750H	9 th	Coffee Lake			Core i9-9880H
Core i9-9880H	9 th	Coffee Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Core i9-9880HK	9 th	Coffee Lake			Core i9-9880H
Core i5-10500	10 th	Comet Lake			Core i7-10700K
Core i5-10600	10 th	Comet Lake			Core i7-10700K
Core i7-10700K	10 th	Comet Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Core i9-10910	10 th	Comet Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Core i5-1030NG7	10 th	Ice Lake			Core i7-1060NG7
Core i5-1038NG7	10 th	Ice Lake			Core i7-1060NG7
Core i7-1060NG7	10 th	Ice Lake	A3501 , A3506 , A3509 , A3510	A3623	Tested
Core i7-1068NG7	10 th	Ice Lake			Core i7-1060NG7

2.2 Security Functional Requirements

2.2.1 Security audit (FAU)

2.2.1.1 Audit Data Generation (Refined) (FAU_GEN.1)

FAU_GEN.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-FAU_GEN.1.1-AGD-01

The evaluator 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.

Summary

Section 6 *Audit Logs* of [CCGUIDE] lists the following types of auditable events for which the TOE generates audit records:

- Start-up of the audit functions;
- Shut-down of the audit functions;
- Authentication events;
- Use of privileged/special rights events for configuration changes;
- Privilege or role escalation events;
- Administrator or root-level access events.

The evaluator compared the [CCGUIDE] with the [ST], and verified that the same set of auditable events is listed in both documents.

Test Assurance Activities

Assurance Activity AA-FAU_GEN.1.1-ATE-01

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 have the proper entries.

Summary

The auditd utility was used to enable and disable auditing, as well as verify the presence of audit records for the events specified in the [ST]. Before each test, the auditd logs were rotated to exclude any unnecessary records from the evidence. Then, the auditreduce tool was used to read the log files. The evaluator verified the following auditable events:

- Start-up of auditd, by presence of "Audit startup" in the log file.
- Shutdown of auditd, by presence of "Audit shutdown" in the log file.
- Successful user authentication, by presence of "Verify password for record type Users 'testuser'" with outcome "success" in the log file.
- Unsuccessful user authentication, by presence of "Verify password for record type Users 'testuser'" with outcome "failure" in the log file.
- Creation and deletion of user accounts, by presence of "GeneratedUID" and "Delete record type Users 'testuserexp'" with outcome "success" in the log file.
- Successful password modification (privileged operation) by a user, by presence of "Modify password for record type Users 'testuser'" with outcome "success" in the log file.
- Unsuccessful password modification (privileged operation) by a user, by presence of "Modify password for record type Users 'testuser'" with outcome "failure" in the log file.
- Successful password modification (privileged operation) by an administrator, by presence of "Modify password for record type Users 'testuser'" with outcome "success" in the log file.
- Unsuccessful password modification (privileged operation) by an administrator, by presence of "Modify password for record type Users 'testuser'" with outcome "failure" in the log file.

To test the generation of Bluetooth audit records, the evaluator deployed the Bluetooth logging profile to the TOE and rebooted to enable verbose Bluetooth logging. Then, the evaluator verified the following auditable events:

- A Linux system was connected to the TOE, but the evaluator rejected the PIN code displayed on the TOE. This caused the pairing to fail. The bluetoothd log files contain the text "Pairing has completed with result 705 on device [Linux system BT address]" and "[Linux system BT address] disconnected with reason STATUS 719".
- A Linux system was connected to the TOE, and the evaluator accepted the PIN code displayed on the TOE. This caused the pairing to succeed. The bluetoothd log files contain the text "Pairing has completed with result 0 on device [Linux system BT address]".

FAU_GEN.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-FAU_GEN.1.2-AGD-01

The evaluator 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 *Audit Logs* of [CCGUIDE] lists a group of sample audit records in the XML format. Sample audit records are provided for each type of auditable events listed in Section 6 *Audit Logs* of [CCGUIDE].

The audit record in the XML format follows the following structure:

```
<record>
  <argument/>
  <path/>
  <attribute/>
  <subject/>
  <text/>
  <return/>
  <identity/>
</record>
```

Each field in the record is explained in Section 6 of [CCGUIDE].

Test Assurance Activities

Assurance Activity AA-FAU_GEN.1.2-ATE-01

The evaluator shall test the OS's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the ST. The evaluator 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

While evaluating FAU_GEN.1.1, the evaluator confirmed the generated audit records match the format specified in the [ST]. The records are stored in Basic Security Module (BSM) format, and always include the date and time, type of the event, as well as the outcome of the event. As demonstrated in FAU_GEN.1.1, specific types of events also include the subject identity (username) of the user affected by the event.

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

Section 7.2.1.2 FAU_GEN.1/BT *Audit data generation (Bluetooth)* in the TSS of [ST] lists the Bluetooth auditable events for which the TOE generates an audit record:

- Start-up and shutdown of the audit functions;
- Auditable events as shown in the following table.

Table 3: Auditable events (Bluetooth)

Requirement	Auditable Events	Additional Audit Record Contents
FIA_BLT_EXT.1	Failed user authorization of Bluetooth device.	User authorization decision (e.g., user rejected connection, incorrect pin entry).
	Failed user authorization for local Bluetooth Service.	Complete BD_ADDR. Bluetooth profile. Identity of local service with service ID.
FIA_BLT_EXT.2	Initiation of Bluetooth connection.	Complete BD_ADDR.
	Failure of Bluetooth connection.	Reason for failure.

The audit records contains at least the following information:

- Date and time of the event;
- Type of event;
- Subject identity;
- The outcome (success or failure) of the event;
- Additional audit record contents as shown in Table 3.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.2 Cryptographic support (FCS)

2.2.2.1 Cryptographic Key Generation (Refined) (FCS_CKM.1)

TSS Assurance Activities

Assurance Activity AA-FCS_CKM.1-ASE-01

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

Summary

Section 7.2.2 *Cryptography* in the TSS of [ST] summarizes the TOE's security features based on cryptography, lists the cryptographic modules implemented in the TOE, and presents the information of the cryptographic algorithms employed by the TOE in a table (Table 15 *Cryptographic algorithm table*). That table is reproduced below, serving as a reference.

Table 4: Cryptographic algorithm table

SFR	Algorithm	Capabilities	Usage
FCS_CKM.1	RSA KeyGen	2048, 3072, 4096	Client authentication in TLS client with mutual authentication
	ECDSA KeyGen	P-256, P-384, P-521	Bluetooth SSP (P-256) Client authentication in TLS client with mutual authentication and key establishment using ephemeral keys in TLS client (ECDHE)
FCS_CKM.2	RSA RSAES-PKCS1-v1_5 Key Establishment	2048, 3072, 4096	Key establishment in TLS client
	ECC Key Establishment (KAS-ECC)	P-256, P-384, P-521	Bluetooth SSP (P-256) Key establishment in TLS client
FCS_COP.1(1)	AES-CBC, AES-GCM	128-bit, 256-bit	Keychains (AES-GCM-256) TLS client (AES-CBC, AES-GCM)
	AES-CCM	128-bit	Bluetooth SSP (AES-CCM-128)
FCS_COP.1(2)	SHA-1, SHA-256, SHA-384, SHA-512		Secure boot (SHA-256 in Apple silicon and SHA-512 in "Intel with T2") TLS client Trusted update (SHA-256 in Apple silicon and SHA-512 in "Intel with T2")
FCS_COP.1(3)	RSA SigVer	2048, 3072, 4096 with: SHA-1, SHA-256, SHA-384, SHA-512	Secure boot (4096 bits with SHA-256 in "Intel with T2") TLS client Trusted update (4096 bits with SHA-256 in "Intel with T2")
	ECDSA SigVer	P-256, P-384, P-521 with: SHA-1, SHA-256, SHA-384, SHA-512	Secure boot (P-512 with SHA-512 in Apple silicon) TLS client Trusted update (P-512 with SHA-512 in Apple silicon)
FCS_COP.1(4)	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384,		Bluetooth BR/EDR (HMAC-SHA-256) TLS client

SFR	Algorithm	Capabilities	Usage
	HMAC-SHA-512		
FCS_RBG_EXT.1	CTR_DRBG	AES	Bluetooth TLS client

The TOE supports two cryptographic algorithms, namely, RSA KeyGen and ECDSA KeyGen, to meet FCS_CKM.1 security requirement. Section 7.2.2.1 *FCS_CKM.1 Cryptographic key generation* in the TSS of [ST] identifies the key sizes and usage of each of those two cryptographic algorithms. That information is shown in the following table.

Table 5: Cryptographic algorithms for FCS_CKM.1 Cryptographic key generation

SFR	Algorithm	Capabilities	Usage
FCS_CKM.1	RSA KeyGen	2048, 3072, 4096	Client authentication in TLS client with mutual authentication
	ECDSA KeyGen	P-256, P-384, P-521	Bluetooth SSP (P-256) Client authentication in TLS client with mutual authentication and key establishment using ephemeral keys in TLS client (ECDHE)

Guidance Assurance Activities

Assurance Activity AA-FCS_CKM.1-AGD-01

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

Summary

Per the TSS (TOE Summary Specification) for FCS_CKM.1 in the [ST], the TOE supports two cryptographic key generation algorithms, namely, RSA KeyGen and ECDSA KeyGen/KeyVer. Summarized in the following table is the information of the key sizes and uses of those two cryptographic algorithms.

Table 6: Cryptographic algorithms for FCS_CKM.1 Cryptographic key generation

SFR	Algorithm	Capabilities	Usage
FCS_CKM.1	RSA KeyGen	2048, 3072, 4096	TLS client mutual authentication
	ECDSA KeyGen/KeyVer	P-256, P-384, P-521	Bluetooth SSP (P-256) TLS client key establishment and mutual authentication

As shown in the above table, RSA and ECC are used as the key generation schemes for TLS communication. The algorithm and key size are selected automatically during the negotiation of TLS session establishment. Therefore, as mentioned in Section 4 *Secure Communications* of [CCGUIDE], no additional configuration is required for proper usage of TLS.

It's specified in the TSS section 7.2.2.1 of [ST] that Bluetooth SSP uses ephemeral ECDH curve P-256 for key establishment. No other cryptographic algorithms or key sizes are available. Therefore, the evaluator determined that no additional configuration is required for the key generation in Bluetooth SSP.

Test Assurance Activities

Assurance Activity AA-FCS_CKM.1-ATE-01

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 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 p_1, p_2, q_1, q_2, p and q shall all be provable primes.
 - Primes p_1, p_2, q_1 , and q_2 shall be provable primes and p and q shall be probable primes.
 - Primes p_1, p_2, q_1, q_2, 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,
- $|p-q| > 2^{nlen/2 - 100}$,
- $p \geq 2^{nlen/2 - 1/2}$,
- $q \geq 2^{nlen/2 - 1/2}$,
- $2^{(nlen/2)} < d < LCM(p-1, q-1)$,
- $e \cdot d = 1 \pmod{LCM(p-1, q-1)}$.

Summary

The evaluator confirmed 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 key generation algorithm, and the certificate information is provided in Table 1, entry FCS_CKM.1 in the AAR report and Table 17, entry FCS_CKM.1 in the [ST].

Assurance Activity AA-FCS_CKM.1-ATE-02

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Summary

The evaluator confirmed that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the ECDSA key generation algorithm, and the certificate information is provided in Table 1, entry FCS_CKM.1 in the AAR report and Table 17, entry FCS_CKM.1 in the [ST].

Assurance Activity AA-FCS_CKM.1-ATE-03

Key Generation for Finite-Field Cryptography (FFC)

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

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p :

- Cryptographic and Field Primes:
 - Primes q and p shall both be provable primes
 - Primes q and field prime p shall both be probable primes
- and two ways to generate the cryptographic group generator g :
- Cryptographic Group Generator:
 - Generator g constructed through a verifiable process
 - Generator g constructed through an unverifiable process

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

- Private Key:
 - $\text{len}(q)$ bit output of RBG where $1 \leq x \leq q-1$
 - $\text{len}(q) + 64$ bit output of RBG, followed by a mod $q-1$ operation where $1 \leq x \leq q-1$

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator will have the TSF generate 25 parameter sets and 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. Verification must also confirm:

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

for each FFC parameter set and key pair.

Summary

This SFR is not applicable, as the [ST] does not claim Finite-Field Cryptography (FFC).

Assurance Activity AA-FCS_CKM.1-ATE-04

Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups

[TD0501] Testing for FFC Schemes using Diffie-Hellman group 14 and/or "safe-prime" groups is done as part of testing in FCS_CKM.2.1

Summary

This SFR is not applicable, as the [ST] does not claim Finite-Field Cryptography (FFC).

2.2.2.2 Cryptographic Key Establishment (Refined) (FCS_CKM.2)

TSS Assurance Activities

Assurance Activity AA-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

The evaluator checked and verified that the two key establishment schemes specified in FCS_CKM.2 correspond to the two key generation schemes identified in FCS_CKM.1. The two key establishment schemes are RSA-based (RSA RSAES-PKCS1-v1_5) and ECC-based (KAS-ECC), and the two corresponding key generation schemes are RSA (RSA KeyGen) and ECC (ECDSA KeyGen).

In addition, Section 7.2.2.2 *FCS_CKM.2 Cryptographic key establishment* in the TSS of [ST] identifies the key sizes and usage of each of those two key establishment schemes. That information is shown in the following table.

Table 7: Cryptographic algorithms for FCS_CKM.2 Cryptographic key establishment

SFR	Algorithm	Capabilities	Usage
FCS_CKM.2	RSA RSAES-PKCS1-v1_5 Key Establishment	2048, 3072, 4096	Key establishment in TLS client
	ECC Key Establishment (KAS-ECC)	P-256, P-384, P-521	Bluetooth SSP (P-256) Key establishment in TLS client

Assurance Activity AA-FCS_CKM.2-ASE-02

SP800-56B Key Establishment Schemes

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

The evaluator will ensure that the TSS describes how the OS handles decryption errors. In accordance with NIST Special Publication 800-56B, the OS must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations.

Summary

As indicated in Section 7.2.2.2 *FCS_CKM.2 Cryptographic key establishment* of [ST], the TOE acts as the sender when using RSA-based key establishment in TLS sessions. When detecting a decryption error, the TOE will warn the invoker that an error has occurred, but it won't provide any cryptographically sensitive data to the invoker or in log files to unauthorized users.

Guidance Assurance Activities

Assurance Activity AA-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

Per the TSS for FCS_CKM.2 in the [ST], the TOE supports two cryptographic key establishment algorithms, namely, RSA RSAES-PKCS1-v1_5 and KAS-ECC. Summarized in the following table is the information of the key sizes and uses of those two cryptographic algorithms.

