www.GossamerSec.com

Assurance Activity Report for Cisco Catalyst 9400X/9600X Series Switches 17.12

Version 0.3
09/17/24

Prepared by:
Gossamer Security Solutions
Accredited Security Testing Laboratory – Common Criteria Testing
Columbia, MD 21045

Prepared for:
National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

Document: AAR-VID11465

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Revision History

Revision	Date	Authors	Summary
Version 0.1	08/20/24	Keenan	Initial draft
Version 0.2	09/11/24	Cummins	Addressed ECR comments
Version 0.3	09/17/24	Gossamer	Addressed ECR comments

The TOE Evaluation was Sponsored by:

Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134

Evaluation Personnel:

- Yoel Fortaleza
- Douglas Kalmus
- Cody Cummins
- Allison Keenan

Common Criteria Versions:

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 2: Security functional components, Version 3.1, Revision 5, April 2017
- Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1, Revision 5, April 2017

Common Evaluation Methodology Versions:

- Common Methodology for Information Technology Security Evaluation, Evaluation Methodology, Version 3.1, Revision 5, April 2017

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Table of Contents

1. Introduction 6
1.1 Equivalence 6
1.1.1 Evaluated Platform Equivalence 6
1.1.2 CAVP Equivalence 7
1.2 References 9
2. Protection Profile SFR Assurance Activities 10
2.1 Security audit (FAU) 10
2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1) 10
2.1.2 Audit Data Generation (MACsec) (MACSEC10:FAU_GEN.1/MACSEC) 12
2.1.3 User identity association (NDcPP22e:FAU_GEN.2) 13
2.1.4 Protected Audit Event Storage (NDcPP22e:FAU_STG_EXT.1) 13
2.2 Cryptographic support (FCS) 17
2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1) 17
2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2) 21
2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4) 24
2.2.4 Cryptographic Operation (AES-CMAC Keyed Hash Algorithm) (MACSEC10:FCS_COP.1/CMAC) 25
2.2.5 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS_COP.1/DataEncryption) 27
2.2.6 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash) 32
2.2.7 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS_COP.1/KeyedHash) 34
2.2.8 Cryptographic Operation (MACsec AES Data Encryption and Decryption) - per TD0728 (MACSEC10:FCS_COP.1/MACSEC) 35
2.2.9 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS_COP.1/SigGen) .. 36
2.2.10 MACsec (MACSEC10:FCS_MACSEC_EXT.1) 38
2.2.11 MACsec Integrity and Confidentiality (MACSEC10:FCS_MACSEC_EXT.2) 40
2.2.12 MACsec Randomness (MACSEC10:FCS_MACSEC_EXT.3) 42
2.2.13 MACsec Key Usage (MACSEC10:FCS_MACSEC_EXT.4) 43
2.2.14 MACsec Key Agreement - per TD0817 (MACSEC10:FCS_MKA_EXT.1) 45
2.2.15 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1) 50
2.2.16 SSH Server Protocol - per TD0631 (NDcPP22e:FCS_SSHS_EXT.1) 52

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.17 TLS Client Protocol Without Mutual Authentication - per TD0670 & TD0790 (NDcPP22e:FCS_TLSC_EXT.1) ... 60
2.3 Identification and authentication (FIA) ... 70
2.3.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1) ... 70
2.3.2 Password Management - per TD0792 (NDcPP22e:FIA_PMG_EXT.1) ... 72
2.3.3 Pre-Shared Key Composition (MACSEC10:FIA_PSK_EXT.1) ... 74
2.3.4 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7) ... 75
2.3.5 Password-based Authentication Mechanism (NDcPP22e:FIA_UAU_EXT.2) ... 76
2.3.6 User Identification and Authentication (NDcPP22e:FIA_UIA_EXT.1) ... 77
2.3.7 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/Rev) ... 80
2.3.8 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2) ... 85
2.4 Security management (FMT) ... 87
2.4.1 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate) ... 87
2.4.2 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData) ... 88
2.4.3 Management of TSF Data (NDcPP22e:FMT_MTD.1/CryptoKeys) ... 90
2.4.4 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1) ... 91
2.4.5 Specification of Management Functions (MACsec) - per TD0748 (MACSEC10:FMT_SMF.1/MACSEC) 93
2.4.6 Restrictions on Security Roles (NDcPP22e:FMT_SMR.2) ... 95
2.5 Protection of the TSF (FPT) ... 96
2.5.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1) ... 97
2.5.2 Protection of CAK Data (MACSEC10:FPT_CAK_EXT.1) ... 97
2.5.3 Failure with Preservation of Secure State - per TD0816 (MACSEC10:FPT_FLS.1) ... 98
2.5.4 Replay Detection - per TD0746 (MACSEC10:FPT_RPL.1) ... 99
2.5.5 Replay Detection for XPN - per TD0728 (MACSEC10:FPT_RPL_EXT.1) ... 101
2.5.6 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1) ... 103
2.5.7 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1) ... 103
2.5.8 TSF testing (NDcPP22e:FPT_TST_EXT.1) ... 105
2.5.9 Trusted update (NDcPP22e:FPT_TUD_EXT.1) ... 108
2.6 TOE access (FTA) ... 113
2.6.1 TSF-initiated Termination (NDcPP22e:FTA_SSL.3) ... 113

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.6.2 User-initiated Termination (NDcPP22e:FTA_SSL.4) 114
2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1) 115
2.6.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1) 117
2.7 Trusted path/channels (FTP) 118
2.7.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1) 118
2.7.2 Inter-TSF Trusted Channel (MACsec Communications) (MACSEC10:FTP_ITC.1/MACSEC) 120
2.7.3 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin) 121
3. Protection Profile SAR Assurance Activities 123
3.1 Development (ADV) 123
3.1.1 Basic Functional Specification (ADV_FSP.1) 123
3.2 Guidance documents (AGD) 124
3.2.1 Operational User Guidance (AGD_OPE.1) 124
3.2.2 Preparative Procedures (AGD_PRE.1) 126
3.3 Life-cycle support (ALC) 127
3.3.1 Labelling of the TOE (ALC_CMC.1) 127
3.3.2 TOE CM Coverage (ALC_CMS.1) 128
3.4 Tests (ATE) 128
3.4.1 Independent Testing - Conformance (ATE_IND.1) 128
3.5 Vulnerability assessment (AVA) 131
3.5.1 Vulnerability Survey (AVA_VAN.1) 132
3.5.2 Additional Flaw Hypothesis (AVA_VAN.1) 134

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

1. Introduction

This document presents evaluations results of the Cisco Catalyst 9400X/9600X Series Switches 17.12 NDcPP22e/MACSEC10 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

This section explains why the test subset was adequate to address all product installations.

1.1.1 Evaluated Platform Equivalence

The TOE is the Cisco Catalyst 9400X/9600X Series Switches all running Internetworking Operating System (IOS)-XE 17.12.

- Catalyst 9400X models:
  - Chassis: C9404R, C9407R, and C9410R
  - With the following Supervisor models: C9400X-SUP-2, C9400X-SUP-2XL
  - With the following Line Card models: C9400-LC-48HX, C9400-LC-48XS
- Catalyst 9600X models:
  - Chassis: **C9606R**
  - With the following Supervisor models: **C9600X-SUP2**
  - With the following Line Card models: **C9600-LC-40YL4CD**, **C9600X-LC-32CD**

The evaluator performed full testing on the following platforms (**bolded** in the lists above) and microarchitectures.

- C9606R – Intel Xeon D-1573N (Broadwell)

The evaluator additionally performed MACsec only testing on the following platform:

- C9407R – Intel Xeon D-1548 (Broadwell)

There is just one software image that is used across all of these devices in the 9400X and 9600X series. Therefore, the software image is exactly the same on every single model in the evaluated configuration. All Security Functionality (with the exception of MACsec encryption) is implemented in the one software image, which leverages the IOS Common Cryptographic Module (IC2M) and CiscoSSL FIPS Object Module (FOM). The TOE implements SSH and MACsec in IC2M, and the TOE implements TLSv1.2 in CiscoSSL.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The Catalyst 9400X supports MACsec encryption using the Unified Access Data Plane (UADP), which is a proprietary Application-Specific Integrated Circuit (ASIC). The MACsec Controller (MSC) is embedded in the ASICs that are utilized in the 9400X supervisor engine.

The Catalyst 9600X supports MACsec encryption using the CDR5M PHY embedded within the line cards. The CDR5M PHY uses the Marvell Alaska C 88X7121M MACsec engine.

While there are different models, they differ primarily in physical form factor, number and types of connections and slots, and relative performance. These differing characteristics primarily affect only non-TSF relevant functionality (such as throughput, processing speed, number and type of network connections supported, number of concurrent connections supported, and amount of storage). All TOE security functions (with the exception of MACsec encryption) are implemented in software, and the TOE security behavior is the same on all the devices for each of the SFRs defined by the Security Target. These SFRs are instantiated by the same version of the TOE software and in the same way on every platform.