Table 8: Cryptographic algorithms for FCS_CKM.2 Cryptographic key establishment

SFR	Algorithm	Capabilities	Usage
FCS_CKM.2	RSA RSAES-PKCS1-v1_5 Key Establishment	2048, 3072, 4096	TLS client key establishment
	ECC Key Establishment (KAS-ECC)	P-256, P-384, P-521	Bluetooth SSP (P-256) TLS client key establishment

As shown in the above table, both RSA-based and ECC-based key establishment schemes are used for TLS communication. The algorithm and key size are selected automatically during the negotiation of TLS session establishment. Therefore, as mentioned in Section 4 *Secure Communications* of [CCGUIDE], no additional configuration is required for proper usage of TLS.

It's specified in TSS section 7.2.2.1 of [ST] that Bluetooth SSP uses ephemeral ECDH curve P-256 for key establishment. No other cryptographic algorithms or key sizes are available. Therefore, the evaluator determined that no additional configuration is required for Bluetooth SSP.

Test Assurance Activities

Assurance Activity AA-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 shall generate 10 sets of test vectors. The data set consists of the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

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

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

The evaluator will verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the OS shall perform the above for each implemented approved MAC algorithm.

Validity Test

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

The evaluator will inject an error in some of the test vectors to test that the OS recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the other information field OI, the data to be MAC'd, or the generated MAC tag. If the OS contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the OS's static private key to assure the OS detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The OS shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator will compare the OS's results with the results using a known good implementation verifying that the OS detects these errors.

Summary

The evaluator confirmed 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 Table 1, entry FCS_CKM.1 in the AAR report and Table 17, entry FCS_CKM.1 in the [ST] [📄](#).

Assurance Activity AA-FCS_CKM.2-ATE-02

RSAES-PKCS1-v1_5 Key Establishment Schemes

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

Summary

The RSA key establishment algorithm is vendor affirmed per Policy Letter #5 Addendum #2 v2.0.

Assurance Activity AA-FCS_CKM.2-ATE-03

Diffie-Hellman Group 14

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

Summary

This SFR is not applicable, as the [ST] [📄](#) does not claim Finite-Field Cryptography (FFC).

Assurance Activity AA-FCS_CKM.2-ATE-04

FFC Schemes using "safe-prime" groups (identified in Appendix D of SP 800-56A Revision 3)

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

Summary

This SFR is not applicable, as the [ST] [📄](#) does not claim Finite-Field Cryptography (FFC).

2.2.2.3 Cryptographic Key Destruction (FCS_CKM_EXT.4)

TSS Assurance Activities

Assurance Activity AA-FCS_CKM_EXT.4-ASE-01

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 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.

[TD0365] If the selection **“destruction of all key encrypting keys protecting target key according to FCS_CKM_EXT.4.1, where none of the KEKs protecting the target key are derived”** is included the evaluator shall 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 shall 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 shall 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 7.2.2.3 *FCS_CKM_EXT.4 Cryptographic key destruction* in the TSS of [ST] describes how the keys are managed in volatile and non-volatile memory.

The TOE offers a repository, called Keychain, to securely store user's important information, such as passwords, encryption keys, and so on. The users can use the Keychain Access app to manage the information stored in the keychain.

The keychain items are encrypted with DEKs (Data Encryption Keys). Each DEK is wrapped with one class key acting as KEK (Key Encryption Key) by the Secure Enclave, which is a dedicated security subsystem in Apple hardware platforms supporting the TOE. The KEKs remain inside the Secure Enclave, wrapped by a key derived from the user's passcode.

When a key is needed for cryptographic operations, the key is unwrapped and solely held in volatile memory in plaintext form for the duration that key is required. The key held in volatile memory is destroyed by an overwrite of zeros.

All DEKs and KEKs stored in non-volatile memory are always in wrapped form. The wrapped key in non-volatile memory is cryptographically destroyed by destroying the KEK wrapping that key. Since KEKs are maintained inside the Secure Enclave, a KEK is not able to be recovered after its destruction.

Section 7.2.2.3 references [CCGUIDE] which provide Keychain Services API documentation. Through the Keychain Services API, applications can manage the information stored in the Keychain.

The evaluator confirmed that the assignment operation is not performed on this SFR in the [ST]. In other words, the ST does not makes use of the open assignment on this SFR. The evaluator verified that the TSS of [ST] does not identify any configurations or circumstances that may not strictly conform to the key destruction requirements specified in this SFR.

Guidance Assurance Activities

Assurance Activity AA-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

Per the definition of FCS_CKM_EXT.4 in the [ST], the TOE employs the following cryptographic key destruction methods:

- The key held in volatile memory is destroyed by an overwrite of zeros;
- The key stored in non-volatile memory, which is always in wrapped form, is cryptographically destroyed by destroying the KEK (Key Encrypting Key) wrapping that key.

According to Section 5 *Storage of Credentials* of [CCGUIDE], there are no instances where key destruction may be delayed since the keys stored in the keychain are protected by the SEP that is part of the hardware. Thus no guidance is necessary.

Regarding the key destruction in non-volatile memory, Section 5 *Storage of Credentials* of [CCGUIDE] clarifies that the KEKs are maintained by the Secure Enclave Processor (SEP). Section 5 of [CCGUIDE] states that there are no instances that KEK destruction may be delayed since the SEP is the specialized security hardware integrated into Apple hardware platforms supporting the TOE.

Test Assurance Activities

Assurance Activity AA-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 held 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.

[TD0365] 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.

- **Test 3:** The following tests apply only to selection a), since the TOE in this instance has more visibility into what is happening within the underlying platform (e.g., a logical view of the media). In selection b), the TOE has no visibility into the inner workings and completely relies on the underlying platform, so there is no reason to test the TOE beyond test 2.
For selection a), the following tests are used to determine the TOE is able to request the platform to overwrite the key with a TOE supplied pattern.
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 specialized debugger to output the (hashed) values of the keys stored in the TOE. Then, the evaluator performed a factory reset on the TOE. After the factory reset, the evaluator used the same debugger to output the values of the new keys. The hashed values were different, implying that the keys had changed. This proves that the cryptographic keys were zeroized and overwritten upon factory reset.

2.2.2.4 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 7.2.2.4 FCS_CKM_EXT.8 Bluetooth key generation in the TSS of [ST] specifies the criteria for new key pair generation:

" The TOE generates a new ECDH key pair for every new Bluetooth connection attempt. "

The TSS also states that static ECDH key pairs are not permitted.

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 deployed the Bluetooth logging profile to the TOE and rebooted to enable verbose Bluetooth logging. Then, the evaluator paired the TOE to a Bluetooth mouse. The pairing was successful, and the public ECDH key sent by the TOE was recorded. The evaluator then unpaired the mouse from the TOE, and initiated the pairing once again. Again, the pairing was successful and the public ECDH key was recorded. Finally, the evaluator compared the two public keys and found that they differed.

2.2.2.5 Cryptographic Operation - Encryption/Decryption (Refined) (FCS_COP.1(1))

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

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

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

Summary

As mention in the TSS section 7.2.2.5 of [ST], the TOE supports AES encryption to meet FCS_COP.1(1) security requirement. Summarized in the following table is the information of the key sizes,modes, and uses of AES algorithm.

Table 9: Cryptographic algorithms for FCS_COP.1(1) Cryptographic operation - encryption/decryption

SFR	Algorithm	Capabilities	Usage
FCS_COP.1(1)	AES-CBC, AES-GCM	128-bit, 256-bit	Keychains (AES-GCM-256) TLS client (AES-CBC, AES-GCM)
	AES-CCM	128-bit	Bluetooth SSP (AES-CCM-128)

As shown in the above table, the TOE employs AES, in CBC and GCM modes, with 128-bit and 256-bits key, for TLS communication. The mode and key size are selected automatically during the negotiation of TLS session establishment. Therefore, as mentioned in Section 4 *Secure Communications* of [CCGUIDE], no additional configuration is required for proper usage of TLS.

As for Keychains and Bluetooth, the TOE uses a single AES mode and key size combination, respectively:

- Keychains: AES-GCM-256 (AES in GCM mode with a 256-bit key);

- Bluetooth: AES-CCM-128 (AES in CCM mode with a 128-bit key).

Therefore, the evaluator determined that no additional configuration is required for Keychains and Bluetooth.

Test Assurance Activities

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

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 shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- KAT-1. To test the encrypt functionality of AES-CBC, the evaluator will supply a set of 10 plaintext values and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all-zeros key. To test the decrypt functionality of AES-CBC, the evaluator will perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.
- KAT-2. To test the encrypt functionality of AES-CBC, the evaluator will supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality of AES-CBC, the evaluator will perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.
- KAT-3. To test the encrypt functionality of AES-CBC, the evaluator will supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost $N-i$ bits be zeros, for i in $[1, N]$. To test the decrypt functionality of AES-CBC, the evaluator will supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key i in each set shall have the leftmost i bits be ones and the rightmost $N-i$ bits be zeros, for i in $[1, N]$. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.
- KAT-4. To test the encrypt functionality of AES-CBC, the evaluator will supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost $128-i$ bits be zeros, for i in $[1, 128]$.

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

AES-CBC Multi-Block Message Test

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

AES-CBC Monte Carlo Tests

The evaluator will test the encrypt functionality using a set of 200 plaintext, IV, and key 3-tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

Input: PT, IV, Key

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

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

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

Summary

The evaluator confirmed 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 AES-CBC algorithm, and the certificate information is provided in Table 1, entry FCS_COP.1(1) in the AAR report and Table 17, entry FCS_COP.1(1) in the [STI].

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

AES-GCM Monte Carlo Tests

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

- 128 bit and 256 bit keys
- Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
- Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator will test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator will test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-GCM Test

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

- 128 bit and 256 bit keys
- Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.
- Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.
- Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator will test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator will test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator will compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

Summary

The evaluator confirmed 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 AES-GCM algorithm, and the certificate information is provided in Table 1, entry FCS_COP.1(1) in the AAR report and Table 17, entry FCS_COP.1(1) in the [ST] [\[ST\]](#).

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

AES-CCM Tests

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

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

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

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

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

To test the decryption-verification functionality of AES-CCM, for EACH combination of supported associated data length, payload length, nonce length and tag length, the evaluator shall supply a key value and 15 nonce, associated data and ciphertext 3-tuples and obtain either a FAIL result or a PASS result with the decrypted payload. The evaluator will supply 10 tuples that should FAIL and 5 that should PASS per set of 15.

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

Summary

The evaluator confirmed 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 AES-CCM algorithm, and the certificate information is provided in Table 1, entry FCS_COP.1(1) in the AAR report and Table 17, entry FCS_COP.1(1) in the [ST] [\[ST\]](#).

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

XTS-AES Test

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

- 256 bit (for AES-128) and 512 bit (for AES-256) keys
- Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a nonzero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

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

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

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

Summary

This SFR is not applicable, as the [ST] does not claim the XTS-AES algorithm.

Assurance Activity AA-FCS_COP.1-1-ATE-05**AES Key Wrap (AES-KW) and Key Wrap with Padding (AES-KWP) Test**

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

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

using a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator will use the AES-KW authenticated-encryption function of a known good implementation.

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

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

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

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

Summary

This SFR is not applicable, as the [ST] does not claim the AES-KW or AES-KWP algorithms.

Assurance Activity AA-FCS_COP.1-1-ATE-06**AES-CTR Test****Test 1: Known Answer Tests (KATs)**

[TD0630] There are four Known Answer Tests (KATs) described below. For all KATs, the plaintext, initialization vector (IV), and ciphertext values shall be 128-bit blocks. The results from each test may either be obtained by the validator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

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

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

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

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

Test 2: Multi-Block Message Test

[TD0630] The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key, IV, and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality by decrypting an i-block message where 1 less-than i less-than-or-equal to 10. For each i the evaluator shall choose a key and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key using a known good implementation.

Test 3: Monte-Carlo Test

[TD0630] For AES-CTR mode perform the Monte Carlo Test for ECB Mode on the encryption engine of the counter mode implementation. There is no need to test the decryption engine.

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

For AES-ECB mode

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

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

If "invoke platform-provided" is selected, the evaluator confirms that SSH connections are only successful if appropriate algorithms and appropriate key sizes are configured. To do this, the evaluator shall perform the following tests:

Test 1: [Conditional: TOE is an SSH server] The evaluator shall configure an SSH client to connect with an invalid cryptographic algorithm and key size for each listening SSH socket connection on the TOE. The evaluator initiates SSH client connections to each listening SSH socket connection on the TOE and observes that the connection fails in each attempt.

Test 2: [Conditional: TOE is an SSH client] The evaluator shall configure a listening SSH socket on a remote SSH server that accepts only invalid cryptographic algorithms and keys. The evaluator uses the TOE to attempt an SSH connection to this server and observes that the connection fails.

Summary

This SFR is not applicable, as the [ST] does not claim the AES-CTR algorithm.

2.2.2.6 Cryptographic Operation - Hashing (Refined) (FCS_COP.1(2))

TSS Assurance Activities

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

The evaluator will check that the association of the hash function with other application cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Summary

Section 7.2.2.6 FCS_COP.1(2) Cryptographic operation - hashing in the TSS of [ST] lists the hash algorithms supported by the TOE:

- SHA-1,
- SHA-256,
- SHA-384,
- SHA-512.

Those hashing algorithms listed above are employed by the TOE for signature services and HMAC services, which are documented in the following two sections respectively:

- Section 7.2.2.7 FCS_COP.1(3) Cryptographic operation - signing;
- Section 7.2.2.8 FCS_COP.1(4) Cryptographic operation - keyed-hash message authentication.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

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

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

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

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

- **Test 3: Selected Long Messages Test (Bit oriented Mode)** - The evaluator will generate an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the i th message is $512 + 99 \cdot i$, where $1 \leq i \leq m$. The message text shall be pseudorandomly generated. The evaluator will compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.
- **Test 4: Selected Long Messages Test (Byte oriented Mode)** - The evaluator will generate an input set consisting of $m/8$ messages, where m is the block length of the hash algorithm. The length of the i th message is $512 + 8 \cdot 99 \cdot i$, where $1 \leq i \leq m/8$. The message text shall be pseudorandomly generated. The evaluator will compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.
- **Test 5: Pseudorandomly Generated Messages Test** - This test is for byte-oriented implementations only. The evaluator will randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluator will then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluator will then ensure that the correct result is produced when the messages are provided to the TSF.

Summary

The evaluator confirmed 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 SHA-1, SHA-256, SHA-384 and SHA-512 algorithms, and the certificate information is provided in Table 1, entry FCS_COP.1(2) in the AAR report and Table 17, entry FCS_COP.1(2) in the [ST].