MACsec encryption is implemented in hardware as mentioned above. The MACsec controller used in each TOE model family is unique, however each model in each family uses the same MACsec controller. All supervisor cards in the 9400X family uses the UADP embedded in the 9400X's supervisor engine, and all line cards in the 9600X model uses the CDR5M PHY. There exists one difference between the 9400X and 9600x in security functionality, the 9400X does not support XPN and therefore FPT_RPL_EXT.1 does not apply to the 9400X family. By testing both the 9400x and 9600x for MACsec, the evaluation lab has determined that the models chosen for testing covers all claimed models.

Since the 9400X and 9600X utilize the same software image, the evaluator tested the NDcPP22e requirements only against the 9600X which is sufficient to cover both the 9400X and 9600X. For Macsec testing, the evaluator ran the tests fully on both devices because of the difference in MACsec controller hardware. The full results for the 9600X are presented in this report. The MACsec testing results for the 9400X are presented in the supplemental DTR submitted with the checkout package, **Supplemental MACsec Detailed Test Report for Cisco Catalyst 9400X/9600X Series Switches 17.12**. The results for MACsec testing across the two devices are extremely similar, and are presented in the exact same way. The only notable difference in testing is that the 9400X does not claim support for extended packet number, and as such does not claim FPT_RPL_EXT.1.

1.1.2 CAVP Equivalence

The TOE provides cryptographic functions to implement TLS, SSH, and MACsec protocols. The cryptographic algorithm implementation has been validated for CAVP conformance. This includes key generation and random bit generation, key establishment methods, key destruction, and the various types of cryptographic operations to provide AES encryption/decryption, signature verification, hash generation, and keyed hash generation.

All algorithms claimed have Cryptographic Algorithm Validation Program (CAVP) certificates running on the processors specified in the table below.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Hardware Model	Processor
Catalyst 9400X Series	Intel Xeon D-1548 (Broadwell)
Catalyst 9600X Series	Intel Xeon D-1573N (Broadwell)

All cryptography is implemented using the IOS Common Cryptographic Module (IC2M) and CiscoSSL FOM cryptographic modules. IC2M applies to SSH and MACsec and CiscoSSL FOM applies to TLS 1.2.

The Catalyst 9400X Hardware Models support MACsec using the proprietary Unified Access Data Plane (UADP) Application-Specific Integrated Circuit (ASIC) (CAVP Cert. #4769). The MACsec Controller (MSC) is embedded within the ASICs that are utilized within Cisco the 9400X Supervisor engine.

The Catalyst 9600X Hardware Models support MACsec using the CDR5M PHY embedded within the Line Cards. The CDR5M PHY uses the 400G MACsec Engine on Marvell Alaska C PHYs version X7121M (CAVP Cert. #A1929).

The following algorithm implementations have been CAVP certified.

SFR	Selection	Algorithm	Implementation	Certificate Number
FCS_CKM.1 – Cryptographic Key Generation (RSA & ECDSA) and Key Verification (ECDSA only)	2048 3072	RSA	IC2M	A1462
	P-256 P-384	ECDSA	IC2M	A1462
	P-256 P-384 P-521	ECDSA	CiscoSSL FOM	A1420
FCS_CKM.2 – Cryptographic Key Establishment	P-256 P-384	KAS ECC	IC2M	A1462
	P-256 P-384 P-521	KAS ECC	CiscoSSL FOM	A1420
FCS_COP.1/DataEncryption – AES Data Encryption/Decryption	AES-CBC-128 AES-CBC-256 AES-GCM-128 AES-GCM-256	AES	IC2M	A1462
	AES-GCM-128 AES-GCM-256	AES	CiscoSSL FOM	A1420
FCS_COP.1/MACSEC	AES-GCM-128 AES-GCM-256	AES	UADP MSC	AES 4769 [1]
			CDR5M PHY	A1929 [2]
	AES-KW 128 bits	AES	IC2M	A1462

[1] The Tested Environment is Synopsys VCS v2011.12mx-SP1-3
[2] The Tested Environment is Synopsys VCS version R-2020.12-SP2-0_Full64

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

SFR	Selection	Algorithm	Implementation	Certificate Number
FCS_COP.1/SigGen – Cryptographic Operation (Signature Generation and Verification)	2048 3072	RSA	IC2M	A1462
	2048 3072	RSA	CiscoSSL FOM	A1420
FCS_COP.1/Hash – Cryptographic Operation (Hash Algorithm)	SHA-256 SHA-512	SHS	IC2M	A1462
	SHA-256 SHA-384	SHS	CiscoSSL FOM	A1420
FCS_COP.1/KeyedHash – Cryptographic Operation (Keyed Hash Algorithm)	HMAC-SHA-256	HMAC	IC2M	A1462
FCS_COP.1/CMAC	AES-CMAC 128, 256 bits	AES-CMAC	IC2M	A1462
FCS_RBG_EXT.1– Random Bit Generation	CTR_DRBG (AES) 256 bits	DRBG	IC2M	A1462
	CTR_DRBG (AES) 256 bits	DRBG	CiscoSSL FOM	A1420

All processors and microarchitectures were addressed as part of the evaluation testing. The algorithm testing for the Broadwell processors matches the tested platforms. Both processor/microarchitecture combinations are tested fully.

1.2 References

The following evidence was used to complete the Assurance Activities:

- Cisco Catalyst 9400X/9600X Series Switches 17.12 Security Target, version 0.6, September 17, 2024 (**ST**)
- Cisco Catalyst 9400X/9600X Series Switches 17.12 CC Configuration Guide, Version 0.4, September 17, 2024 (**Admin Guide**)

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profiles and describes the findings in each case.

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For the administrative task of generating/import of, changing, or deleting of cryptographic keys as defined in FAU_GEN.1.1c, the TSS should identify what information is logged to identify the relevant key.

For distributed TOEs the evaluator shall examine the TSS to ensure that it describes which of the overall required auditable events defined in FAU_GEN.1.1 are generated and recorded by which TOE components. The evaluator shall ensure that this mapping of audit events to TOE components accounts for, and is consistent with, information provided in Table 1, as well as events in Tables 2, 4, and 5 (where applicable to the overall TOE). This includes that the evaluator shall confirm that all components defined as generating audit information for a particular SFR should also contribute to that SFR as defined in the mapping of SFRs to TOE components, and that the audit records generated by each component cover all the SFRs that it implements.

Section 6 (FAU_GEN.1) in the ST states that each of the events specified in the audit record is in enough detail to identify the user for which the event is associated, when the event occurred, where the event occurred, the

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

outcome of the event, and the type of event that occurred such as generating keys, including the key identifier. Additionally, the start-up and shut-down of the audit functionality is audited.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation and ensure that it provides an example of each auditable event required by FAU_GEN.1 (i.e. at least one instance of each auditable event, comprising the mandatory, optional and selection-based SFR sections as applicable, shall be provided from the actual audit record).

The evaluator shall also make a determination of the administrative actions related to TSF data related to configuration changes. The evaluator shall examine the guidance documentation and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the cPP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are related to TSF data related to configuration changes. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

Section “Auditing” in the **Admin Guide** provides a table identifying all auditable events required by FAU_GEN.1 and provides examples of each audit record for each SFR (where applicable). The content of these audit examples match the content of audits generated during testing as recorded in the DTR.

From a review of the ST, the Guidance and through testing the evaluator also determined that the guidance contains all of the administrative actions and their associated audit events that are relevant to the PP and to use of the TOE. These administrative actions are consistent with the security requirements implemented in the TOE and were found to have appropriate management capabilities identified in the guidance documentation.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of auditable events to TOE components in the Security Target. For all events involving more than one TOE component when an audit event is triggered, the evaluator has to check that the event has been audited on both sides (e.g. failure of building up a secure communication channel between the two components). This is not limited to error

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

cases but includes also events about successful actions like successful build up/tear down of a secure communication channel between TOE components.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly.

The evaluator created a list of the required audit events. The evaluator then collected the audit event when running the other security functional tests described by the protection profiles. For example, the required event for FPT_STM.1 is Changes to Time. The evaluator collected these audit records when modifying the clock using administrative commands. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.2 Audit Data Generation (MACsec) (MACSEC10:FAU_GEN.1/MACSEC)

2.1.2.1 MACSEC10:FAU_GEN.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 MACSEC10:FAU_GEN.1.2/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.1.3 User identity association (NDcPP22e:FAU_GEN.2)

2.1.3.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

See NDcPP2e:FAU_GEN.1

Component Guidance Assurance Activities: The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU_GEN.1.

See NDcPP2e:FAU_GEN.1

Component Testing Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

For distributed TOEs the evaluator shall verify that where auditable events are instigated by another component, the component that records the event associates the event with the identity of the instigator. The evaluator shall perform at least one test on one component where another component instigates an auditable event. The evaluator shall verify that the event is recorded by the component as expected and the event is associated with the instigating component. It is assumed that an event instigated by another component can at least be generated for building up a secure channel between two TOE components. If for some reason (could be e.g. TSS or Guidance Documentation) the evaluator would come to the conclusion that the overall TOE does not generate any events instigated by other components, then this requirement shall be omitted.

See NDcPP2e:FAU_GEN.1

2.1.4 Protected Audit Event Storage (NDcPP22e:FAU_STG_EXT.1)

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.1.4.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it describes the means by which the audit data are transferred to the external audit server, and how the trusted channel is provided.

The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally; what happens when the local audit data store is full; and how these records are protected against unauthorized access.

The evaluator shall examine the TSS to ensure it describes whether the TOE is a standalone TOE that stores audit data locally or a distributed TOE that stores audit data locally on each TOE component or a distributed TOE that contains TOE components that cannot store audit data locally on themselves but need to transfer audit data to other TOE components that can store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs it contains a list of TOE components that store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs that contain components which do not store audit data locally but transmit their generated audit data to other components it contains a mapping between the transmitting and storing TOE components.

The evaluator shall examine the TSS to ensure that it details the behavior of the TOE when the storage space for audit data is full. When the option 'overwrite previous audit record' is selected this description should include an outline of the rule for overwriting audit data. If 'other actions' are chosen such as sending the new audit data to an external IT entity, then the related behaviour of the TOE shall also be detailed in the TSS.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator shall examine the TSS to ensure that it details whether the transmission of audit information to an external IT entity can be done in real-time or periodically. In case the TOE does not perform transmission in real-time the evaluator needs to verify that the TSS provides details about what event stimulates the transmission to be made as well as the possible acceptable frequency for the transfer of audit data.

For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how audit data transfer to the external audit server is implemented among the different TOE components (e.g. every TOE components does its own transfer or the data is sent to another TOE component for central transfer of all audit events to the external audit server).

For distributed TOEs the evaluator shall examine the TSS to ensure it describes which TOE components are storing audit information locally and which components are buffering audit information and forwarding the information to another TOE component for local storage. For every component the TSS shall describe the behaviour when local storage space or buffer space is exhausted.

Section 6 (FAU_STG_EXT.1) in the ST indicates that the TOE is a standalone device that stores audit data locally. The TOE is not distributed. The TOE can be configured to export syslog records to a specified, external syslog server in real-time. The TOE protects communications with an external syslog server using TLS.

The size of the logging files on the TOE is configurable by the Administrator with the minimum value being 4096 (default) to 2,147,483,647 bytes of available disk space. For audit records stored internally to the TOE, the audit records are stored in a circular log file where the TOE overwrites the oldest audit records when the audit trail becomes full. Only Authorized Administrators can clear the local logs, and local audit records are stored in a directory that does not allow Administrators to modify the contents.

Component Guidance Assurance Activities: The evaluator shall also examine the guidance documentation to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

The evaluator shall also examine the guidance documentation to determine that it describes the relationship between the local audit data and the audit data that are sent to the audit log server. For example, when an audit event is generated, is it simultaneously sent to the external server and the local store, or is the local store used as a buffer and 'cleared' periodically by sending the data to the audit server.

The evaluator shall also ensure that the guidance documentation describes all possible configuration options for FAU_STG_EXT.1.3 and the resulting behaviour of the TOE for each possible configuration. The description of possible configuration options and resulting behaviour shall correspond to those described in the TSS.

Section “TLS - Syslog” and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog, how to configure the supported TLS ciphersuites (consistent with the requirements in the ST), how to configure and

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

authenticate the Root and Intermediate Trustpoint CA certificates and how to configure the reference identifier for the peer.

Section “Enable Remote Syslog Server” in the **Admin Guide** provides instructions for using the “logging host” command to enable the TOE to transmit audit data to a specified external audit server using the configured TLS profile. This section states that when an audit event is generated, it is simultaneously sent to the external server and the local store.

Section “Configure Local Logging Buffer Size” in the **Admin Guide** provides instructions for how to configure the size of the local logging buffer. This section states that if the local storage space for audit data is full the TOE will overwrite the oldest audit record to make room for the new audit record.

Component Testing Assurance Activities: Testing of the trusted channel mechanism for audit will be performed as specified in the associated assurance activities for the particular trusted channel mechanism. The evaluator shall perform the following additional test for this requirement:

a) Test 1: The evaluator shall establish a session between the TOE and the audit server according to the configuration guidance provided. The evaluator shall then examine the traffic that passes between the audit server and the TOE during several activities of the evaluator's choice designed to generate audit data to be transferred to the audit server. The evaluator shall observe that these data are not able to be viewed in the clear during this transfer, and that they are successfully received by the audit server. The evaluator shall record the particular software (name, version) used on the audit server during testing. The evaluator shall verify that the TOE is capable of transferring audit data to an external audit server automatically without administrator intervention.

b) Test 2: The evaluator shall perform operations that generate audit data and verify that this data is stored locally. The evaluator shall perform operations that generate audit data until the local storage space is exceeded and verifies that the TOE complies with the behaviour defined in FAU_STG_EXT.1.3. Depending on the configuration this means that the evaluator has to check the content of the audit data when the audit data is just filled to the maximum and then verifies that

1) The audit data remains unchanged with every new auditable event that should be tracked but that the audit data is recorded again after the local storage for audit data is cleared (for the option 'drop new audit data' in FAU_STG_EXT.1.3).

2) The existing audit data is overwritten with every new auditable event that should be tracked according to the specified rule (for the option 'overwrite previous audit records' in FAU_STG_EXT.1.3)

3) The TOE behaves as specified (for the option 'other action' in FAU_STG_EXT.1.3).

c) Test 3: If the TOE complies with FAU_STG_EXT.2/LocSpace the evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2/LocSpace are correct when performing the tests for FAU_STG_EXT.1.3.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

d) Test 4: For distributed TOEs, Test 1 defined above should be applicable to all TOE components that forward audit data to an external audit server. For the local storage according to FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3 the Test 2 specified above shall be applied to all TOE components that store audit data locally. For all TOE components that store audit data locally and comply with FAU_STG_EXT.2/LocSpace Test 3 specified above shall be applied. The evaluator shall verify that the transfer of audit data to an external audit server is implemented.

Test 1: The successful establishment of the TLS syslog connection was demonstrated in FTP_ITC.1. In each case the TOE initiated the connection without administrator intervention. The use of TLS ensured that no audits were viewed in cleartext. The audits collected as part of FAU_GEN.1 throughout testing were gathered from the remote syslog server running rsyslog version 8.16.0 thus demonstrating that audits were successfully received by the remote syslog server.

Test 2: The option "overwrite previous audit records" is selected in FAU_STG_EXT.1.3. The evaluator issued the 'show logging' command in order to view the contents of the TOE's log buffer. Next the evaluator performed commands in order to generate audit activity and verify that audit records are overwritten when the log buffer is at full capacity. The evaluator issued the "show logging", "show configuration" and "show fips status" commands. The evaluator then issued the "show logging" command and reviewed the TOE log buffer again. The evaluator observed that new audit records were generated including those associated with the commands performed. A comparison of the logs confirmed that the oldest logs below were overwritten.

Test 3: Not applicable. The TOE does not claim support for FAU_STG_EXT.2/LocSpace.

Test 4: Not applicable. The TOE is not a distributed TOE.

2.2 Cryptographic Support (FCS)

2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.2.1.1 NDcPP22e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Section 6 (FCS_CKM.1) in the ST provides tables which identify RSA and ECC schemes that meet FIPS PUB 186-4 for asymmetric key generation. The TOE generates RSA keys of size 2048 or 3072 bits in support of device authentication for SSH remote administration and for TLS sessions. The TOE also generates ECC NIST curves p-256, and p-384 for key establishment with SSH remote administration and generates ECC NIST curves p-256, p-384, and p-521 in support of key establishment for TLS sessions.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all cryptographic protocols defined in the Security Target.

Section "FIPS Mode" in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

The following sections in the **Admin Guide** provide further guidance for configuring SSH and TLS with the supported key generation schemes and key sizes.

- Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for generating an RSA **2048** or **3072 bit** key for SSH. It also describes how to configure ecdh-sha2-nistp256 and ecdh-sha2-nistp384 as SSH key exchange methods

- Section "TLS - Syslog" and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog and how to configure the supported TLS ciphersuites (consistent with the requirements in the ST). The supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS ciphersuites. The TOE presents the Supported Elliptic Curves Extension with NIST curves secp256r1, secp384r1, and secp521r1 in the Client Hello. This behavior is performed by default and there is no security management function to disable it.

Component Testing Assurance Activities: 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.

Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

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:

a) Random Primes:

- Provable primes

- Probable primes

b) Primes with Conditions:

- Primes p1, p2, q1, q2, p and q shall all be provable primes

- Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

- Primes p1, p2, q1, q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

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:

- 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:

- 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:

- len(q) bit output of RBG where 1 <=x <= q-1

- len(q) + 64 bit output of RBG, followed by a mod q-1 operation and a +1 operation, where 1<= x<=q-1.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set.

To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set.

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

Testing for FFC Schemes using safe-prime groups is done as part of testing in CKM.2.1.

(TD0580 applied)

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.2.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. It is sufficient to provide the scheme, SFR, and service in the TSS.