2.2.2.7 Cryptographic Operation - Signing (Refined) (FCS_COP.1(3))

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

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

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

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

ECDSA Algorithm Tests

- **Test 1: ECDSA FIPS 186-4 Signature Generation Test.** For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator will generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S . To determine correctness, the evaluator will use the signature verification function of a known good implementation.
- **Test 2: ECDSA FIPS 186-4 Signature Verification Test.** For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator will generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator will verify that 5 responses indicate success and 5 responses indicate failure.

Summary

Test 1: Not applicable. The [ST] does not claim ECDSA signature generation.

Test 2: the evaluator confirmed that Automated Cryptographic Validation Testing System (ACVTS) is used to test all cryptographic algorithms in accordance with the CAVP test procedures. The results have been validated by the CAVP for the ECDSA signature verification algorithm, and the certificate information is provided in Table 1, entry FCS_COP.1(3) in the AAR report and Table 17, entry FCS_COP.1(3) in the [ST].


Assurance Activity AA-FCS_COP.1-3-ATE-02

RSA Signature Algorithm Tests

- **Test 1: Signature Generation Test.** The evaluator will verify the implementation of RSA Signature Generation by the OS using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator will have the OS use its private key and modulus value to sign these messages. The evaluator will verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.
- **Test 2: Signature Verification Test.** The evaluator will perform the Signature Verification test to verify the ability of the OS to recognize another party's valid and invalid signatures. The evaluator will inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The evaluator will verify that the OS returns failure when validating each signature.

Summary

Test 1: Not applicable. The [ST]  does not claim RSA signature generation.

Test 2: the evaluator confirmed 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 signature verification algorithm, and the certificate information is provided in Table 1, entry FCS_COP.1(3) in the AAR report and Table 17, entry FCS_COP.1(3) in the [ST] .

2.2.2.8 Cryptographic Operation - Keyed-Hash Message Authentication (Refined) (FCS_COP.1(4))

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.


Test Assurance Activities

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

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

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

Summary

The evaluator confirmed 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 HMAC-SHA-1, HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512 algorithms, and the certificate information is provided in Table 1, entry FCS_COP.1(4) in the AAR report and Table 17, entry FCS_COP.1(4) in the [ST] .

2.2.2.9 Random Bit Generation (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_RBG_EXT.1.1-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.

Summary

The evaluator confirmed 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 CTR_DRBG (AES) algorithm, and the certificate information is provided in Table 1, entry FCS_RBG_EXT.1 in the AAR report and Table 17, entry FCS_RBG_EXT.1 in the [ST][a](#).

FCS_RBG_EXT.1.2

TSS Assurance Activities

Assurance Activity AA-FCS_RBG_EXT.1.2-ASE-01

Documentation shall be produced - and the evaluator will perform the activities - in accordance with Appendix E in [OSPPv4.2.1][a](#) and the Clarification to the Entropy Documentation and Assessment Annex in [CCEVS-EDAC][a](#).

In the future, specific statistical testing (in line with NIST SP800-90B) will be required to verify the entropy estimates.

Summary

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

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_RBG_EXT.1.2-ATE-01

Documentation shall be produced - and the evaluator will perform the activities - in accordance with Appendix E in [OSPPv4.2.1] and the Clarification to the Entropy Documentation and Assessment Annex in [CCEVS-EDAC].

In the future, specific statistical testing (in line with NIST SP800-90B) will be required to verify the entropy estimates.

Summary

The vendor provided an Entropy Assessment Report to the evaluator. The evaluator reviewed the report, and determined that the provided information is in line with Appendix E in OSPPv4.2.1 and the Clarification to the Entropy Documentation and Assessment Annex in CCEVS-EDAC. The evaluation activities were performed according to these guidelines.

2.2.2.10 Storage of Sensitive Data (FCS_STO_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FCS_STO_EXT.1-ASE-01

The evaluator will check the TSS to ensure that it lists all persistent sensitive data for which the OS provides a storage capability. For each of these items, the evaluator will confirm that the TSS lists for what purpose it can be used, and how it is stored. The evaluator will confirm that cryptographic operations used to protect the data occur as specified in FCS_COP.1(1).

The evaluator will also consult the developer documentation to verify that an interface exists for applications to securely store credentials.

Summary

Section 7.2.2.10 *FCS_STO_EXT.1 Storage of sensitive data* in the TSS of [ST] describes how sensitive data is encrypted and stored by the TOE in non-volatile storage. The following table shows each type of sensitive data, the purpose it can be used, and how it is stored.

Table 10: Storage of sensitive data

Sensitive data	Purpose	Storage
TOE Username	Authentication	Local directory services
Trusted Certificates	Establishing TLS sessions	Keychain
Private Keys	Establishing TLS sessions	Keychain

Section 7.2.2.10 references [CCGUIDE] which describes the handling of TOE usernames via the "Users & Group" GUI.

The TOE offers a repository, called Keychain, to securely store sensitive data. The evaluator checked and confirmed that it can be accessed through the Keychain Access app in the /System/Applications/Utilities/ folder. An initial default keychain is created for each user, though users can create other keychains for specific purposes.

In addition to user keychains, the TOE relies on a number of system-level keychains that maintain authentication assets. Keychain items are encrypted using AES in GCM mode with 256-bit keys. The evaluator checked and confirmed that AES-GCM-256 cryptographic operations is specified in FCS_COP.1(1). Section 7.2.2.10 *FCS_STO_EXT.1 Storage of sensitive data* references the Keychain Services API documentation in [CCGUIDE]. The evaluator checked that documentation and verified that an interface exists for applications to securely store credentials in the Keychain.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

No assurance activities defined.

2.2.2.11 TLS Client Protocol (FCS_TLSC_EXT.1)

FCS_TLSC_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.1-ASE-01

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

Summary

Section 7.2.2.11 *FCS_TLSC_EXT.1 TLS client protocol* in the TSS of [ST] lists TLS 1.2 cipher suites supported by the TOE, as show below:

- The TOE implements TLS 1.2 (RFC 5246) supporting the following cipher suites:
 - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
 - TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246
 - TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288
 - TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288
 - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
 - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
 - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- The TOE also supports the following TLS 1.2 cipher suites not specified in [OSPPv4.2.1]:
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

The evaluator verified that these cipher suites described in the TSS are consistent with those specified in FCS_TLSC_EXT.1.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.1-AGD-01

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

Summary

The TSS sections 7.2.2.11 - 7.2.2.13 of [ST] describes how the TOE implements TLS 1.2 (RFC 5246) client functionality:

- The TOE supports the following cipher suites:
 - TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246
 - TLS_RSA_WITH_AES_256_CBC_SHA as defined in RFC 5246,
 - TLS_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5288,
 - TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
 - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
 - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
 - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
 - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,
 - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA,
 - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA,
 - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,
 - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA.
- The TOE supports the Elliptic Curves Extensions secp256r1, secp384r1, and secp521r1.
- The TOE supports mutual authentication using X.509v3 certificates.

Section 4 *Secure Communications* of [CCGUIDE] provides related guidance for configuring TLS. Section 4 of [CCGUIDE] enumerates the following two scenarios where the TOE uses TLS as a TLS client:

- The TOE communicates with the Apple Update Server;
- Applications communicate with TLS servers specified by the applications.

Section 4 of [CCGUIDE] also states that applications invoke TLS using the URLSession class with the “https” protocol and applications perform TLS client authentication using an X.509 certificate by specifying the certificate in a URLCredential instance.

The evaluator determined that the cipher suite and key selection are determined automatically during the negotiation of TLS session establishment. Therefore, as mentioned in Section 4 *Secure Communications* of [CCGUIDE], no additional configuration is required for proper usage of TLS.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.1-ATE-01

The evaluator will also perform the following tests:

- **Test 1:** The evaluator will establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
- **Test 2:** The evaluator will attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage field and verify that a connection is not established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.
- **Test 3:** The evaluator will send a server certificate in the TLS connection that does not match the server-selected cipher suite (for example, send a ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite or send a RSA certificate while using one of the ECDSA cipher suites.) The evaluator will verify that the OS disconnects after receiving the server's Certificate handshake message.

- **Test 4:** The evaluator will configure the server to select the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the client denies the connection.
- **Test 5:** The evaluator will perform the following modifications to the traffic:
 - **Test 5.1:** Change the TLS version selected by the server in the Server Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.
 - **Test 5.2** Modify at least one byte in the server's nonce in the Server Hello handshake message, and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE cipher suite) or that the server denies the client's Finished handshake message.
 - **Test 5.3:** Modify the server's selected cipher suite in the Server Hello handshake message to be a cipher suite not presented in the Client Hello handshake message. The evaluator will verify that the client rejects the connection after receiving the Server Hello
 - **Test 5.4:** If an ECDHE or DHE ciphersuite is selected, modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message.
 - **Test 5.5:** Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.
 - **Test 5.6:** Send a garbled message from the Server after the Server has issued the Change Cipher Spec message and verify that the client denies the connection.

Summary

The nscurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

Test 1 - The evaluator verified that a TLS connection was successfully established for each of the following ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_GCM_SHA256
- TLS_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

Test 2 - The evaluator created a server certificate with "Server Authentication" in the extendedKeyUsage(EKU) field and verified that connection was successfully established. The evaluator then created a second, otherwise identical certificate without the "Server Authentication" option and verified that the connection was rejected.

Test 3 - The evaluator configured HTTPS to expect TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384. The evaluator then makes use of a modified version of OpenSSL in which the cipher definition TLS_RSA_WITH_AES_256_GCM_SHA384 is usable and modified to return the identifier for TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384. The TLS server is also configured to use an RSA certificate.

The evaluator then confirms that the TLS connection to server with RSA key but ECDSA cipher suite is correctly rejected.

Test 4 - The evaluator modified the OpenSSL cipher of AES128_SHA to be instead the TLS_NULL_WITH_NULL_NULL cipher. The evaluator then verified that the connection to the TLS server with TLS NULL cipher is rejected.

Test 5 -

Test 5.1 - The evaluator modified the TLS version selected by the server, and verified that the connection to server with too low TLS version was rejected as expected.

Test 5.2 - The evaluator modified the nonce of the server, and verified that the connection to server with modified server nonce was rejected as expected.

Test 5.3 - The evaluator modified the selected cipher suite of the server, and verified that the connection to server with modified server cipher suite was rejected as expected.

Test 5.4 - The evaluator modified the signature block in the server's key exchange handshake message, and verified that the connection to server with modified signature block was rejected as expected.

Test 5.5 - The evaluator modified the server finished message, and verified that the connection to server with modified server finished was rejected as expected.

Test 5.6 - The evaluator configured OpenSSL to send random data after the server issues the "Change Cipher Spec" message, and verified that the connection to server with random data was rejected as expected.

FCS_TLSC_EXT.1.2

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.2-ASE-01

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

Summary

Section 7.2.2.11 *FCS_TLSC_EXT.1 TLS client protocol* in the TSS of [ST][\[d\]](#) describes the client's method of establishing reference identifiers as below:

" The TOE verifies that server certificate is valid according to FIA_X509_EXT.1 and that the presented identifier matches the reference identifier according to RFC 6125. The reference identifiers supported are DNS and IP addresses in the SAN. Wildcards are supported. The TOE does not support certificate pinning. "

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.2-AGD-01

The evaluator will verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Summary

Section 4 *Secure Communications* of [CCGUIDE][\[d\]](#) states that the TOE automatically generates reference identifiers which match the SAN. Applications on the TOE invoke TLS using the URLSession class with the "https" protocol. The reference identifiers are automatically generated from the host portion of URLs. The TOE only generates reference identifiers that will match the SAN.

Therefore, as mentioned in Section 4 *Secure Communications* of [CCGUIDE][\[d\]](#), no additional configuration is required for proper usage of TLS.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.2-ATE-01

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

- **Test 1:** The evaluator will present a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. The evaluator will verify that the connection fails.
- **Test 2:** The evaluator will present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator will repeat this test for each supported SAN type.
- **Test 3 [conditional]:** If the TOE does not mandate the presence of the SAN extension, the evaluator will present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator will verify that the connection succeeds. If the TOE does mandate the presence of the SAN extension, this test shall be omitted.
- **Test 4:** The evaluator will present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator will verify that the connection succeeds.
- **Test 5:** The evaluator will perform the following wildcard tests with each supported type of reference identifier:
 - **Test 5.1:** The evaluator will present a server certificate containing a wildcard that is not in the left-most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.
 - **Test 5.2:** The evaluator will present a server certificate containing a wildcard in the left-most label but not preceding the public suffix (e.g. *.example.com). The evaluator will configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds. The evaluator will configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator will configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails.
 - **Test 5.3:** The evaluator will present a server certificate containing a wildcard in the left-most label immediately preceding the public suffix (e.g. *.com). The evaluator will configure the reference identifier with a single left-most label (e.g. foo.com) and verify that the connection fails. The evaluator will configure the reference identifier with two left-most labels (e.g. bar.foo.com) and verify that the connection fails.
- **Test 6:** [conditional] If URI or Service name reference identifiers are supported, the evaluator will configure the DNS name and the service identifier. The evaluator will present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator will repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.
- **Test 7:** [conditional] If pinned certificates are supported the evaluator will present a certificate that does not match the pinned certificate and verify that the connection fails.

Summary

The nscurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

Test 1 - The evaluator created a server certificate in which no identifier is present for Subject Alternative Name(SAN), and a Common Name (CN) does not match the reference identifier. The evaluator was able to verify that the connection to server with "wrong CN, no SAN" was rejected as expected.

Test 2 - The evaluator created a server certificate in which CN matches the reference identifier, but contains no identifier in the SAN that matches the reference identifier. The supported SAN types included are DNS and URI.

The evaluator was able to verify that the connection to server with right CN, wrong DNS-SAN was rejected as expected.

The evaluator was able to verify that the connection to server with right CN, wrong URI-SAN was rejected as expected.

Test 3 - Not Applicable since the TOE mandates the presence of a SAN and does not process the CN.

Test 4 - The evaluator created a server certificate in which a Common Name (CN) does not match the reference identifier, but does contain an identifier in the SAN (DNS-SAN for the purposes of this test) that does match. The evaluator was able to verify that the connection to server with wrong CN, right DNS-SAN was established successfully.

Test 5 -

Test 5.1 - The evaluator created a certificate with a wildcard not in the left-most label of the presented identifier. More specifically, the evaluator attempted to make a TLS connection to server.subdomain.domain.com with server.*.domain.com DNS name, and verified that this connection was rejected as expected.

Test 5.2 - The evaluator created a certificate with a wildcard in the left-most label of the presented identifier (*.domain.com). More specifically, the evaluator attempted to make a TLS connection to server.domain.com with *.domain.com DNS name, and verified that this connection was established successfully.

Using the same certificate, the evaluator attempted to make a TLS connection to domain.com with *.domain.com DNS name, and verified that this connection was rejected as expected.