The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be:

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	Administration
ECDH	FCS_SSHC_EXT.1	Audit Server
ECDH	FCS_IPSEC_EXT.1	Authentication Server

The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

(TD0580 applied)

Section 6 (FCS_CKM.2) in the ST provides tables which identify ECC schemes that the TOE implements to perform cryptographic key establishment. For key establishment in SSH remote administration sessions, the TOE generates

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

ECC NIST curves p-256, and p-384 for EC-DH key establishment that meets NIST SP 800-56A, Revision 3. For key establishment in TLS sessions (with the audit server), the TOE generates ECC NIST curves p-256, p-384, and p-521 for EC-DH key establishment that meets NIST SP 800-56A, Revision 3.

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

See FCS_CKM.1 where this activity has been performed.

Component Testing Assurance Activities: Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

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

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

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

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

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the other info and TOE id fields.

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

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

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses RSAES-PKCS1-v1_5.

FFC Schemes using 'safe-prime' groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

(TD0580 applied)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.3.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for2). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

The evaluator shall check to ensure the TSS identifies how the TOE destroys keys stored as plaintext in non-volatile memory, and that the description includes identification and description of the interfaces that the TOE uses to destroy keys (e.g., file system APIs, key store APIs).

Note that where selections involve 'destruction of reference' (for volatile memory) or 'invocation of an interface' (for non-volatile memory) then the relevant interface definition is examined by the evaluator to ensure that the interface supports the selection(s) and description in the TSS. In the case of non-volatile memory the evaluator includes in their examination the relevant interface description for each media type on which plaintext keys are stored. The presence of OS-level and storage device-level swap and cache files is not examined in the current version of the Evaluation Activity.

Where the TSS identifies keys that are stored in a non-plaintext form, the evaluator shall check that the TSS identifies the encryption method and the key-encrypting-key used, and that the key-encrypting-key is either itself stored in an encrypted form or that it is destroyed by a method included under FCS_CKM.4.

The evaluator shall check that the TSS identifies any configurations or circumstances that may not conform to the key destruction requirement (see further discussion in the Guidance Documentation section below). Note that reference may be made to the Guidance Documentation for description of the detail of such cases where destruction may be prevented or delayed.

Where the ST specifies the use of 'a value that does not contain any CSP' to overwrite keys, the evaluator examines the TSS to ensure that it describes how that pattern is obtained and used, and that this justifies the claim that the pattern does not contain any CSPs.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Section 6 (FCS_CKM.4) in the ST indicates that the TOE meets all requirements specified in FIPS 140-2 for destruction of keys and Critical Security Parameters (CSPs) when no longer required for use. The TSS does not identify any configurations or circumstances that may not conform to the key destruction requirement.

Section 6.1 in the ST provides a table which further describes that the TOE zeroizes all secrets, keys, and associated values when they are no longer required. The table identifies all the relevant secret and private keys, their storage location and how and when they are zeroized.

Component Guidance Assurance Activities: A TOE may be subject to situations that could prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and any other supporting information used). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command [Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table)] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

Section “Zeroize Private Key” in the **Admin Guide** provides instructions for zeroizing private keys stored in NVRAM that are generated for SSH using the “crypto key zeroize” command. Other keys stored in SDRAM are zeroized when no longer in use, zeroized with a new value of the key, or zeroized on power-cycle. The Guide does not identify any configurations or circumstances that do not strictly conform to the key destruction requirements or any situation where key destruction may be delayed at the physical layer.

Component Testing Assurance Activities: None Defined

2.2.4 Cryptographic Operation (AES-CMAC Keyed Hash Algorithm) (MACSEC10:FCS_COP.1/CMAC)

2.2.4.1 MACSEC10:FCS_COP.1.1/CMAC

TSS Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it specifies the following values used by the AES-CMAC function: key length, hash function used, block size, and output MAC length used.

Section 6 (FCS_COP.1/CMAC) in the ST states that the TOE implements keyed-hash message authentication in accordance with AES-CMAC and cryptographic key sizes 128 and 256 bits with message digest size of 128 bits, block size of 128 bits, and MAC length of 128 bits which meets NIST SP 800-38B.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Test 1: CMAC Generation Test

To test the generation capability of AES-CMAC, the evaluator shall provide to the TSF, for each key length-message length-CMAC length tuple (in bytes), a set of eight arbitrary key-plaintext tuples that will result in the generation of a known MAC value when encrypted. The evaluator shall then verify that the correct MAC was generated in each case.

Test2: CMAC Verification Test

To test the generation capability of AES-CMAC, the evaluator shall provide to the TSF, for each key length-message length-CMAC length tuple (in bytes), a set of 20 arbitrary key-MAC tuples that will result in the generation of known messages when verified. The evaluator shall then verify that the correct message was generated in each case.

The following information should be used by the evaluator to determine the key length-message length-CMAC length tuples that should be tested:

- Key length: values will include the following:

o 16

o 32

- Message length: values will include the following:

o 0 (optional)

o Largest value supported by the implementation (no greater than 65536)

o Two values divisible by 16

o Two values not divisible by 16

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

- CMAC length

o Smallest value supported by the implementation (no less than 1)

o 16

o Any supported CMAC length between the minimum and maximum values

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.5 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS_COP.1/DataEncryption)

2.2.5.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it identifies the key size(s) and mode(s) supported by the TOE for data encryption/decryption.

Section 6 (FCS_COP.1/DataEncryption) in the ST states the TOE provides symmetric encryption and decryption capabilities using AES in CBC mode (SSH only) and GCM mode (128 and 256 bits) as described in ISO/IEC 18033-3, ISO/IEC 10116, and ISO/IEC 19772. AES is implemented in the SSH and TLS protocols.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected mode(s) and key size(s) defined in the Security Target supported by the TOE for data encryption/decryption.

Section "FIPS Mode" in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH: aes256-cbc, aes128-cbc, aes128-gcm@openssh.com, aes256-gcm@openssh.com.

Section "TLS - Syslog" and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog, how to

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

configure the supported TLS ciphersuites (consistent with the requirements in the ST) and how to configure and authenticate the Root and Intermediate Trustpoint CA certificates.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

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

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

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.

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

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.

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

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

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

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

```
# Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

CT[1] = AES-CBC-Encrypt(Key, IV, PT)

PT = IV

else:

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

PT = CT[i-1]
```

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

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

AES-GCM Test

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

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

128 bit and 256 bit keys

a) 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.

a) 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.

b) Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

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

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

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

AES-CTR Known Answer Tests

The Counter (CTR) mode is a confidentiality mode that features the application of the forward cipher to a set of input blocks, called counters, to produce a sequence of output blocks that are exclusive-ORed with the plaintext to produce the ciphertext, and vice versa. Due to the fact that Counter Mode does not specify the counter that is used, it is not possible to implement an automated test for this mode. The generation and management of the counter is tested through FCS_SSH*_EXT.1.4. If CBC and/or GCM are selected in FCS_COP.1/DataEncryption, the test activities for those modes sufficiently demonstrate the correctness of the AES algorithm. If CTR is the only selection in FCS_COP.1/DataEncryption, the AES-CBC Known Answer Test, AES-GCM Known Answer Test, or the following test shall be performed (all of these tests demonstrate the correctness of the AES algorithm):

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

KAT-1 To test the encrypt functionality, the evaluator shall supply a set of 5 plaintext values for each selected keysize and obtain the ciphertext value that results from encryption of the given plaintext using a key value of all zeros.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

KAT-2 To test the encrypt functionality, the evaluator shall supply a set of 5 key values for each selected keysize and obtain the ciphertext value that results from encryption of an all zeros plaintext using the given key value.

KAT-3 To test the encrypt functionality, the evaluator shall supply a set of key values for each selected keysize as described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values. A set of 128 128-bit keys, a set of 192 192-bit keys, and/or a set of 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].

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384 ciphertext values will be generated if all keysizes are selected). 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].

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10 (test shall be performed using AES-ECB mode). For each i the evaluator shall choose a key 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 using a known good implementation. The evaluator shall perform this test using each selected keysize.

AES-CTR Monte-Carlo Test

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

# 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. The evaluator shall perform this test using each selected keysize.

There is no need to test the decryption engine.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.6 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash)

2.2.6.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Section 6 (FCS_COP.1/Hash) in the ST states that the TOE provides cryptographic hashing services using SHA-256, SHA-384, and SHA-512 as specified in ISO/IEC 10118-3:2004 (with key sizes and message digest sizes of 256, 384, and 512 bits respectively).

The TOE provides keyed-hashing message authentication services using HMAC-SHA-256 that operates on 512-bit blocks with key size and message digest size of 256 bits bits as specified in ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”.

SHA-512 hashing is used for verification of software image integrity.

Section 6 (FCS_SSHS_EXT.1) in the ST indicates that SSH key exchange implementation supports the following key exchange algorithm:

ecdh-sha2-nistp256

ecdh-sha2-nistp384

Section 6 (FCS_TLSC_EXT.1) in the ST provides the list of supported TLS ciphersuites which implement either SHA256 or SHA384.

Section 6 (FPT_APW_EXT.1) in the ST states that all passwords are stored using a SHA-2 hash.