Using the same certificate, the evaluator attempted to make a TLS connection to server.subdomain.domain.com with *.domain.com DNS name, and verified that this connection was rejected as expected.

Test 5.3 - The evaluator created a certificate with a wildcard in the left-most label of the presented identifier, immediately preceding the public suffix (*.com). More specifically, the evaluator attempted to make a TLS connection to server.domain.com with *.com DNS name, and verified that this connection was rejected as expected.

Test 6 - The evaluator created a server certificate with a configured DNS name and service identifier. The evaluator was able to verify that the connection to server with right service identifier and right DNS name was established successfully.

Using the same server certificated, the evaluator was able to verify that the connection to server with wrong service identifier and right DNS name was rejected as expected.

Test 7 - Not Applicable since certificate pinning is not supported by the TOE.

FCS_TLSC_EXT.1.3

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.1.3-ATE-01

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

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

Summary

The nscurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

Test 1 - The evaluator created a server certificate without a valid certification path. The evaluator was able to verify that the connection to server with invalid certificate path was rejected as expected.

The evaluator then validated this certificate using a trusted CA cert, and was able to verify that the connection to server with valid certificate was established successfully.

Finally, the evaluator deleted the trusted CA certificate, and was able to verify that the connection to server was rejected due to invalid certificate path.

Test 2 - The vendor provided a full CA, as well as a sub-CA signed by the vendor root-CA for this testing. Using these vendor-provided CAs, the evaluator configured a connection in which a connection attempt is made with a revoked certificate. The evaluator was able to verify that the connection to server was rejected due to revoked certificate.

Test 3 - The evaluator generated an already-expired certificate, and used this expired cert in an attempt to connect with the TLS server. The evaluator was able to verify that the connection to server was rejected due to expired certificate.

Test 4 - The evaluator created a server certificate with no valid identifier. The evaluator was able to verify that a peer attempting to connect with this certificate is rejected by the server due to invalid identifier.

2.2.2.12 TLS Client Protocol (FCS_TLSC_EXT.2)

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.2-ASE-01

The evaluator will verify that TSS describes support for the Supported Groups Extension and whether the required behavior is performed by default or may be configured.

Summary

Section 7.2.2.12 *FCS_TLSC_EXT.2 TLS client protocol* in the TSS of [ST] states that the TOE presents the Supported Elliptic Curves Extension in the Client Hello with the following NIST curves: secp256r1, secp384r1, and secp521r1. Those NIST curves are also specified in FCS_COP.1(3), FCS_CKM.1 and FCS_CKM.2. In addition, the required behavior is performed by default.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.2-AGD-01

If the TSS indicates that support for the Supported Groups Extension must be configured to meet the requirement, the evaluator will verify that AGD guidance includes configuration instructions for the Supported Groups Extension.

Summary

Per the definition of FCS_TLSC_EXT.2 in the [ST], the TOE supports the Elliptic Curves Extensions secp256r1, secp384r1, and secp521r1. The TSS for FCS_TLSC_EXT.2 (Section 7.2.2.12 of [ST]) provides the consistent information. The TSS also states that the TOE provides the support for Elliptic Curve Extensions by default.

Therefore, as mentioned in Section 4 *Secure Communications* of [CCGUIDE], no additional configuration is required for proper usage of TLS.

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.2-ATE-01

The evaluator will also perform the following test:

The evaluator will 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 nscurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

For each of the supported curves (P-256, P-384, and P-521), the evaluator verified that an ECDHE TLS connection was successfully established.

2.2.2.13 TLS Client Protocol (FCS_TLSC_EXT.4)

TSS Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.4-ASE-01

The evaluator will ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Summary

Section 7.2.2.13 FCS_TLSC_EXT.4 TLS client protocol in the TSS of [ST] states that the TOE supports mutual authentication using client-side X.509v3 certificates.

Guidance Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.4-AGD-01

The evaluator will 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

Per the definition of FIA_X509_EXT.2 in the [ST], the TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS and HTTPS connections.

Section 4 Secure Communications of [CCGUIDE] provides the following instructions for configuring the client-side certificates:

" Applications can perform TLS client authentication using an X.509 certificate by specifying the certificate in a URLCredential instance. "

Test Assurance Activities

Assurance Activity AA-FCS_TLSC_EXT.4-ATE-01

The evaluator will also perform the following test:

- **Test 1:** The evaluator will establish a connection to a peer 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 will establish a connection to a peer 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) messages.

Summary

The nscurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

Test 1: the evaluator configured the webserver to disable mutual (client certificate) authentication. Then, the evaluator verified that a TLS connection was successfully established.

Test 2: the evaluator configured the webserver to enable mutual (client certificate) authentication. Note: the nscurl utility currently does not support client certificates. Therefore, this test was performed using Safari. The evaluator verified that a TLS connection was successfully established and the server prompted the Safari client for a client certificate. This was further confirmed using the logged TLS packets.

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-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 7.2.3.1 *FDP_ACF_EXT.1 Access controls for protecting user data* in the TSS of [ST] presents the access control policy enforced by the TOE. The TOE enforces access control using four file system security schemes: sandbox entitlements, POSIX access control lists (ACLs), Unix (BSD) permissions, and per-file BSD flags that override Unix permissions. The TSS describes in detail the file system security schemes above.

The TSS includes a list of access control rules for determining the accesses to particular files and directories by particular users. The evaluator examined the access control rules and determined that the rules are complete and unambiguous.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FDP_ACF_EXT.1-ATE-01

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

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

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

Summary

The `sysadminctl -addUser` and `sysadminctl -deleteUser` commands are used to add and delete two standard user accounts. The `createhomedir` command is used to create a home directory for each user.

Test 1: as `testuser_1`, the evaluator created a file in its home directory. Then, as `testuser_2`, the evaluator attempted to read the file. The evaluator verified that the file could not be read.

Test 2: as `testuser_1`, the evaluator created a file in its home directory. Then, as `testuser_2`, the evaluator attempted to write to the file. The evaluator verified that the file could not be written to.

Test 3: as `testuser_1`, the evaluator created a file in its home directory. Then, as `testuser_2`, the evaluator attempted to remove the file. The evaluator verified that the file could not be removed.

Test 4: as `testuser_1`, the evaluator attempted to create a file the home directory of `testuser_2`. The evaluator verified that the file could not be created.

Test 5: as `testuser_1`, the evaluator created a file in its home directory. Then, as `testuser_1`, the evaluator attempted to write to the file. The evaluator verified that the file could be written to.

Test 6: as `testuser_1`, the evaluator created a file in its home directory. Then, as `testuser_1`, the evaluator attempted to remove the file. The evaluator verified that the file could be removed.

2.2.4 Identification and authentication (FIA)

2.2.4.1 Authentication failure handling (Refined) (FIA_AFL.1)

FIA_AFL.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_AFL.1.1-ATE-01

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

Summary

The evaluator first created a new temporary user ("testuser"). Then, the evaluator used the `pwpolicy` command to change the maximum number of authentication attempts for `testuser` to 3. Finally, the the evaluator attempted to login three times with an incorrect password to `testuser`. The evaluator observed that the account was permanently locked and the events ("Verify password for record type Users 'testuser'" with outcome "failure") were written to the audit log.

FIA_AFL.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_AFL.1.2-ATE-01

- **Test 1:** The evaluator will attempt to authenticate repeatedly to the system with a known bad password. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.
- **Test 2:** The evaluator will attempt to authenticate repeatedly to the system with a known bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.
- **Test 3:** The evaluator will attempt to authenticate repeatedly to the system using both a bad password and a bad certificate. Once the defined number of failed authentication attempts has been reached the evaluator will ensure that the account that was being used for testing has had the actions detailed in the assignment list above applied to it. The evaluator will ensure that an event has been logged to the security event log detailing that the account has had these actions applied.

Summary

Test 1: this test is covered by [FIA_AFL.1.1](#) above.

Test 2: not applicable. The [\[ST\]](#) does not claim certificate-based authentication.

Test 3: not applicable. The [\[ST\]](#) does not claim certificate-based authentication.

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 7.2.4.2 *FIA_BLT_EXT.1 Bluetooth user authorization* in the TSS of [\[ST\]](#) states that user permission is required for Bluetooth pairing. Bluetooth pairing can only occur when it is explicitly authorized through the System Settings » Bluetooth interface of the TOE device. During the pairing time, users need to manually enter a PIN or manually confirm that a provided PIN matches on both devices, depending on the remote Bluetooth device.

The same requirement also applies to the applications that use Bluetooth.

Guidance Assurance Activities

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

The evaluator shall examine the API documentation provided as a means of satisfying the requirements for the ADV assurance class (see section 5.2.2 in the MDF PP and GPOS PP) and verify that this API documentation does not include any API for programmatic entering of pairing information (e.g. PINs; numeric codes; or "yes/no" responses) intended to bypass manual user input during pairing.

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

The evaluator examined the provided API documentation for Bluetooth [COREBT] (Core Bluetooth framework) and verified that there is no means for bypassing manual user input during pairing.

Section 3.12.2 *Add/Pair* of [CCGUIDE] provides the instructions to authorize the pairing with a Bluetooth device. This involves turning on the Bluetooth feature on the TOE (System Settings > Bluetooth), put the Bluetooth device in discoverable mode, select the Bluetooth device from GUI on the TOE, and then enter a PIN if prompted.

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 use the NoInputNoOutput Bluetoothctl agent. Then, the Linux system was connected and paired to the TOE. The evaluator verified that the TOE required explicit user confirmation to continue. After permission was granted, the pairing completed successfully.

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 7.2.4.3 *FIA_BLT_EXT.2 Bluetooth mutual authentication* in the TSS of [ST] affirms that the data transfers between the TOE and a given Bluetooth device cannot complete until the Bluetooth device is paired and mutually authenticated with the TOE. This protection is fulfilled by the internal logic of the TOE's Bluetooth device driver program.

Section 7.2.4.3 also states that the TOE supports RFCOMM and L2CAP data transfer.

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 first connected and paired a Linux system to the TOE. Then, the evaluator removed the Linux system from the list of known devices on the TOE. This forced the TOE to consider the Linux system as an untrusted device. Finally, the evaluator attempted to send a file to the TOE using OBEX Push. The evaluator verified that the TOE required mutual authentication (PIN code confirmation) to continue. After permission was granted, the file transfer completed successfully.

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 7.2.4.4 *FIA_BLT_EXT.3 Rejection of duplicate Bluetooth connections* in the TSS of [ST] describes how the TOE handle duplicate Bluetooth connections. A duplicate connection attempt from the same BD_ADDR for an established connection will be discarded.

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 first connected and paired the TOE to a Bluetooth mouse. The Bluetooth MAC address of the TOE and the mouse were recorded for future use. Then, the evaluator used BlueZ to configure the Bluetooth MAC address of a specialized Bluetooth dongle to the mouse's Bluetooth

MAC address. Finally, the evaluator attempted to connect and pair using the Bluetooth dongle to the TOE. This failed (i.e. the TOE ignored the connection/pairing attempts), and the initial connection between the TOE and the Bluetooth mouse was unaffected.

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 7.2.4.5 FIA_BLT_EXT.4 Secure Simple Pairing in the TSS of [ST] describes the Secure Simple Pairing (SSP) process. Bluetooth devices are required to use SSP to pair with the TOE. SSP uses ECDH for authentication and key exchange, in addition to AES for data encryption.

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 connected and paired a Linux system with Secure Simple Pairing (SSP) support to the TOE. Using hcidump, the evaluator verified that SSP was indeed advertised to the TOE. The pairing was successful, and SSP was used to establish an encrypted connection.

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 7.2.4.6 FIA_BLT_EXT.6 Trusted Bluetooth device user authorization in the TSS of [ST] serves as TSS for both FIA_BLT_EXT.6 and FIA_BLT_EXT.7. The Bluetooth profiles supported by the TOE are listed in this section. According to the SFRs of FIA_BLT_EXT.6 and FIA_BLT_EXT.7, the TOE does not support any Bluetooth profiles for which explicit user authorization is required before a remote device can gain access to the associated services.

The TOE establishes a "trusted relationship" with an authorized device during the pairing process. The TOE automatically authorizes the remote Bluetooth device for all Bluetooth profiles the remote device announces to support. The difference in behavior between a trusted device and an untrusted device is explained:

" The only difference in behavior between a trusted device and an untrusted device is that the untrusted device must first be manually authorized [...]"

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

Note, the TOE authorizes a device upon successful pairing. When pairing was unsuccessful (i.e. either declined or something else happened), the remote Bluetooth device is not authorized. Per the [ST] [\[ST\]](#), all services are authorized automatically, no additional authorization is required.

Test 1: the evaluator attempted to connect and pair a Linux system to the TOE. The evaluator verified that mutual authentication (PIN code confirmation) was required to continue. After rejecting the PIN code, the pairing did not complete and the TOE disconnected.

Test 2: the evaluator attempted to connect and pair a Linux system to the TOE. The evaluator verified that mutual authentication (PIN code confirmation) was required to continue. After accepting the PIN code, the pairing completed successfully and a Bluetooth connection was established.

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 see [AA-BTPPM-FIA_BLT_EXT.6-ASE-01](#) for the provided information.

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

Note, the TOE authorizes a device upon successful pairing. When pairing was unsuccessful (i.e. either declined or something else happened), the remote Bluetooth device is not authorized. Per the [ST] [\[ST\]](#), all services are authorized automatically, no additional authorization is required.

Test 1: the evaluator attempted to connect and pair a Linux system to the TOE. The evaluator verified that mutual authentication (PIN code confirmation) was required to continue. After rejecting the PIN code, the pairing did not complete and the TOE disconnected.

Test 2: the evaluator attempted to connect and pair a Linux system to the TOE. The evaluator verified that mutual authentication (PIN code confirmation) was required to continue. After accepting the PIN code, the pairing completed successfully and a Bluetooth connection was established.

Test 3 through 5: Not applicable. Authorization is provided with successful pairing for all services defined by the Bluetooth device.

2.2.4.8 Multiple Authentication Mechanisms (Refined) (FIA_UAU.5)

FIA_UAU.5.1

TSS Assurance Activities

Assurance Activity AA-FIA_UAU.5.1-ASE-01

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

Summary

Section 7.2.4.8 FIA_UAU.5 Multiple authentication mechanisms in the TSS of [ST] [\[ST\]](#) states that the TOE supports authentication based on username and smart card.

There are two phases of a smart card authentication system:

Paring the smart card with the user account:

During the paring process to associate a smart card with a user account, the user enters the PIN to release the smart card's certificate (which contains its public key). Then the TOE stores that information for future use.

Authenticating the user to the TOE:

When a user inserts a smart card to authenticate, the user enters the associated PIN to unlock the smart card. Once unlocked, the smart card can perform a signing operation. Then the TOE verifies the signature using the paired certificate for authentication.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_UAU.5.1-ATE-01

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

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

Summary

Test 1: the evaluator logged in to a test user account using the correct password. The evaluator verified that authentication was successful.