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

Section “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The following sections in the **Admin Guide** provide further guidance for configuring SSH and TLS with the supported hash algorithms.

- Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for configuring the sha hashes as part of the supported key exchange algorithm: ecdh-sha2-nistp256,ecdh-sha2-nistp384, the supported MAC algorithms: hmac-**sha2-256**, and the supported host key algorithms: rsa-**sha2-256**, rsa-**sha2-512**. Thus, it is clear that the configuration of these algorithms includes configuration of the supported hash sizes.

- Section "TLS - Syslog" and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog and how to configure the supported TLS ciphersuites (consistent with the requirements in the ST). The supported cryptographic algorithms and key strengths are configured implicitly by defining the supported TLS ciphersuites. The hash/keyed hash algorithms are specified in the TLS ciphersuite. Table 6 in the **Admin Guide** maps the TLS ciphersuite configuration options to the TLS ciphersuite and in each case indicates whether **SHA256** and/or **SHA384** will be configured.

Component Testing Assurance Activities: 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 shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test - Bit-oriented Mode

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

Short Messages Test - Byte-oriented Mode

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

Selected Long Messages Test - Bit-oriented Mode

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 99*i, where 1 <= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.7 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS_COP.1/KeyedHash)

2.2.7.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

Section 6 (FCS_COP.1/KeyedHash) in the ST states that the TOE provides keyed-hashing message authentication services using HMAC-SHA-256 that operates on 512-bit blocks with key size and message digest size of 256 bits as specified in ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the values used by the HMAC function: key length, hash function

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

used, block size, and output MAC length used defined in the Security Target supported by the TOE for keyed hash function.

Section “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

The following sections in the **Admin Guide** provide further guidance for configuring SSH and TLS with the supported keyed hash algorithms.

- Section “SSH Remote Administration Protocol” in the **Admin Guide** provides instructions for configuring the supported MAC algorithms: **hmac-sha2-256**.

Component Testing Assurance Activities: For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and message data using a known good implementation.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.8 Cryptographic Operation (MACsec AES Data Encryption and Decryption) - per TD0728 (MACSEC10:FCS_COP.1/MACSEC)

2.2.8.1 MACSEC10:FCS_COP.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the supported AES modes that are required for this PP-Module in addition to the ones already required by the NDcPP in FCS_COP.1/DataEncryption.

Section 6 (FCS_COP.1/MACSEC) of the ST states the TOE implements keyed-hash message authentication in accordance with AES-CMAC and cryptographic key sizes 128 and 256 bits with message digest size of 128 bits, block size of 128 bits, and MAC length of 128 bits which meets NIST SP 800-38B.

The TSF implements symmetric encryption and decryption capabilities using AES GCM mode (128 and 256 bits) as described in ISO/IEC 18033-3 and ISO/IEC 19772. The TSF implements AES Key Wrap with a key size of 128 bits as specified in NIST SP800-38F.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

AES is implemented in the MACsec protocol.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform testing for AES-GCM as required by the NDcPP in FCS_COP.1/DataEncryption.

In addition to the tests specified in the NDcPP for other iterations of FCS_COP.1, the evaluator shall perform the following tests:

Test 3: KW-AE Test: To test the authenticated encryption capability of AES key warp (KW), the evaluator shall provide five sets of 100 messages and keys to the TOE for each key length supported by the TSF. Each set of messages and keys shall correspond to one of five plaintext message lengths (detailed below). The evaluator shall have the TSF encrypt the messages with the associated key. The evaluator shall verify that the correct ciphertext was generated in each case.

Test 4: KW-AD Test: To test the authenticated decryption capability of AES KW, the evaluator shall provide five sets of 100 ciphertext values and keys to the TOE for each key length supported by the TSF. Each set of ciphertexts and keys shall correspond to one of five plaintext message lengths (detailed below). For each set of 100 cyphertext values, 20 shall not be authentic (i.e. fail authentication). The evaluator shall have the TSF decrypt the ciphertext messages with the associated key. The evaluator will then verify the correct plaintext was generated or the failure to authenticate was correctly detected.

The messages in each set for both tests shall be the following lengths:

- two lengths that are non-zero multiples of 128 bits (two semiblock lengths)

- two that are odd multiples of the semiblock length (64 bits)

- the largest supported plaintext length less than or equal to 4096 bits

(TD0728 applied)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.9 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS_COP.1/SigGen)

2.2.9.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and key size supported by the TOE for signature services.

Section 6 (FCS_COP.1/SigGen) in the ST states that the TOE provides cryptographic signature services using a RSA Digital Signature Algorithm with key sizes of 2048 or 3072 bits as specified in FIPS PUB 186-4.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected cryptographic algorithm and key size defined in the Security Target supported by the TOE for signature services.

Section "FIPS Mode" in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography.

The following sections in the **Admin Guide** provide further guidance for configuring SSH and TLS such that RSA signature services are automatically configured.

- Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for generating an RSA 2048 or 3072 bit key for use with SSH remote administration. RSA signature services using 2048 or 3072 key sizes are automatically configured when SSH is configured as instructed in the steps above.
- Section "TLS – Syslog" and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog and how to configure the supported TLS ciphersuites (consistent with the requirements in the ST). RSA signature services using 2048 or 3072 key sizes are automatically configured when TLS is configured as instructed in the steps above.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

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

ECDSA FIPS 186-4 Signature Verification Test

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

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

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator generates or obtains 10 messages for each modulus size/SHA combination supported by the TOE. The TOE generates and returns the corresponding signatures.

The evaluator shall verify the correctness of the TOE's signature using a trusted reference implementation of the signature verification algorithm and the associated public keys to verify the signatures.

Signature Verification Test

For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (d, e). Each private key d is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, e, messages, or signatures are altered so that signature verification should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key e values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

The evaluator verifies that the TOE confirms correct signatures on the original messages and detects the errors introduced in the altered messages.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.10 MACsec (MACSEC10:FCS_MACSEC_EXT.1)

2.2.10.1 MACSEC10:FCS_MACSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.2 MACSEC10:FCS_MACSEC_EXT.1.2

TSS Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.3 MACSEC10:FCS_MACSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.4 MACSEC10:FCS_MACSEC_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the ability of the TSF to implement MACsec in accordance with IEEE 802.1AE-2018. The evaluator shall also determine that the TSS describes the ability of the TSF to derive SCI values from peer MAC address and port data and to reject traffic that does not have a valid SCI. Finally, the evaluator shall check the TSS for an assertion that only EAPOL and MACsec Ethernet frames, and MAC control frames are accepted by the MACsec interface.

Section 6 (FCS_MACSEC_EXT.1) of the ST states the TOE implements MACsec in compliance with Institute of Electrical and Electronics Engineers (IEEE) Standard 802.1AE-2018. The MACsec connections maintain confidentiality of transmitted data and takes measures against frames transmitted or modified by unauthorized devices.

The Secure Channel Identifier (SCI) is composed of a globally unique 48-bit Message Authentication Code (MAC) Address and the Secure System Address (port). The SCI is part of the SecTAG if the Secure Channel (SC) bit is set and will be at the end of the tag. Any MAC Protocol Data Units (MPDUs) during a given session that contain an SCI other than the one used to establish that session is rejected.

Only Extensible Authentication Protocol over LAN (EAPOL) (Physical Address Extension (PAE) EtherType 88-8E), MACsec frames (EtherType 88-E5), and MAC control frames (EtherType 88-08) are permitted. All others are rejected.

Component Guidance Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 5: The evaluator shall successfully establish a MACsec channel between the TOE and a MACsec-capable peer in the operational environment and verify that the TSF logs the communications. The evaluator shall capture the traffic between the TOE and the operational environment to determine the SCI that the TOE uses to identify the peer. The evaluator shall then configure a test system to capture traffic between the peer and the TOE to modify the SCI that is used to identify the peer. The evaluator then verifies that the TOE does not reply to this traffic and logs that the traffic was discarded.

Test 6: The evaluator shall send Ethernet traffic to the TOE's MAC address that iterates through the full range of supported EtherType values (refer to List of Documented EtherTypes) and observes that traffic for all EtherType values is discarded by the TOE except for the traffic which has an EtherType value of 88-8E, 88-E5, or 8808. Note that there are a large number of EtherType values so the evaluator is encouraged to execute a script that automatically iterates through each value.

Test 5: The evaluator configured MACsec between the TOE and a peer. The evaluator captured the SCI value that is used to negotiate the successful connection. The evaluator sent a MACsec encrypted ping packet from the peer to the TOE. The TOE responded with a MACsec reply, indicating that it accepted the peer's MACsec packet. The evaluator then sent a packet with an incorrect SCI value. The evaluator observed that the TOE rejected the incorrect packet and did not send a reply.

Test 6: The evaluator configured MACsec between the TOE and a peer. The evaluator then configured the peer test device to send Ethernet traffic that iterates through the range of supported EtherType values. The evaluator confirmed from packet captures that the TOE does not respond to any EtherTypes except for 88-8E or 88-E5. MACsec and EAPOL EtherTypes can be seen throughout other MACsec tests.

2.2.11 MACsec Integrity and Confidentiality (MACSEC10:FCS_MACSEC_EXT.2)

2.2.11.1 MACSEC10:FCS_MACSEC_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 MACSEC10:FCS_MACSEC_EXT.2.2

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.3 MACSEC10:FCS_MACSEC_EXT.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the methods that the TOE implements to provide assurance of MACsec integrity. This should include any confidentiality offsets used, the use of an ICV (including the supported length), and ICV generation with the SAK, using the SCI as the most significant bits of the initialization vector (IV) and the 32 least significant bits of the PN as the IV.

Section 6 (FCS_MACSEC_EXT.2) of the ST states that the TOE implements the MACsec requirement for integrity protection with the confidentiality offsets of 0, 30 and 50 using the 'mka-policy confidentiality-offset' command.

An offset value of 0 does not offset the encryption and offset values of 30 and 50 offset the encryption by 30 and 50 characters respectively.

An Integrity Check Value (ICV) of 16-bytes derived with the SAK is used to provide assurance of the integrity of MPDUs.

The TOE derives the ICK from a CAK using KDF, using the SCI as the most significant bits of the Initialization Vector (IV) and the 32 least significant bits of the PN as the IV.

Component Guidance Assurance Activities: If any integrity verifications are configurable, such as the confidentiality offsets used or the mechanism used to derive an ICK, the evaluator shall verify that instructions for performing these functions are documented.

Section "MACSEC and MKA Configuration" in the **Admin Guide** provides instructions for configuring the "confidentiality-offset" in the MKA policy. It also provides instructions for configuring the MACsec PSK which includes configuring the aes-cmac algorithm and the key-string. The key-string is the CAK that is used for ICV validation by the MKA protocol. The CAK is not used directly but derives two further keys from the CAK using the AES cipher in CMAC mode. The derived keys include the ICV Key (ICK) used to verify the integrity of MPDUs and to prove the transmitter of the MKPDU possesses the CAK.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 7: The evaluator shall transmit MACsec traffic to the TOE from a MACsec-capable peer in the operational environment. The evaluator shall verify via packet captures, audit logs, or both that the frame bytes after the MACsec Tag values in the received traffic is not obviously predictable.

Test 8: The evaluator shall transmit valid MACsec traffic to the TOE from a MACsec-capable peer in the operational environment that is routed through a test system set up as a man-in-the-middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure.

Test 7: A valid test MACsec connection was performed in MACsec10:FCS_MACSEC_EXT.1-t1. The evaluator viewed that none of the frame bytes after MACsec Tag value were obviously predictable, indicating that the data was successfully encrypted.

Test 8: The evaluator configured MACsec between the TOE and a peer. The evaluator attempted to send a MACsec encrypted ICMP packet in which the encrypted packet contained a modified byte in the end of the data payload. This effectively creates an invalid packet in which the ICV does not match the entire packet. The evaluator observed that the TOE rejected the incorrect packet and did not send a reply.

2.2.12 MACsec Randomness (MACSEC10:FCS_MACSEC_EXT.3)

2.2.12.1 MACSEC10:FCS_MACSEC_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.12.2 MACSEC10:FCS_MACSEC_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the method used to generate SAKs and nonces and that the strength of the CAK and the size of the CAK's key space are provided.

Section 6 (FCS_MACSEC_EXT.3) of the ST states that each SAK is generated using the KDF specified in IEEE 802.1X-2010 section 6.2.1 using the following transform - KS-nonce = a nonce of the same size as the required SAK, obtained from a Random Number Generator (RNG) each time an SAK is generated.

Each of the keys used by MKA is derived from the CAK.

The key string is the CAK that is used for ICV validation by the MKA protocol. The CAK is not used directly but derives two further keys from the CAK using the AES cipher in CMAC mode.

The derived keys are tied to the identity of the CAK, and thus restricted to use with that par-ticular CAK. These are the ICV Key (ICK) used to verify the integrity of MPDUs and to prove that the transmitter of the MKPDU possesses the CAK, and the Key Encrypting Key (KEK) used by the Key Server, elected by MKA, to transport a succession of SAKs, for use by MACsec, to the other member(s) of a CA.

The key size is 32-bit hexadecimal in length for AES 128-bit CMAC mode encryption and 64-bit hexidecimal in length for AES 256-bit CMAC mode encryption.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: Testing of the TOE's MACsec capabilities and verification of the deterministic random bit generator is sufficient to demonstrate that this SFR has been satisfied.

Please see test results for other MACsec test cases, which show the TOE's MACsec capabilities. Please see FCS_RBG_EXT.1 for verification of the DRBG.

2.2.13 MACsec Key Usage (MACSEC10:FCS_MACSEC_EXT.4)

2.2.13.1 MACSEC10:FCS_MACSEC_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.13.2 MACSEC10:FCS_MACSEC_EXT.4.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.3 MACSEC10:FCS_MACSEC_EXT.4.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.4 MACSEC10:FCS_MACSEC_EXT.4.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.5 MACSEC10:FCS_MACSEC_EXT.4.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the SAK is wrapped prior to being distributed using the AES implementation specified in this PP-Module.

Section 6 (FCS_MACSEC_EXT.4) of the ST states that MACsec peer authentication is achieved by only using pre-shared keys.

The SAKs are distributed between these peers using AES Key Wrap. Prior to distribution of the SAKs between these peers, the TOE uses AES Key Wrap in accordance with AES as specified in ISO/IEC 18033-3, AES in CMAC mode as specified in NIST SP800-38B, and GCM as specified in ISO/IEC 19772.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component Guidance Assurance Activities: If the method of peer authentication is configurable, the evaluator shall verify that the guidance provides instructions on how to configure this. The evaluator shall also verify that the method of specifying a lifetime for CAKs is described.

Section "MACSEC and MKA Configuration" in the **Admin Guide** provides instructions for configuring the MACsec PSK in a key chain and includes specifying the key-string and the lifetime for CAKs.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 9: For each supported method of peer authentication in FCS_MACSEC_EXT.4.1, the evaluator shall follow the operational guidance to configure the supported method (if applicable). The evaluator shall set up a packet sniffer between the TOE and a MACsec-capable peer in the operational environment. The evaluator shall then initiate a connection between the TOE and the peer such that authentication occurs and a secure connection is established. The evaluator shall wait one minute and then disconnect the TOE from the peer and stop the sniffer. The evaluator shall use the packet captures to verify that theSC was established via the selected mechanism and that the non-VLAN EtherType of the first data frame sent between the TOE and the peer is 88-E5.

Test 10: The evaluator shall capture traffic between the TOE and a MACsec-capable peer in the operational environment. The evaluator shall then cause the TOE to distribute a SAK to that peer, capture the MKPDUs from that operation, and verify the key is wrapped in the captured MKPDUs.

Test 9: The evaluator configured MACsec between the TOE and a peer. The evaluator started a packet capture and then stopped the packet capture after 1 minute. The evaluator verified that the first data frame sent between the TOE and the test peer has an EtherType value of 0x88E5. At the same time, the evaluator ensured that the TOE is the MKA key server by ensuring that the test peer's MKA priority is set to the maximum value (255) and that the TOE's MKA priority is set to a higher priority. The evaluator then analyzed the packet capture and verified that the TOE sends an MKPDU packet to the test peer that contains an AES wrapped SAK.

Test 10: This test was performed as part of Test 1.

2.2.14 MACsec Key Agreement - per TD0817 (MACSEC10:FCS_MKA_EXT.1)

2.2.14.1 MACSEC10:FCS_MKA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Testing Assurance Activities: None Defined

2.2.14.2 MACSEC10:FCS_MKA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.3 MACSEC10:FCS_MKA_EXT.1.3

TSS Assurance Activities: The evaluator shall examine the TSS to verify that it describes the methods that the TOE implements to provide assurance of MKA integrity, including the use of an ICV and the ability to use a KDF to derive an ICK.

Section 6 (FCS_MKA_EXT.1) of the ST states that for the Data Integrity Check, MACsec uses MKA to generate an ICV for the frame arriving on the port. If the generated ICV is the same as the ICV in the frame, then the frame is accepted; otherwise, it is dropped. The key string is the CAK that is used for ICV validation by the MKA protocol.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests:

Test 11: The evaluator shall transmit MKA traffic (MKPDUs) to the TOE from a MKA-capable peer in the operational environment. The evaluator shall verify via packet captures, audit logs, or both that the last 16 octets of the MKPDUs in the received traffic do not appear to be predictable.

Test 12: The evaluator shall transmit valid MKA traffic to the TOE from a MKA-capable peer in the operational environment that is routed through a test system set up as a man-in-the-middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure.

Test 11: A successful MACsec connection was established in MACSEC10:FCS_MACSEC_EXT.1 test 5 above. The evaluator observed the ICV in the MKPDU packets and determined they were not obviously predictable.