Test 2: the evaluator logged in to a test user account using an incorrect password. The evaluator verified that authentication was not successful.

Assurance Activity AA-FIA_UAU.5.1-ATE-02

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

The evaluator will then conduct the following tests:

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

Summary

The A YubiKey USB security token, which stores an asymmetric key, is used as a smart card during testing. The YubiKey Manager was installed on the TOE to configure the PIN and pair the YubiKey for macOS account login.

Test 1: the evaluator rebooted the TOE with the YubiKey inserted. On boot, the OS requested a PIN instead of a password. The evaluator logged in using the correct PIN and verified that the authentication was successful.

Test 2: the evaluator rebooted the TOE with the YubiKey inserted. On boot, the OS requested a PIN instead of a password. The evaluator logged in using an incorrect PIN and verified that the authentication was not successful.

Assurance Activity AA-FIA_UAU.5.1-ATE-03

If **X.509 certificate authentication** is selected, the evaluator will generate an X.509v3 certificate for a user with the Client Authentication Enhanced Key Usage field set. The evaluator will provision the OS for authentication with the X.509v3 certificate. The evaluator will ensure that the certificates are validated by the OS as per FIA_x509_EXT.1.1 and then conduct the following tests:

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

Summary

This test is not applicable, as the [ST] [\[redacted\]](#) does not claim X.509 certificate authentication.

FIA_UAU.5.2

TSS Assurance Activities

Assurance Activity AA-FIA_UAU.5.2-ASE-01

The evaluator will ensure that the TSS describes each mechanism provided to support user authentication and the rules describing how the authentication mechanism(s) provide authentication.

Summary

Section 7.2.4.8 FIA_UAU.5 Multiple authentication mechanisms in the TSS of [ST] [\[redacted\]](#) describes the following authentication mechanisms supported by the TOE:

- Authentication based on username and password;
- Authentication based on username and smart card.

The TSS provides detailed descriptions on how those two authentication mechanisms work.

Password-based authentication:

The user provides the username and password. A cryptographic key is derived from the password using a Password-Based Derivation Key Function 2 (PBKDF2) with SHA-256. If that derived key can successfully unwrap the user's keybag, the user is granted access; otherwise, the access is denied.

Smart card authentication:

The user inserts the smart card and enters the associated PIN. Entering the correct PIN will unlock the smart card. Once unlocked, the smart card can generate a digital signature. If the TOE can verify the signature successfully with the paired certificate, the user is granted access; otherwise, the access is denied.

Guidance Assurance Activities

Assurance Activity AA-FIA_UAU.5.2-AGD-01

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

Summary

Per the TSS for FIA_UAU.5 in the [ST] [\[redacted\]](#), the TOE supports authentication based on

- username/password, and
- username/smart card.

[CCGUIDE] provides the configuration guidance for each authentication mechanism as follows:

- Username/password: Section 3.3 *Password Policy*,
- Username/smart card: Section 3.2.4 *Smart Card Authentication*.

Test Assurance Activities

Assurance Activity AA-FIA_UAU.5.2-ATE-01

- **Test 1:** For each authentication mechanism selected, the evaluator will enable that mechanism and verify that it can be used to authenticate the user at the specified authentication factor interfaces.
- **Test 2:** For each authentication mechanism rule, the evaluator will ensure that the authentication mechanism(s) behave as documented in the TSS.

Summary

Test 1: this test is covered by Test 1 of FIA_UAU.5.1.

Test 2: this test is covered by Test 1 and Test 2 of FIA_UAU.5.1.

2.2.4.9 X.509 Certificate Validation (FIA_X509_EXT.1)

FIA_X509_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FIA_X509_EXT.1.1-ASE-01

The evaluator will ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

[TD0715] 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 7.2.4.9 *FIA_X509_EXT.1 X.509 Certificate validation* in the TSS of [ST] describes where and when the validation of certificates is performed, as shown below:

" X.509 certificates are validated when imported into the TOE's trusted certificate store, during session establishment with a peer, and prior to presenting a certificate to the peer during trusted channel implementation using TLS for mutual authentications. "

From the information above, the evaluator determined that the check of validity of the certificates takes place at TOE's trusted certificate store.

Section 7.2.4.9 of [ST] also describes the following certificate path validation algorithm employed by the TOE:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate;
- The OS shall validate a certificate path by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The TSF shall validate that any CA certificate includes caSigning purpose in the key usage field.
- The OS shall validate the revocation status of the certificate using OCSP. The certificate is accepted if its revocation status cannot be determined.

Section 6.1.4.9 X.509 Certificate Validation (FIA_X509_EXT.1) of [ST] has the following statement:

" The OS shall validate the revocation status of the certificate using OCSP as specified in RFC 6960 with no exceptions. "

Therefore, there are not exceptional use cases where the TOE cannot perform revocation checking.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_X509_EXT.1.1-ATE-01

[TD0715] The tests described must be performed in conjunction with the other certificate services evaluation activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. The evaluator will create a chain of at least four certificates: the node certificate to be tested, two Intermediate CAs, and the self-signed Root CA.

- Test 1: The evaluator shall demonstrate that validating a certificate without a valid certification path results in the function failing, for each of the following reasons, in turn:
 - by establishing a certificate path in which one of the issuing certificates is not a CA certificate,
 - by omitting the basicConstraints field in one of the issuing certificates,
 - by setting the basicConstraints field in an issuing certificate to have CA=False,
 - by omitting the CA signing bit of the key usage field in an issuing certificate, and
 - by setting the path length field of a valid CA field to a value strictly less than the certificate path.The evaluator shall then establish a valid certificate path consisting of valid CA certificates, and demonstrate that the function succeeds. The evaluator shall then remove trust in one of the CA certificates, and show that the function fails.
- Test 2: The evaluator will demonstrate that validating an expired certificate results in the function failing.
- Test 3 [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 shall be performed for each method. The evaluator will test revocation of the node certificate and revocation of the intermediate CA certificate (i.e. the intermediate CA certificate should be revoked by the root CA). 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 4: If any OCSP option is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set and verify that validation of the CRL fails.
- Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)
- Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)
- Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The signature of the certificate will not validate.)
- Test 8a: (Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.
- Test 8b: (Conditional on support for EC certificates as indicated in FCS_COP.1(3)). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

- *Test 9 [Conditional, to be performed if exceptions to performing revocation are selected]: 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.*

Summary

The nscurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

Test 1 - Success: the evaluator used OpenSSL to generate a valid 4-way certificate chain. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was successfully established, proving the certificate path validation succeeded.

Test 1 - Failure: for each of the below failure cases, the evaluator configured the webserver to use the generated certificate chain and verified that a TLS connection was not established, proving the certificate path validation failed.

- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates is not a CA.
- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates does not have the basicConstraints extension.
- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates has the CA flag in the basicConstraints extension set to FALSE.
- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates has the CA signing bit missing.
- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates has a path length of zero.
- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates is missing (removed trust).

Test 2: the evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates is expired. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was not established, proving that the certificate path validation failed.

The [ST] claims OSCP to validate the revocation status of a certificate, with no exceptions. A local OpenSSL server was used to provide OSCP responses.

Test 3 - Success: The evaluator used OpenSSL to generate a valid 4-way certificate chain with no revoked certificates. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was successfully established, proving the certificate path validation succeeded.

Test 3 - Failure: for each of the below failure cases, the evaluator configured the webserver to use the generated certificate chain and verified that a TLS connection was not established, proving the certificate path validation failed.

- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the intermediate certificates is revoked using OSCP.
- The evaluator used OpenSSL to generate a 4-way certificate chain in which one of the leaf certificates is revoked using OSCP.

Test 4: the evaluator used OpenSSL to generate a certificate which does not have the OSCP signing purpose. Then, the evaluator configured the OpenSSL OSCP server to use the generated certificate. Finally, the evaluator verified that a TLS connection was successfully established, proving the certificate path validation succeeded (i.e. the OSCP response was rejected).

Test 5: the evaluator used OpenSSL to generate a certificate with the first 8 bytes modified. Then, the evaluator attempted to import this certificate into the macOS Keychain. The evaluator confirmed that the certificate was not imported due to a validation error.

Test 6: the evaluator attempted to import a certificate with the last bytes modified into the macOS Keychain. The evaluator confirmed that the certificate was not imported due to a validation error.

Test 7: the evaluator attempted to import a certificate with the public key modified into the macOS Keychain. The evaluator confirmed that the certificate was not imported due to a validation error.

Test 8a: the evaluator used OpenSSL to generate a valid 4-way EC certificate chain. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was successfully established, proving the certificate path validation succeeded.

Test 8b: the evaluator used OpenSSL to generate a 4-way EC certificate chain using explicit EC parameters. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was not established, proving the certificate path validation failed.

Test 9: Not applicable: no exceptions are defined.

FIA_X509_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_X509_EXT.1.2-ATE-01

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

Summary

The nsurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

Test 1: the evaluator used OpenSSL to generate a 4-way certificate chain with the basicConstraints extension not present. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was not established, proving that the certificate path validation failed.

Test 2: the evaluator used OpenSSL to generate a 4-way certificate chain with the CA flag in the basicConstraints extension not set. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was not established, proving that the certificate path validation failed.

Test 3: the evaluator used OpenSSL to generate a 4-way certificate chain with the CA flag in the basicConstraints extension set to TRUE. Then, the evaluator configured the webserver to use the generated certificate chain. Finally, the evaluator verified that a TLS connection was successfully established, proving the certificate path validation succeeded.

2.2.4.10 X.509 Certificate Authentication (FIA_X509_EXT.2)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FIA_X509_EXT.2-ATE-01

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

The evaluator will repeat the activity for any other selections listed.

Summary

The [ST] [\[ST\]](#) claims X.509v3 certificates for the TLS and HTTPS protocols. However, a TLS connection is inherently established as part of an HTTPS connection. Consequently, TLS is not tested separately.

The nscurl utility (which uses the OS's TLS and X.509 implementations) was used to establish HTTPS connections to a local OpenSSL webserver. All HTTPS packets were logged using tcpdump.

The evaluator configured the webserver to use an X.509v3 certificate. Then, the evaluator verified that an HTTPS connection was successfully established.

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-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 7.2.5.3 *FMT_SMF_EXT.1 Specification of management functions* in the TSS of [ST] [\[ST\]](#) serves as TSS for both FMT_MOF_EXT.1 and FMT_SMT_EXT.1. This section provides two lists of management functions. The management functions in the first list are accessible by the Administrator; and the management functions in the second list are accessible by the User.

The management functions which are in the first list but not in the second list are the management functions that are restricted to only Administrators. Users are prevented from accessing those Administrator-only management functions by the requirement of entering the correct Administrator password.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FMT_MOF_EXT.1-ATE-01

- **Test 1:** For each function that is indicated as restricted to the administrator, the evaluation shall perform the function as an administrator, as specified in the Operational Guidance, and determine that it has the expected effect as outlined by the Operational Guidance and the SFR. The evaluator 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 examined the management functions listed in Table 10 of the [ST] as mentioned below.

Function 3: the evaluator attempted to modify the auditd configuration to set the capacity to 5MB, as a non-administrator (non-root) user. The evaluator verified that the attempt was unsuccessful.

Function 4: the evaluator imported a configuration profile to configure minimum password length and attempted to install it. The evaluator verified that an administrator password was required to complete the installation.

Function 5: the evaluator imported a configuration profile to configure minimum number of special characters in password and attempted to install it. The evaluator verified that an administrator password was required to complete the installation.

Function 10: the evaluator attempted to enable and disable the socketfilterfw firewall, as a non-administrator (non-root) user. The evaluator verified that the attempts were unsuccessful.

Function 12: the evaluator downloaded an enrollment profile to enroll into remote management and attempted to install it. The evaluator verified that an administrator password was required to complete the installation.

Function 13: the evaluator attempted to modify the syslog configuration to set the server IP address to 1.2.3.4, as a non-administrator (non-root) user. The evaluator verified that the attempt was unsuccessful.

Function 14: the evaluator attempted to enable and disable auditd, as a non-administrator (non-root) user. The evaluator verified that the attempts were unsuccessful.

Function 15: the evaluator attempted to change the NTP server to 0.pool.ntp.org, as a non-administrator (non-root) user. The evaluator verified that the attempt was unsuccessful.

Function 16: the evaluator attempted to enable and disable automatic update checking, downloading, or installation, as a non-administrator (non-root) user. The evaluator verified that the attempts were unsuccessful.

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

Section 7.2.5.4 *FMT_SMF_EXT.1/BT Specification of management functions* in the TSS of [ST] serves as TSS for both FMT_MOF_EXT.1/BT and FMT_SMT_EXT.1/BT. It is specified in this section that the Bluetooth-related management function BT-1, to disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes, is supported by the TOE.

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 management function to describe how the function is performed and any role restrictions on the subjects that are authorized to perform the function.

Summary

Per the TSS for FMT_MOF_EXT.1/BT in the [ST], the TOE supports only one Bluetooth management function, BT-1. Both Administrators and Standard user roles are authorized to perform BT-1 function.

The following table lists the Bluetooth management functions claimed in the [ST] and the [CCGUIDE] sections providing the corresponding instructions.

Table 11: Management functions (Bluetooth)

#	Management Function	Administrator	User	Guidance section
BT-1	Configure the Bluetooth trusted channel. <ul style="list-style-type: none"> Disable/enable the Discoverable (for BR/EDR) and Advertising (for LE) modes; 	M	X	3.12.3

Application Note: M - Mandatory support by the specified role. X - Supported by the specified role.

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

This SFR is not applicable, as there are no Bluetooth security functions restricted to the administrator.

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-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

Based on the management functions specified in [OSPPv4.2.1], the [ST] claims a collection of management functions supported by the TOE. Section 3 *Configuration and Management* of [CCGUIDE] provides the instructions on performing each of those management functions supported by the TOE.

The following table lists the management functions which are defined in [OSPPv4.2.1] and claimed in the [ST], as well as the corresponding [CCGUIDE] sections which provide the instructions.

Table 12: Management functions (Operating System)

#	Management Function	Administrator	User	Guidance section
1	Enable/disable screen lock	M	X	3.4.1
2	Configure screen lock inactivity timeout	M	X	3.4.2
3	Configure local audit storage capacity	X	-	3.8.2
4	Configure minimum password length	X	-	3.3
5	Configure minimum number of special characters in password	X	-	3.3
6	Configure minimum number of numeric characters in password	-	-	N/A
7	Configure minimum number of uppercase characters in password	-	-	N/A
8	Configure minimum number of lowercase characters in password	-	-	N/A
9	Configure lockout policy for unsuccessful authentication attempts through timeouts between attempts, limiting number of attempts during a time period	-	-	N/A
10	Configure host-based firewall	X	-	3.6
11	Configure name/address of directory server with which to bind	-	-	N/A
12	Configure name/address of remote management server from which to receive management settings	X	-	3.7
13	Configure name/address of audit/logging server to which to send audit/logging records	X	-	3.8.1
14	Configure audit rules	X	-	3.8.2
15	Configure name/address of network time server	X	-	3.9
16	Enable/disable automatic software update	X	-	3.10
17	Configure Wi-Fi interface	X	X	3.11
18	Enable/disable Bluetooth interface	-	X	3.12.1

#	Management Function	Administrator	User	Guidance section
19	Enable/disable no other external interfaces	-	-	N/A
20	No other management functions to be provided by the TSF	-	-	N/A

Application Note: M - Mandatory support by the specified role. X - Supported by the specified role. Grey/Hyphen - Not supported by the specified role.

Test Assurance Activities

Assurance Activity AA-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 in FMT_SFM_EXT.1.1. 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

The evaluator examined the management functions listed in Table 10 of the [ST] as mentioned below.

Function 1: the evaluator imported a configuration profile to configure the maximum screen lock timeout and installed it. The evaluator verified that the screen lock timeout is limited to the 2 minutes as specified in the configuration profile. Then, the evaluator removed the configuration profile and imported a configuration profile with no screen lock timeout restrictions. Finally, the evaluator verified that the screen lock timeout could be set to any value.

Function 2: this test is covered by the test for function 1.

Function 3: the evaluator logged in as root using sudo and successfully modified the auditd configuration to set the capacity to 5MB.

Function 4: the evaluator imported a configuration profile to configure minimum password length and installed it. Then, the evaluator attempted to change the password to a 2-character password. The evaluator verified that such attempt fails.

Function 5: the evaluator imported a configuration profile to configure minimum number of special characters in password and installed it. Then, the evaluator attempted to change the password to a 8-character password without any special character. The evaluator verified such attempt fails.

Function 10: the evaluator logged in as root using sudo and successfully enabled and disabled the socketfilterfw firewall.

Function 12: the evaluator downloaded an enrollment profile to enroll into remote management and installed it. The evaluator verified that the device was successfully enrolled into remote management.

Function 13: the evaluator logged in as root using sudo and successfully modified the syslog configuration to set the server IP address to 1.2.3.4.

Function 14: the evaluator logged in as root using sudo and successfully enabled and disabled auditd.

Function 15: the evaluator logged in as root using sudo and successfully changed the NTP server to 0.pool.ntp.org.

Function 16: the evaluator logged in as root using sudo and successfully enabled and disabled automatic update checking, downloading, and installation.

Function 17 - administrator: the evaluator logged in as root using sudo and successfully added a new wireless network using networksetup.

Function 17 - user: the evaluator logged in as a user. Then, the evaluator successfully added a new wireless network using the System Settings UI.

Function 18: the evaluator successfully enabled and disabled Bluetooth using the System Settings UI.

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 7.2.5.4 *FMT_SMF_EXT.1/BT Specification of management functions* in the TSS of [ST] serves as TSS for both *FMT_MOF_EXT.1/BT* and *FMT_SMT_EXT.1/BT*. Section 7.2.5.4 states that the TOE supports both Bluetooth BR/EDR and LE, and the TOE uses Secure Simple Pairing (SSP) for security.

The Bluetooth profiles supported by the TOE are listed in this section, as shown below:

- Hands-Free Profile (HFP 1.6);
- Phone Book Access Profile (PBAP);
- Advanced Audio Distribution Profile (A2DP);
- Audio/Video Remote Control Profile (AVRCP 1.4);
- Personal Area Network Profile (PAN);
- Human Interface Device Profile (HID);
- Message Access Profile (MAP).

The TOE supports BT-1 management function only. So the assurance activities for other the management functions, e.g., BT-4, BT-5, BT-8, and BT-9, are not applicable.

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

Based on the management functions specified in [BT], the [ST] claims one Bluetooth management function, BT-1, supported by the TOE.

Section 3.12.3 *Enable/Disable Discoverable/Advertising Mode* of [CCGUIDE] provides the instructions to disable/enable the discoverable (for BR/EDR) and advertising (for LE) modes. This is controlled through Bluetooth Sharing option in System Setting > General pane.

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 first disabled Bluetooth on the TOE, and used a Linux system to verify that the TOE is not visible. Then, the evaluator enabled Bluetooth on the TOE, verified that the TOE is visible to the Linux system, and successfully paired the Linux system to the TOE. These findings were confirmed using hcidump.

Test 2 through 9: Not applicable as the ST does not claim these functions.

2.2.6 Protection of the TSF (FPT)

2.2.6.1 Access controls (FPT_ACF_EXT.1)

FPT_ACF_EXT.1.1

TSS Assurance Activities

Assurance Activity AA-FPT_ACF_EXT.1.1-ASE-01

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

Summary

Section 7.2.6.1 *FPT_ACF_EXT.1 Access controls* in the TSS of [ST] specifies the locations of kernel drivers/modules, security audit logs, shared libraries, system executables, and system configuration files, as below:

Kernel drivers/modules

/System/Library/Extensions/

Security audit logs

/var/audit

Shared libraries

~/Library/Frameworks
/System/Library/Frameworks
/System/Library/PrivateFrameworks

System configuration files

/Library/Preferences
~/Library/Preferences
/etc/security/audit-control

System executables (Applications)

/System/Applications
/Applications

As mentioned in Section 7.2.6.1 of [ST], the TOE provides access control policy through System Integrity Protection and Standard file permissions (FDP_ACF_EXT.1). System Integrity Protection is designed to allow modification of protected files only by processes that are signed by Apple and have special entitlements to write to system files. System Integrity Protection is used to protect the following parts of the system from unauthorized modification:

Kernel drivers/modules

/System/Library/Extensions/

Shared libraries

~/Library/Frameworks

/System/Library/Frameworks

/System/Library/PrivateFrameworks

System executables (Applications)

/System/Applications

/Applications (Apps that are installed with the TOE but updated independently)

Standard file permissions are used to protect the following from unauthorized modification:

Security audit logs

/var/audit

System configuration files

/Library/Preferences

~/Library/Preferences

/etc/security/audit-control

System executables (Applications)

/Applications (Apps installed by a traditional installer or copied into /Applications)

Standard file permissions are also used to prevent unprivileged users from reading Security audit logs.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities**Assurance Activity AA-FPT_ACF_EXT.1.1-ATE-01**

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

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

Summary

The evaluator created an unprivileged user account ("testuser").

Test 1: as testuser, the evaluator attempted to modify all *.kext (kernel extension) files in /System/Library. The evaluator verified that all attempts were unsuccessful.

Test 2: as testuser, the evaluator attempted to modify all files in /var/audit. The evaluator verified that all attempts were unsuccessful.

Test 3: as testuser, the evaluator attempted to modify all *.dylib (shared library) files in /System/Library and /usr/lib. The evaluator verified that all attempts were unsuccessful.

Test 4: as testuser, the evaluator attempted to modify all files in /usr/bin. The evaluator verified that all attempts were unsuccessful.

Test 5: as testuser, the evaluator attempted to modify all files in /etc and /private/etc. The evaluator verified that all attempts were unsuccessful.

Test 6: the [ST] claims applications and TSF-data as additional components protected from modification. As testuser, the evaluator attempted to modify all *.app (application) and *.plist (TSF-data) files in /System/Library. The evaluator verified that all attempts were unsuccessful.

FPT_ACF_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_ACF_EXT.1.2-ATE-01

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

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

Summary

The evaluator created two unprivileged user accounts ("testuser_1" and "testuser_2").

Test 1: as testuser_1, the evaluator attempted to read from all files in /var/audit. The evaluator verified that all attempts were unsuccessful.

Test 2: as testuser_2, the evaluator attempted to read from all files in the /Library/Keychains/ folder for testuser_1. The evaluator verified that all attempts were unsuccessful.

Test 3: not applicable. There are no other objects specified in the assignment.

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-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 three executables tested in the evaluation are Safari (web browser), Mail (mail client), and top. For each of the selected executables, the application was started normally. After starting the application, the evaluator used the vmmap tool to list the allocated memory and recorded the memory offset used for the process. This procedure was performed twice for each application. Finally, the evaluator compared the two recorded memory offsets and verified they are different.

2.2.6.3 Stack Buffer Overflow Protection (FPT_SBOP_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FPT_SBOP_EXT.1-ASE-01

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, such as stack cookie, stack guard, and stack canaries. The TSS must include a rationale for any binaries that are not protected in this manner.

Summary

Section 7.2.6.3 *FPT_SBOP_EXT.1 Stack buffer overflow protection* in the TSS of [ST] describes the stack-based buffer overflow protections used by the TOE, which are referred to as stack canaries in the ST. Stack canaries prevent buffer-overflow attacks through detecting if the stack has been overwritten when returning from a function.

All TOE binaries are compiled with stack-based overflow protections enabled. However, some compiled binaries do not contain stack canaries for certain reasons. The evaluator verified that the TSS provides a rationale for those binaries not protected with stack canaries. The rationale lists five types of compiled binaries not containing stack canaries and explains the reasons:

- Type 1: The compiler can optimize away stack usage (which macOS heavily relies on for performance reasons).
- Type 2: Some binaries are just small entry points that rely on system frameworks for all of their functionality. There, the binary itself is going to be really small (less than ~1000 instructions, sometimes as small as 10 instructions), so is much less likely to need stack protection.
- Type 3: There are very short program/functions that do not access the stack (and just forward to system frameworks to perform the real work)
- Type 4: There are tiny binaries (very few instructions) with a single trivial function that do not need stack protections or tiny wrappers that do not make use of the stack.
- Type 5: Some binaries do not access the stack in any kind of vulnerable way.

Besides stack canaries, the TOE employs ASLR (Address Space Layout Randomization) technique to guard against buffer-overflow attacks. ASLR randomizes process address space memory locations, preventing attackers from reliably jumping to specific data that has been written to the stack.

Assurance Activity AA-FPT_SBOP_EXT.1-ASE-02

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 definition of FPT_SBOP_EXT.1 SFR in OSPP provides two options for selection operation:

- "employ stack-based buffer overflow protections", and

- "not store parameters/variables in the same data structures as control flow values".

The ST doesn't select the second option. In other words, the TOE does store parameters/variables in the same data structure as control flow values.

This assurance activity is not applicable and therefore considered to be satisfied.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_SBOP_EXT.1-ATE-01

The evaluator will also perform the following test:

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

Summary

The [ST] [\[ST\]](#) claims that all TOE binaries are protected from stack-based buffer overflow attacks using stack canaries and ASLR (tested by FPT_ASLR_EXT.1). However, note that not all TOE binaries will contain stack canaries, e.g. when the stack is not used at all, when the binaries are very small, or when the stack as used cannot be exploited.

The evaluator used an automated script to scan all binaries in the TOE, except those listed as Type 1-5 in the [ST] [\[ST\]](#). For each of those, the evaluator used otool and nm to confirm the presence of the functions "__stack_chk_fail" and "__stack_chk_guard". This shows that stack canaries were employed.

2.2.6.4 Boot Integrity (FPT_TST_EXT.1)

TSS Assurance Activities

Assurance Activity AA-FPT_TST_EXT.1-ASE-01

The evaluator will verify that the TSS section of the ST includes a comprehensive description of the boot procedures, including a description of the entire bootchain, for the TSF. 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 7.2.6.4 FPT_TST_EXT.1 Boot integrity in the TSS of [ST] [\[ST\]](#) describes the bootchain for two types of Apple Mac computers based on the CPU type:

- Apple silicon Macs: Mac computers based on Apple silicon;
- "Intel with T2" Macs: Mac computers based on Intel processor with Apple T2 security chip.

The evaluator verified that the entire bootchain on both hardware platforms is described, starting from the first step of loading the Boot ROM firmware until the last step when the TOE (macOS) begins execution. The software loaded in each step of bootchain, including bootloaders and OS kernel, is cryptographically verified using digital signature.

Although the boot procedures on both platforms are different, they have the same first step: loading the Boot ROM. The Boot ROM code contains the Apple Root CA public key, which is used to verify the digital signatures of software the TOE loads in the bootchain. The Boot ROM is immutable code, referred to as the hardware root of trust. It is laid down during chip fabrication and is audited for vulnerabilities and implicitly trusted.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TST_EXT.1-ATE-01

The evaluator will also perform the following tests:

- **Test 1:** The evaluator will perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors and that the OS properly boots.
- **Test 2:** The evaluator will modify a TSF executable that is part of the bootchain verified by the TSF (i.e. Not the first-stage bootloader) and attempt to boot. The evaluator will ensure that an integrity violation is triggered and the OS does not boot (Care must be taken so that the integrity violation is determined to be the cause of the failure to load the module, and not the fact that in such a way to invalidate the structure of the module.).
- **Test 3: [TD0493] [conditional]** If the ST author indicates that the integrity verification is performed using a public key in an X509 certificate, the evaluator will verify that the boot integrity mechanism includes a certificate validation according to in 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 1: the evaluator rebooted the TOE without making any modifications and verified that the reboot was successful.

Test 2: using specialized Apple equipment, the evaluator modified the pre-computed integrity values that are verified during the boot process. Then, the evaluator confirmed the TOE stops its boot process and the OS does not boot. This procedure was performed for both kernel space and user space integrity tests.

Test 3: not applicable. The [ST] does not claim X.509 certificates for boot integrity verification

2.2.6.5 Trusted Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.1.1-ATE-01

[TD0463] 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(3). 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 shall send a query response with a bad signature and verify that the signature verification fails. The evaluator shall then send a query response with a good signature and verify that the signature verification is successful.

Summary

The evaluator downloaded the update installation application (used to test FPT_TUD_EXT.2.2) from the Apple servers using HTTPS (i.e. HTTP over TLS). This is the exact protocol described in FTP_ITC_EXT.1.

FPT_TUD_EXT.1.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.1.2-ATE-01

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

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

Summary

The evaluator obtained the update installation application (.app file) for macOS Ventura from the Apple servers.

Test 1: the evaluator modified the application by overwriting some of the data inside with random bytes. This caused the integrity value of the application to no longer be valid. Then, the evaluator executed the application and verified that the OS refused to continue.

Test 2: the evaluator executed the original application and verified that the OS allowed the user to install the update. Note that the evaluator did not actually perform the software update, as this process is irreversible.

2.2.6.6 Trusted Update for Application Software (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.2.1-ATE-01

[TD0463] 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(3). The digital signature verification may be performed as part of a network protocol occurs as described in FTP_ITC_EXT.1. If the signature verification is not performed as part of a trusted channel, the evaluator shall send a query response with a bad signature and verify that the signature verification fails. The evaluator shall then send a query response with a good signature and verify that the signature verification is successful.