Test 12: The evaluator disabled data replay protection in the TOE in order to execute this test. The evaluator configured MACsec between the TOE and a peer. The evaluator captured a previously sent MKPDU packet and modified the last byte in the packet. The evaluator attempted to send the modified packet to the TOE. The evaluator observed the TOE successfully rejecting the packet and reporting an error in the audit log.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.14.4 MACSEC10:FCS_MKA_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The tests below require the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with. Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the key server and principal actor (peer). The evaluator shall then perform the following tests using a traffic sniffer to capture this traffic:

Test 13a: The evaluator shall configure the TOE to establish a MKA session with a new peer. The evaluator shall verify that the TOE sends a fresh SAK to the peer and sends other MKPDUs required for a new session. The evaluator shall verify from packet captures that MKPDUs are sent at least once every two seconds or every half-second, in accordance with the SFR selection.

Test 13b: (Conditional - If 'EAPTLS with DevIDs' is selected in FCS_MACSEC_EXT.4.1) The evaluator shall use EAP-TLS to derive a CAK and configure the TOE's peer to send '0' in the MKA parameter field for MACsec Capability (Table 11-6 in 802.1X-2020). The evaluator shall observe that the peer is deleted from the connection after MKA Life Time has passed.

Test 14a: (Conditional - if any 'group CAK' selection is made in FCS_MKA_EXT.1.5) The evaluator shall configure the TOE to send a fresh SAK with two peers as active participants. The evaluator shall verify that the TOE sends a fresh SAK to the peers and sends other MKPDUs required for a new session. The evaluator shall verify from packet captures that MKPDUs are sent at least once every half second in accordance with the MKA Bounded Hello Time.

Test 14b: (Conditional - if any 'group CAK' selection is made in FCS_MKA_EXT.1.5) Disconnect one of the peers. Arbitrarily introduce an artificial delay in sending a fresh SAK following the change in the Live Peer List. For this delayed fresh SAK, use a man-in-the-middle device to observe that the MKA Life Time of 6.0 seconds is enforced by the TSF.

(TD0817 applied, supersedes TD0787 and TD0805)

Test 13a: The TOE was configured with delay protection enabled. The evaluator set up two MACsec peers to connect to the TOE. The evaluator ensured that the TOE is the Key Server by setting an MKA priority that is lower than the two MACsec peers. The evaluator then started an MKA session between the TOE and the two active participant peers, also taking a packet capture of the session. The evaluator analyzed the packet capture and noted that the TOE sends MKPDUs at least once every half-second.

Test 13b: Not applicable as the TOE does not support EAP-TLS with DevIDs.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Test 14a: Not applicable as the TOE does not support Group CAKs.

Test 14b: Not applicable as the TOE does not support Group CAKs.

2.2.14.5 MACSEC10:FCS_MKA_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.6 MACSEC10:FCS_MKA_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.7 MACSEC10:FCS_MKA_EXT.1.7

TSS Assurance Activities: The evaluator shall verify that the TSS describes the TOE's compliance with IEEE 802.1X-2010 and 802.1Xbx-2014 for MKA, including the values for MKA and Hello timeout limits and support for data delay protection. The evaluator shall also verify that the TSS describes the ability of the PAE of the TOE to establish unique CAs with individual peers and group CAs using a group CAK such that a new group SAK is distributed every time the group's membership changes. The evaluator shall also verify that the TSS describes the invalid MKPDUs that are discarded automatically by the TSF in a manner that is consistent with the SFR, and that valid MKPDUs are decoded in a manner consistent with IEEE 802.1X-2010 section 11.11.4.

Section 6 (FCS_MKA_EXT.1) of the ST states the TOE implements the MKA Protocol in accordance with IEEE 802.1X-2010 and 802.1Xbx-2014.

The data delay protection is enabled for MKA as a protection guard against an attack on the configuration protocols that MACsec is designed to protect by alternately delaying and deliver-ing their MPDUs. The "Delay Protection" does not operate if MKA operation is suspended. An MKA Lifetime Timeout limit of 6.0 seconds and Hello Timeout limit of 2.0 seconds is enforced by the TOE.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The TOE discards MACsec Key Agreement Protocol Data Units (MKPDUs) that do not satisfy the requirements listed under FCS_MKA_EXT.1.7. All valid MKPDUs that meet the require-ments as defined under FCS_MKA_EXT.1.8 are decoded in a manner conformant to IEEE 802.1x-2010 Section 11.11.4.

On successful peer authentication, a unique connectivity association is formed between the peers and a secure Connectivity Association Key Name (CKN) is exchanged. After the ex-change, the MKA ICV is validated with a Connectivity Association Key (CAK), which is effective-ly a secret key. The TOE does not support group CAKs.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation provides instructions on how to configure the TOE to act as the key server in an environment with multiple MACsec-capable devices.

Section "MACSEC and MKA Configuration" in the **Admin Guide** provides instructions for configuring the key-server priority in the MKA policy to ensure that the TOE can act as the Key Server when connecting with MACsec peers.

Testing Assurance Activities: The tests below require the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with. Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the key server and principal actor (peer). The evaluator shall then perform the following tests:

Test 15: (Conditional - if any 'group CAK' selection is made in FCS_MKA_EXT.1.5) The evaluator shall perform the following steps:

1. Load one PSK onto the TOE and device B and a second PSK onto the TOE and device C. This defines two pairwise CAs.

2. Generate a group CAK for the group of three devices using ieee8021XKayCreateNewGroup.

3. Observe via packet capture that the TOE distributes the group CAK to the two peers, protected by AES key wrap using their respective PSKs.

4. Verify that B can form an SA with C and connect securely.

5. Disable the KaY functionality of device C using ieee8021XPaePortKayMkaEnable.

6. Generate a group CAK for the TOE and B using ieee8021XKayCreateNewGroup and observe they can connect.

7. The evaluator shall have B attempt to connect to C and observe this fails.

8. Re-enable the KaY functionality of device C.

9. Invoke ieee8021XKayCreateNewGroup again.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

10. Verify that both the TOE can connect to C and that B can connect to C.

(TD0817 applied, supersedes TD0787)

Test 16: The evaluator shall start an MKA session between the TOE and an environmental MACsec peer and then perform the following steps:

1. Send an MKPDU to the TOE's individual MAC address from a peer. Verify the frame is dropped and logged.

2. Send an MKPDU to the TOE that is less than 32 octets long. Verify the frame is dropped and logged.

3. Send an MKPDU to the TOE whose length in octets is not a multiple of 4. Verify the frame is dropped and logged.

4. Send an MKPDU to the TOE that is one byte short. Verify the frame is dropped and logged.

5. Send an MKPDU to the TOE with unknown Agility Parameter. Verify the frame is dropped and logged.

Test 15: Not applicable. The TOE does not support Group CAKs.

Test 16: The evaluator set up two MACsec peers to connect to the TOE. The evaluator ensured that the TOE is the Key Server by setting an MKA priority that is lower than the two MACsec peers. The evaluator then started an MKA session between the TOE and the two active participant peers, also taking a packet capture of the session. The evaluator then sent five modified MKA packets according to the conditions identified in this test case. In each case the TOE detected the failure and rejected the packet.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.15 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.15.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.15.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

The evaluator shall examine the TSS to determine that it specifies the DRBG type, identifies the entropy source(s) seeding the DRBG, and state the assumed or calculated min-entropy supplied either separately by each source or the min-entropy contained in the combined seed value.

The Entropy description is provided in a separate (non-ST) document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section 6 (FCS_RBG_EXT.1) in the ST states that the TOE implements a NIST-approved AES-CTR DRBG, as specified in ISO/IEC 18031:2011 seeded by an entropy source that accumulates entropy from a from a TSF-platform based noise source. The DRBG is seeded with a minimum of 256 bits of entropy, which is at least equal to the greatest security strength of the keys and hashes that it will generate.

Component Guidance Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

The evaluator shall confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

The Entropy description is provided in a separate (non-ST) Cisco proprietary document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section "FIPS Mode" in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode of operation. FIPS Mode of operation must be enabled in the evaluated configuration and allows the TOE to use only approved cryptography including the proper DRBG methods. Otherwise, there is no further configuration required for configuring RNG functionality.

Component Testing Assurance Activities: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

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

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2.

2.2.16 SSH Server Protocol - per TD0631 (NDcPP22e:FCS_SSHS_EXT.1)

2.2.16.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.2 NDcPP22e:FCS_SSHS_EXT.1.2

TSS Assurance Activities: The evaluator shall check to ensure that the TSS contains a list of supported public key algorithms that are accepted for client authentication and that this list is consistent with signature verification algorithms selected in FCS_COP.1/SigGen (e.g., accepting EC keys requires corresponding Elliptic Curve Digital Signature algorithm claims).

The evaluator shall confirm that the TSS includes the description of how the TOE establishes a user identity when an SSH client presents a public key or X.509v3 certificate. For example, the TOE could verify that the SSH client's presented public key matches one that is stored within the SSH server's authorized_keys file.