Summary

The evaluator downloaded the update installation application (used to test FPT_TUD_EXT.2.2) from the Apple servers using HTTPS (i.e. HTTP over TLS). This is the exact protocol described in FTP_ITC_EXT.1.

FPT_TUD_EXT.2.2

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

No assurance activities defined.

Test Assurance Activities

Assurance Activity AA-FPT_TUD_EXT.2.2-ATE-01

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

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

Summary

The evaluator obtained the update installation application (.app file) for macOS Ventura from the Apple servers. As the macOS installer files contain both the OS (kernel) and application (user space), these tests are identical to those described in FPT_TUD_EXT.1.2.

Test 1: the evaluator modified the application by overwriting some of the data inside with random bytes. This caused the integrity value of the application to no longer be valid. Then, the evaluator executed the application and verified that the OS refused to continue.

Test 2: the evaluator executed the original application and verified that the OS allowed the user to install the update. Note that the evaluator did not actually perform the software update, as this process is irreversible.

2.2.6.7 Write XOR Execute Memory Pages (FPT_W^X_EXT.1)

TSS Assurance Activities

No assurance activities defined.

Guidance Assurance Activities

Assurance Activity AA-FPT_WVX_EXT.1-AGD-01

The evaluator will inspect the vendor-provided developer documentation and verify that no memory-mapping can be made with write and execute permissions except for the cases listed in the assignment.

Summary

The evaluator inspected the vendor-provided developer documentation and verified that no memory-mapping can be made with write and execution permissions except for those listed in the ST.

Test Assurance Activities

Assurance Activity AA-FPT_WVX_EXT.1-ATE-01

The evaluator will also perform the following tests.

- **Test 1:** *The evaluator will acquire or construct a test program which attempts to allocate memory that is both writable and executable. The evaluator will run the program and confirm that it fails to allocate memory that is both writable and executable.*
- **Test 2:** *The evaluator will acquire or construct a test program which allocates memory that is executable and then subsequently requests additional write/modify permissions on that memory. The evaluator will run the program and confirm that at no time during the lifetime of the process is the memory both writable and executable.*
- **Test 3:** *The evaluator will acquire or construct a test program which allocates memory that is writable and then subsequently requests additional execute permissions on that memory. The evaluator will run the program and confirm that at no time during the lifetime of the process is the memory both writable and executable.*

Summary

Test 1: the evaluator wrote a C program that attempts to mmap memory using the PROT_EXEC | PROT_WRITE flags. The evaluator executed the program and verified the mmap call failed.

Test 2: the evaluator wrote a C program that first mmaps memory using the PROT_EXEC flag and subsequently attempts to modify the flags to PROT_EXEC | PROT_WRITE using mprotect. The evaluator executed the program and verified the mprotect call failed.

Test 3: the evaluator wrote a C program that first mmaps memory using the PROT_WRITE flag and subsequently attempts to modify the flags to PROT_EXEC | PROT_WRITE using mprotect. The evaluator executed the program and verified the mprotect call failed.

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-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 configured the policy banner message with "TEST TEST Warning Message TEST TEST" and rebooted the TOE. Then, the evaluator verified that the message was displayed before the user login screen is available.

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 7.2.8.1 FTP_BLT_EXT.1 *Bluetooth encryption* in the TSS of [ST]  describes the encryption techniques used for Bluetooth data transmission. Bluetooth devices are required to pair with the TOE using Secure Simple Pairing (SSP), which employs ECDH for key generation and AES for data encryption.


SSP applies to both Bluetooth BR/EDR and Bluetooth LE. Furthermore, Bluetooth encryption is always enabled.

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

It is mentioned in TSS section 7.2.8 *Trusted path/channels* of [ST]  that the Bluetooth data communication between the TOE and Bluetooth devices is always encrypted with AES-CCM-128. In addition, the TOE enforces the persistence of Bluetooth encryption. If the remote Bluetooth device stops encrypting while connected to the TOE, the TOE terminates the connection.

The encryption during Bluetooth data transmission is enforced by default. Therefore, no additional configuration is required.

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

There are no test EAs for this component. Testing for this SFR is addressed through the evaluation of 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 7.2.8.2 *FTP_BLT_EXT.2 Persistence of Bluetooth encryption* in the TSS of [ST] states that the TOE terminates the connection if a remote Bluetooth device stops encrypting 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

It is mentioned in TSS section 7.2.8 of [ST] that the Bluetooth data communication between the TOE and Bluetooth devices is always encrypted with AES-CCM-128.

The encryption during Bluetooth data transmission is not configurable. Therefore, no additional configuration is required.

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 first connected and paired a Linux system to the TOE. Note that Linux has a default encryption key size of 128 bits, identical to the TOE minimum encryption key size (also verified using hcidump). Then, the evaluator disabled Link Level Encryption using hcitool. Using hcidump, the evaluator confirmed that the Bluetooth connection was immediately terminated.

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 7.2.8.3 *FTP_BLT_EXT.3/BR Bluetooth encryption parameters (BR/EDR)* in the TSS of [ST] serves as TSS for both *FTP_BLT_EXT.3/BR* and *FTP_BLT_EXT.3/LE*. Section 7.2.8.3 specifies the TOE uses AES CCM mode to secure Bluetooth communications in both BR/EDR radio and LE radio.

The key size for both BR/EDR encryption and LE encryption is 128 bits. That is the only key size supported by the TOE for Bluetooth communications. Therefore, the minimum key size is 128 bits and smaller key sizes cannot be negotiated.

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

It is mentioned in TSS section 7.2.8 of [ST] that the Bluetooth connections via BR/EDR are encrypted using 128-bit AES Counter with CBC-MAC (AES-CCM-128). No other key sizes are supported.

The minimum encryption key size for BR/EDR encryption is not configurable. Therefore, no additional configuration is required.

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

Note that Linux has a default encryption key size of 128 bits, identical to the TOE minimum encryption key size. As part of these tests, Bluetooth Low Energy (LE) was disabled using BlueZ.

Test 1: the evaluator successfully connected and paired a Linux system to the TOE. Using hcidump, the evaluator verified that the key used for encryption was 16 bytes (128 bits) long.

Test 2: Not applicable. The TOE cannot configure the key size.

Test 3: the Bluetooth source code for the Linux system was modified to reduce the key size to 8 bits. Then, the evaluator attempted to connect and pair the Linux system to the TOE. Using hcidump, the evaluator verified that the key used for encryption was only 1 byte (8 bits) long. Moreover, the TOE immediately terminated the connection and the pairing did not complete.

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

Please see AA-BTPPM-FTP_BLT_EXT.3-BR-ASE-01 for the provided information.

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

It is mentioned in TSS section 7.2.8 of [ST] that the Bluetooth connections via LE are encrypted using 128-bit AES Counter with CBC-MAC (AES-CCM-128). No other key sizes are supported.

The minimum encryption key size for LE encryption is not configurable. Therefore, no additional configuration is required.

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

Note that Linux has a default encryption key size of 128 bits, identical to the TOE minimum encryption key size. As part of these tests, Bluetooth Low Energy (LE) was enabled using BlueZ.

Test 1: the evaluator successfully connected and paired a Linux system to the TOE. Using hcidump, the evaluator verified that the key used for encryption was 16 bytes (128 bits) long.

Test 2: Not applicable. The TOE cannot configure the key size.

Test 3: the Bluetooth source code for the Linux system was modified to reduce the key size to 8 bits. Then, the evaluator attempted to connect and pair the Linux system to the TOE. Using hcidump, the evaluator verified that the key used for encryption was only 1 byte (8 bits) long. Moreover, the TOE immediately terminated the connection and the pairing did not complete.

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-FTP_ITC_EXT.1-ATE-01

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

Summary

The [ST] specifies that the TLS protocol is selected to provide a trusted channel to the update server and other authenticated TLS servers.

The nsurl utility (which uses the OS's TLS and X.509 implementations) was used to establish TLS connections to a local OpenSSL webserver. All TLS packets were logged using tcpdump.

The evaluator verified that a TLS connection was successfully established, proving the trusted channel is established in conformance with the selected protocol.

2.2.8.6 Trusted Path (FTP_TRP.1)

TSS Assurance Activities

Assurance Activity AA-FTP_TRP.1-ASE-01

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

Summary

Section 7.2.8.6 FTP_TRP.1 *Trusted path* in the TSS of [ST] confirms that the TOE provides a trusted path between itself and local users, which is specified by the FTP_TRP.1 SFR in Section 6.1.8.6 *Trusted Path (FTP_TRP.1)* of [ST]. Section 7.2.8.6 also clarifies that the remote administrative communication path is not supported in the evaluated TOE configuration.

As per the Application Note in [OSPPv4.2.1],

" If local users access is selected and no unprotected traffic is sent to remote users, then this requirement is met. "

the evaluator determined that the requirement of using the trusted path for all remote administrative actions does not apply here. This assurance activity is not applicable and therefore considered to be satisfied.

Guidance Assurance Activities

Assurance Activity AA-FTP_TRP.1-AGD-01

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

Summary

Per the TSS for FTP_TRP.1 in the [ST], the TOE provides a trusted path between itself and local users, and the remote administrative communication path is not supported in the evaluated TOE configuration. The following Application Note is provided in [OSPPv4.2.1],

" If local users access is selected and no unprotected traffic is sent to remote users, then this requirement is met. "

Based the above information, the evaluator determined that the requirement of establishing the remote administrative sessions does not apply here. This assurance activity is not applicable and therefore considered to be satisfied.

Test Assurance Activities

Assurance Activity AA-FTP_TRP.1-ATE-01

The evaluator will also perform the following tests:

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

Summary

Test 1: Not applicable. The [ST] does not claim support for remote administration.

Test 2: Not applicable. The [ST] does not claim support for remote administration.

Test 3: Not applicable. The [ST] does not claim support for remote administration.

Test 4: Not applicable. The [ST] does not claim support for remote administration.

2.3 Security Assurance Requirements

2.3.1 Guidance documents (AGD)

2.3.1.1 Operational user guidance (AGD_OPE.1)

Assurance Activity AA-AGD_OPE.1-AGD-01

Some of the contents of the operational guidance are verified by the guidance evaluation activities in Section 5.1 of [OSPPv4.2.1] and evaluation of the OS according to the [CEM].

If cryptographic functions are provided by the OS, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the OS. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the OS. The documentation must describe the process for verifying updates to the OS by verifying a digital signature - this may be done by the OS or the underlying platform. The evaluator will verify that this process includes the following steps: Instructions for obtaining the update itself. This should include instructions for making the update accessible to the OS (e.g., placement in a specific directory). Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the hash/digital signature. The OS will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

Summary

As mentioned in Section 1.7 *Cryptographic engine warning* of [CCGUIDE], the TOE employs Apple Corecrypto Module to implement all the cryptographic functionalities. Apple Corecrypto Module is the only cryptographic engine associated with the evaluated TOE configuration.

Section 2.2 *Update* of [CCGUIDE] describes the steps to upgrade the OS software, in both CLI and GUI. The TOE supports Apple's Rapid Security Response feature, which allows Apple to provide security fixes to users more frequently.

Section 2.4 *Installation/Verification Process* of [CCGUIDE] discusses the technical details of the digital signature verification for OS updates. Signature verification of the TOE image is performed by the Secure Enclave Processor (SEP) when the system reboots. As the kernel boots, each stage of the boot process validates the digital signatures of the next stage using the public key that is embedded in the Mac's Boot ROM during manufacturing.

Section 3 *Configuration and Management* is organized according to the security functionalities of the TOE in the scope of evaluation. Furthermore, Section 1.3 *Excluded Functionality* of [CCGUIDE] lists the following functionalities which are not included in the evaluated TOE configuration:

- Bonjour,
- VPN split tunnel,
- Siri.

2.3.1.2 Preparative procedures (AGD_PRE.1)

Assurance Activity AA-AGD_PRE.1-AGD-01

As indicated in the introduction above, there are significant expectations with respect to the documentation—especially when configuring the operational environment to support TOE functional requirements. The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

Summary

The developer has provided the [CCGUIDE] document as the primary user guidance. Section 1.1 *Target of Evaluation* of [CCGUIDE] provides a table (Table 2 Hardware platforms) listing all hardware platforms where the TOE runs as operating system. All the hardware platforms supporting the TOE are Apple's Mac computers.

While performing the evaluation activities under Independent testing (ATE_IND), the evaluator found the TOE provides the same user interfaces on different platforms. The user guidance describes the preparative and operational procedures in the context of the user interfaces provided by the TOE. Therefore, the evaluator determined that the user guidance adequately addresses all platforms claimed for the TOE in the ST.

2.3.2 Life-cycle support (ALC)

2.3.2.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activity AA-ALC_CMC.1-ALC-01

The evaluator will check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator will check the AGD guidance and OS samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the OS, the evaluator will examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Summary

[ST] section 1.2, *TOE Identification*, identifies the TOE as Apple macOS 13 Ventura . Appendix A.1 *Devices Covered by this Evaluation* of [ST] lists the mobile devices covered by the evaluation in Table 16 "Hardware platforms".

The administrative guidance [CCGUIDE] contains the TOE reference as Apple macOS 13 Ventura.

In addition, for the delivery of the evaluated TOE, [CCGUIDE] section 2, *Installation, and Update, and Recovery* provides instructions on how to obtain the evaluated TOE.

2.3.2.2 TOE CM coverage (ALC_CMS.1)

Assurance Activity AA-ALC_CMS.1-ALC-01

The "evaluation evidence required by the SARs" in this PP is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the OS is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the 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 shall provide information on how to configure the environment to ensure that buffer overflow protection mechanisms in the environment(s) are invoked (e.g., compiler and linker flags). The evaluator will ensure that this documentation also includes an indication of whether such protections are on by default, or have to be specifically enabled. The evaluator will ensure that the TSF is uniquely identified (with respect to other products from the TSF vendor), and that documentation provided by the developer in association with the requirements in the ST is associated with the TSF using this unique identification.

Summary

The evaluator 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. See <https://developer.apple.com/xcode/> for more details on Xcode IDE.

The vendor provided "Apple Platform Security" [AP_SEC] where section *Gatekeeper and runtime protection in macOS* describes runtime protection as follows:

System files, resources, and the kernel are shielded from a user's app space. All apps from the App Store are sandboxed to restrict access to data stored by other apps. If an app from the App Store needs to access data from another app, it can do so only by using the APIs and services provided by macOS.

All third-party apps are "sandboxed," so they are restricted from accessing files stored by other apps or from making changes to the device. Sandboxing is designed to prevent apps from gathering or modifying information stored by other apps. Each app has a unique home directory for its files, which is randomly assigned when the app is installed.

In addition, the ST section 7.2.6.3 describes that TOE protects all TOE binaries from stack-based buffer overflow attacks using the following mechanisms:

- ASLR to randomize executable locations on the stack, preventing attackers from jumping to specific data that has been written to the stack
- Stack canaries to detect if the stack has been overwritten when returning from a function

All TOE binaries are compiled with stack-based overflow protection enabled. Majority of compiled binaries contain stack canaries except for those in the subset identified in the ST.

The ST and [AP_SEC] also describe the System Integrity Protection technology which restricts components to read-only in specific critical file system locations to help prevent malicious code from modifying them. System Integrity Protection is a computer-specific setting that's on by default.

2.3.2.3 Extension: Timely Security Updates (ALC_TSU_EXT.1)

Assurance Activity AA-ALC_TSU_EXT.1-ALC-01

The evaluator will verify that the TSS contains a description of the timely security update process used by the developer to create and deploy security updates. The evaluator will verify that this description addresses the entire application. The evaluator will also verify that, in addition to the OS developer's process, any third-party processes are also addressed in the description. The evaluator will also verify that each mechanism for deployment of security updates is described.

The evaluator will verify that, for each deployment mechanism described for the update process, the TSS lists a time between public disclosure of a vulnerability and public availability of the security update to the OS patching this vulnerability, to include any third-party or carrier delays in deployment. The evaluator will verify that this time is expressed in a number or range of days.

The evaluator will verify that this description includes the publicly available mechanisms (including either an email address or website) for reporting security issues related to the OS. The evaluator shall verify that the description of this mechanism includes a method for protecting the report either using a public key for encrypting email or a trusted channel for a website.

Summary

Section 7.1.1 of [ST] *Timely Security Updates* provides information regarding timely security updates to the TOE.

Apple generally does not disclose, discuss, or confirm security issues until a full investigation is complete and any necessary patches or releases are available. Once a security issue has been confirmed and a patch has been made available, references containing technical details on the patches are made available and Common Vulnerabilities and Exposures (CVEs) are released.

Apple distributes information about security issues in its products through its "Apple Security Updates" page (<https://support.apple.com/HT201222>).

Security advisories are also provided through the security announce mailing list (<https://lists.apple.com/mailman/listinfo/security-announce/>).

Potential security vulnerabilities can be reported by following the procedures on the "Report a security or privacy vulnerability" page (<https://support.apple.com/HT201220>). This includes sending an email to "product-security@apple.com" and includes the ability to encrypt information using the Apple Product Security PGP key (<https://support.apple.com/kb/HT201214>).

Additionally, the TOE supports Apple's Rapid Security Responses (RSR) feature. This feature allows security updates to the TOE to be applied when available without waiting for the next cumulative macOS update. RSR are automatically applied to TOE devices by default and, if necessary, the user will be prompted to restart the device.

Users can check the RSR feature settings by going to Settings > General > Software Update > Automatic Updates then making sure that "Security Responses & System Files" is turned on.

Apple releases information about security issues of the TOE, thus third-party or carrier delays in deployment is not applicable.

2.3.3 Tests (ATE)

2.3.3.1 Independent testing - conformance (ATE_IND.1)

Assurance Activity AA-ATE_IND.1-ATE-01

The evaluator will prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM] and the body of this PP's 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 shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

Summary

The Detailed Test Report [DTR] specifies all the tests covering the assurance activities from [OSPPv4.2.1] and [BT]. 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 test configuration is made up

of the TOE devices connected to a private WLAN network which also hosts a Linux system and a macOS Server. The macOS Server is created on a macOS system upon which the macOS Server software, available from the Apple app store, is installed.

The following tools were used for testing:

- Homebrew along with
 - OpenSSL 3 version 3.1.4
 - GnuTLS version 3.8.1 (and its dependencies: nettle, p11-kit)
 - tshark version 4.2.0
 - pkg-config version 0.29.2_3
- Expect
- Linux system (Arch Linux kernel 6.6.1) with KDE (version 5.27.9), BlueZ utilities (version 5.66), and Wireshark (version 4.3.0)
- Special Bluetooth dongle that allows MAC address programming (LogiLink BT0037)
- YubiKey 5C for PIN authentication

2.3.4 Vulnerability assessment (AVA)

2.3.4.1 Vulnerability survey (AVA_VAN.1)

Assurance Activity AA-AVA_VAN.1-AVA-01

The evaluator will generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Summary

The evaluator searched for publicly known vulnerabilities applicable to the TOE from the following sources:

- "Apple security releases" web page
<https://support.apple.com/HT201222>
- MITRE Common Vulnerabilities and Exposures (CVE) List
https://cve.mitre.org/cve/search_cve_list.html
- NIST National Vulnerability Database
<https://web.nvd.nist.gov/view/vuln/search>
- CISA Known Exploited Vulnerabilities Catalog
<https://www.cisa.gov/known-exploited-vulnerabilities-catalog>

The following search terms were used for CVE search on MITRE, NIST, and CISA web sites:

- macOS 13.2.1
- macOS corecrypto
- macOS commoncrypto
- macOS http
- macOS https

- macOS tls
- macOS tcp
- macOS ip
- macOS Bluetooth
- macOS T2
- Apple silicon
- Apple M1
- Apple M1 Ultra
- Apple M1 Max
- Apple M1 Pro
- Apple M2
- Apple M2 Pro
- Apple M2 Max
- Apple T2
- Intel with T2
- Apple Secure Enclave
- Apple Security Enclave Processor
- Apple SEP
- Amber Lake
- Cascade Lake
- Coffee Lake
- Comet Lake
- Skylake
- ARM 8.5
- ARM 8.6
- Core i5-1030NG7
- Core i5-1038NG7
- Core i5-10500
- Core i5-10600
- Core i5-8210Y
- Core i5-8257U
- Core i5-8259U
- Core i5-8279U
- Core i5-8500
- Core i5-8500B
- Core i5-8557U
- Core i5-8600
- Core i5-9600K
- Core i7-1060NG7
- Core i7-1068NG7
- Core i7-10700K
- Core i7-8557U
- Core i7-8559U

- Core i7-8569U
- Core i7-8700
- Core i7-8700B
- Core i7-8750H
- Core i7-8850H
- Core i7-9750H
- Core i9-10910
- Core i9-8950HK
- Core i9-9880H
- Core i9-9900K
- Core i9-9980HK
- Xeon W-2140B
- Xeon W-2150B
- Xeon W-2170B
- Xeon W-2191B
- Xeon W-3223
- Xeon W-3235
- Xeon W-3245
- Xeon W-3265M
- Xeon W-3275M

Besides examining the fixes published by the developer, the evaluator performed CVE search on the following dates:

- 2023-05-08
- 2023-06-21
- 2023-07-25
- 2023-08-28
- 2023-09-21
- 2023-11-03
- 2023-11-21
- 2024-01-09

The evaluator's CVE search found no potential vulnerabilities in the TOE apart from the ones listed in "Apple security releases" page.

"Apple security releases" web page lists security updates and Rapid Security Responses for macOS, which address the publicly known vulnerabilities in the TOE. Rapid Security Responses (RSRs) deliver important security improvements between software updates and are available only for the latest version of the TOE. RSRs will be included in a subsequent software update. As of the date of this report, there is no RSRs for the latest version of macOS (13.6.3). Below are the security updates related to this evaluation:

- macOS Ventura 13.3
<https://support.apple.com/en-us/HT213670>
Released date: 2023-03-27
- macOS Ventura 13.3.1
<https://support.apple.com/en-us/HT213721>
Released date: 2023-04-07

- macOS Ventura 13.4
<https://support.apple.com/en-us/HT213758>
Released date: 2023-05-18
- macOS Ventura 13.4.1
<https://support.apple.com/en-us/HT213813>
Released date: 2023-06-21
- macOS Ventura 13.5
<https://support.apple.com/en-us/HT213843>
Released date: 2023-07-24
- macOS Ventura 13.5.1
This update has no published CVE entries
Released date: 2023-08-17
- macOS Ventura 13.5.2
<https://support.apple.com/en-us/HT213906>
Released date: 2023-09-07
- macOS Ventura 13.6
<https://support.apple.com/en-us/HT213931>
Released date: 2023-09-21
- macOS Ventura 13.6.1
<https://support.apple.com/en-us/HT213985>
Released date: 2023-10-25
- macOS Ventura 13.6.2
This update has no published CVE entries
Released date: 2023-11-07
- macOS Ventura 13.6.3
<https://support.apple.com/en-us/HT214038>
Released date: 2023-12-11

All publicly known vulnerabilities applicable to the TOE have been addressed in subsequent releases or patches of macOS after the TOE version in the evaluated configuration (13.2.1). As of the date of this report, the current version of macOS is 13.6.3. Since the TOE version is no longer available for distribution, the evaluator determined that no further consideration and testing of these vulnerabilities are required.

A Appendixes

A.1 References

AP_SEC	Apple Platform Security Author(s) Apple Inc. Date May 2022 Location https://support.apple.com/guide/security
BT	PP-Module for Bluetooth Version 1.0 Date 2021-04-15 Location https://www.niap-ccevs.org/Profile/Info.cfm?PPID=425&id=425
CC	Common Criteria for Information Technology Security Evaluation Version 3.1R5 Date April 2017 Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R5.pdf Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART2V3.1R5.pdf Location http://www.commoncriteriaportal.org/files/ccfiles/CCPART3V3.1R5.pdf
CCEVS-EDAC	Entropy Documentation and Assessment Clarification Annex Author(s) NVLAP Version 1.0 Date 2015-05-08 Location https://www.niap-ccevs.org/Documents_and_Guidance/ccevs/Entropy%20Documentation%20and%20Assessment%20Clarification.pdf
CCEVS-LG	CCEVS LabGrams Author(s) NIAP Date November 2023 Location https://www.niap-ccevs.org/Documents_and_Guidance/labgrams.cfm
CCEVS-PL	CCEVS Scheme Policy Letters Author(s) NIAP Date November 2023 Location https://www.niap-ccevs.org/Documents_and_Guidance/policy.cfm
CCEVS-PUB	CCEVS Scheme Publications Author(s) NIAP Date November 2023 Location https://www.niap-ccevs.org/Documents_and_Guidance/guidance_docs.cfm
CCEVS-TD	Technical Decisions Author(s) NIAP Date November 2023

	Location https://www.niap-ccevs.org/Documents_and_Guidance/view_tds.cfm
CCGUIDE	Apple macOS 13 Ventura Common Criteria Configuration Guide Version 1.1 Date 2024-01-12 File name agd/vid11347_macos_cc_guide_v1.1.pdf
CEM	Common Methodology for Information Technology Security Evaluation Version 3.1R5 Date April 2017 Location http://www.commoncriteriaportal.org/files/ccfiles/CEMV3.1R5.pdf
CFG_GPOS-BT_V1.0	PP-Configuration for General Purpose Operating Systems and Bluetooth Version 1.0 Date 2021-04-15 Location https://www.niap-ccevs.org/MMO/PP/CFG_GPOS-BT_V1.0.pdf
COREBT	Core Bluetooth Framework Author(s) Apple Inc. Date July 2023 received Location https://developer.apple.com/documentation/corebluetooth
DTR	Apple macOS 13 Ventura Detailed Test Report Version 1.1 Date 2024-01-08 File name ate/macOS_13_GPOS_DTR_v1.1.pdf
FIPS186-4	Digital Signature Standard (DSS) Date 2013-07-19 Location https://csrc.nist.gov/pubs/fips/186-4/final
FIPS198-1	The Keyed-Hash Message Authentication Code (HMAC) Date 2008-07-16 Location https://csrc.nist.gov/pubs/fips/198-1/final
OSPPv4.2.1	Protection Profile for General Purpose Operating Systems Version 4.2.1 Version 4.2.1 Date 2019-04-22 Location https://www.niap-ccevs.org/MMO/PP/PP_OS_V4.2.1.pdf
RFC8017	PKCS #1: RSA Cryptography Specifications Version 2.2 Author(s) K. Moriarty, B. Kaliski, J. Jonsson, A. Rusch Date 2016-11-01 Location http://www.ietf.org/rfc/rfc8017.txt
SP800-38A	Recommendation for Block Cipher Modes of Operation: Methods and Techniques Date 2001-12-01 Location https://csrc.nist.gov/pubs/sp/800/38/a/final

SP800-38C	Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality Date 2007-07-20 Location https://csrc.nist.gov/pubs/sp/800/38/c/upd1/final
SP800-38D	Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC Date 2007-11-28 Location https://csrc.nist.gov/pubs/sp/800/38/d/final
SP800-56A-Rev3	Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography Date 2018-04-16 Location https://csrc.nist.gov/pubs/sp/800/56/a/r3/final
SP800-90A-Rev1	Recommendation for Random Number Generation Using Deterministic Random Bit Generators Date 2015-06-24 Location https://csrc.nist.gov/pubs/sp/800/90/a/r1/final
ST	Apple macOS 13 Ventura Security Target Version 1.1 Date 2024-01-12 File name ase/st-macos-13-v1.1.pdf

A.2 Glossary

Augmentation

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

Authentication data

Information used to verify the claimed identity of a user.

Authorised user

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

Class

A grouping of CC families that share a common focus.

Component

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

Connectivity

The property of the TOE which allows interaction with IT entities external to the TOE.

This includes exchange of data by wire or by wireless means, over any distance in any environment or configuration.

Dependency

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

Deterministic RNG (DRNG)

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

Element

An indivisible statement of security need.

Entropy

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

Evaluation

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

Evaluation Assurance Level (EAL)

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

Evaluation authority

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

Evaluation scheme

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

Exact conformance

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

Extension

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

External entity

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

Family

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

Formal

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

Guidance documentation

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

Identity

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

Informal

Expressed in natural language.

Object

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

Operation (on a component of the CC)

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

Operation (on an object)

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

Operational environment

The environment in which the TOE is operated.

Organisational Security Policy (OSP)

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

Package

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

PP evaluation

Assessment of a PP against defined criteria.

Protection Profile (PP)

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

Random number generator (RNG)

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

Refinement

The addition of details to a component.

Role

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

Secret

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

Secure state

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

Security attribute

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

Security Function Policy (SFP)

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

Security objective

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

Security Target (ST)

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

Seed

Value used to initialize the internal state of an RNG.

Selection

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

Semiformal

Expressed in a restricted syntax language with defined semantics.

ST evaluation

Assessment of an ST against defined criteria.

Subject

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

Target of Evaluation (TOE)

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

TOE evaluation

Assessment of a TOE against defined criteria.

TOE resource

Anything useable or consumable in the TOE.

TOE Security Functionality (TSF)

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

Transfers outside of the TOE

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

True RNG (TRNG)

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

Trusted channel

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

Trusted path

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

TSF data

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

TSF Interface (TSFI)

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

User

See external entity

User data

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