If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, then the evaluator shall confirm its role in the authentication process is described in the TSS. (TD0631 applied)

Section 6 (FCS_SSHS_EXT.1) in the ST indicates he TOE supports user public key and/or password based authentication. The TOE uses rsa-sha2-512 and rsa-sha2-256 for host key authentication and ssh-rsa for client public key authentication. This is consistent with the RStA digital signature algorithm selected in FCS_COP.1/SigGen. When the SSH client presents a public key, the TSF verifies it matches the one configured for the Administrator account. If the presented public key does not match the one configured for the Administrator account, access is denied.

Section 6 (FIA_UIA_EXT.1) states administrative access to the TOE is facilitated through a local password-based authentication and SSH public key authentication mechanisms on the TOE through which all Administrator actions are mediated. Once a potential (unauthenticated) administrative user attempts to access the TOE through an interactive administrative interface, the TOE prompts the user for a user name and password or SSH public key authentication. No access is allowed to the administrative functionality of the TOE until the administrator is successfully identified and authenticated.

The process for authentication is the same for administrative access whether administration is occurring via a directly connected console or remotely via SSHv2 secured connection.

At initial login, the administrative user is prompted to provide a username. After the user provides the username, the user is prompted to provide the administrative password associated with the user account. The TOE then either grants administrative access (if the combination of username and password is correct) or indicates that the login was unsuccessful. The TOE does not provide a reason for failure in the cases of a login failure.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test objective: The purpose of these tests is to verify server supports each claimed client authentication method.

Test 1: For each supported client public-key authentication algorithm, the evaluator shall configure a remote client to present a public key corresponding to that authentication method (e.g., 2048-bit RSA key when using ssh-rsa

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

public key). The evaluator shall establish sufficient separate SSH connections with an appropriately configured remote non-TOE SSH client to demonstrate the use of all applicable public key algorithms. It is sufficient to observe the successful completion of the SSH Authentication Protocol to satisfy the intent of this test.

Test 2: The evaluator shall choose one client public key authentication algorithm supported by the TOE. The evaluator shall generate a new client key pair for that supported algorithm without configuring the TOE to recognize the associated public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

Test 3: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication succeeds when the correct password is provided by the connecting SSH client.

Test 4: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication fails when the incorrect password is provided by the connecting SSH client.

(TD0631 applied)

Test 1: The TOE supports ssh-rsa for client public key authentication. The evaluator generated an RSA 2048 bit key pair on the test server and then configured an admin user on the TOE with the public key. The evaluator then attempted to login to the TOE using this ssh-rsa public key and observed that the login was successful.

Test 2: The evaluator next demonstrated an unsuccessful login using an unrecognized public key.

Test 3: The evaluator attempted to connect to the TOE using a SSH client alternately using the correct and incorrect password. The evaluator found that only the correct password would yield a successful SSH session.

Test 4: This was performed as part of Test 3 above where an unsuccessful SSH connection was demonstrated using an incorrect password.

2.2.16.3 NDcPP22e:FCS_SSHS_EXT.1.3

TSS Assurance Activities: The evaluator shall check that the TSS describes how 'large packets' in terms of RFC 4253 are detected and handled.

Section 6 (FCS_SSHS_EXT.1) in the ST states that SSHv2 connections will be dropped if the TOE receives a packet larger than 65,806 bytes. Large packets are detected by the SSHv2 implementation and dropped internal to the SSH process.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

The evaluator created and sent a packet to the TOE that was larger than the maximum packet size. The TOE rejected the packet and the connection was closed.

2.2.16.4 NDcPP22e:FCS_SSHS_EXT.1.4

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

Section 6 (FCS_SSHS_EXT.1) in the ST states that the TSF's SSH transport implementation supports the following encryption algorithms: aes128-cbc, aes256-cbc, aes128-gcm@openssh.com, aes256-gcm@openssh.com. This is consistent with the algorithms specified in the requirement.

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

Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for configuring the supported encryption algorithms used in SSH: aes256-gcm@openssh.com, aes128-gcm@openssh.com, aes256-cbc, aes128-cbc.

Testing Assurance Activities: The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall start session establishment for an SSH connection from a remote client (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator attempted to connect to the TOE using a SSH client alternately configured with each of the claimed ciphers. The evaluator observed successful connections between the TOE and the SSH client. The evaluator determined from the packet capture that the TOE only offers the claimed ciphers.

2.2.16.5 NDcPP22e:FCS_SSHS_EXT.1.5

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the SSH server's host public key algorithms supported are specified and that they are identical to those listed for this component. (TD0631 applied)

SSH host key authentication with rsa-sha2-256 and rsa-sha2-512 and client public key authentication with ssh-rsa. This is consistent with the requirement.

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

set of algorithms advertised by the TOE may have to be restricted to meet the requirements).

Section “SSH Remote Administration Protocol” in the **Admin Guide** provides instructions for configuring the supported host key algorithms: rsa-sha2-256, rsa-sha2-512 and the client public key authentication algorithm: ssh-rsa.

Testing Assurance Activities: Test objective: This test case is meant to validate that the TOE server will support host public keys of the claimed algorithm types.

Test 1: The evaluator shall configure (only if required by the TOE) the TOE to use each of the claimed host public key algorithms. The evaluator will then use an SSH client to confirm that the client can authenticate the TOE server public key using the claimed algorithm. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Has effectively been moved to FCS_SSHS_EXT.1.2.

Test objective: This negative test case is meant to validate that the TOE server does not support host public key algorithms that are not claimed.

Test 2: The evaluator shall configure a non-TOE SSH client to only allow it to authenticate an SSH server host public key algorithm that is not included in the ST selection. The evaluator shall attempt to establish an SSH connection from the non-TOE SSH client to the TOE SSH server and observe that the connection is rejected.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

(TD0631 applied)

Test 1: The evaluator established an SSH connection with the TOE using each claimed host key algorithm. The connection was successful.

Test 2: The evaluator attempted to connect to the TOE using a host public key algorithm that is not included in the ST selection and observed that the connection failed.

2.2.16.6 NDcPP22e:FCS_SSHS_EXT.1.6

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

Section 6 (FCS_SSHS_EXT.1) in the ST states that the TSF's SSH transport Implementation supports the following MAC algorithms: hmac-sha2-256. When aes128-gcm@openssh.com or aes256-gcm@openssh.com is used as the encryption algorithm the MAC algorithm is implicit. This is consistent with the requirement.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the 'none' MAC algorithm is not allowed).

Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for configuring the supported MAC algorithms: hmac-sha2-256. When aes128-gcm@openssh.com or aes256-gcm@openssh.com is used as the encryption algorithm, the MAC algorithm is implicit.

Testing Assurance Activities: Test 1 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall establish an SSH connection using each of the algorithms, except 'implicit', specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 2 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall configure an SSH client to only allow a MAC algorithm that is not included the ST selection. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Test 1: The evaluator established an SSH connection with the TOE using each of the claimed MAC algorithms. The evaluator observed a successful connection using each claimed integrity algorithm.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using the HMAC-MD5 algorithm. The connection attempt failed.

2.2.16.7 NDcPP22e:FCS_SSHS_EXT.1.7

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

Section 6 (FCS_SSHS_EXT.1) in the ST states that the TSF's SSH key exchange implementation supports the following key exchange algorithm: ecdh-sha2-nistp256, and ecdh-sha2-nistp384. This is consistent with the requirement.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for configuring the supported key exchange algorithm: ecdh-sha2-nistp256, and ecdh-sha2-nistp384.

Testing Assurance Activities: Test 1: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

Test 1: This test was performed as part of test 2 below where the evaluator attempted to establish an SSH connection with the TOE using diffiehellman-group1-sha1 key exchange. The connection attempt failed.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using each allowed key exchange method: ecdh-sha2-nistp256, and ecdh-sha2-nistp384. The connections succeeded.

2.2.16.8 NDcPP22e:FCS_SSHS_EXT.1.8

TSS Assurance Activities: The evaluator shall check that the TSS specifies the following:

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

a) Both thresholds are checked by the TOE.

b) Rekeying is performed upon reaching the threshold that is hit first.

Section 6 (FCS_SSHS_EXT.1) in the ST states that the TSF's SSH implementation will perform a rekey after no longer than one hour or no more than one gigabyte of data has been transmitted with the same session key. Both thresholds are checked. Rekeying is performed upon reaching whichever threshold is met first. The Administrator can configure lower rekey values if desired. The minimum time value is 10 minutes. The minimum volume value is 100 kilobytes.

Guidance Assurance Activities: If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, then the evaluator shall check that the guidance documentation describes how to configure those thresholds. Either the allowed values are specified in the guidance documentation and must not exceed the limits specified in the SFR (one hour of session time, one gigabyte of transmitted traffic) or the TOE must not accept values beyond the limits specified in the SFR. The evaluator shall check that the guidance documentation describes that the TOE reacts to the first threshold reached.

Section “SSH Remote Administration Protocol” in the **Admin Guide** provides instructions for configuring time-based and volume-based rekey values. SSH connections with the same session keys cannot be used longer than one hour, and with no more than one gigabyte of transmitted data. Values can be configured to be lower if desired. The minimum time value is 10 minutes. The minimum volume value is 100 kilobytes. The Guide warns that in order to ensure rekeying is performed before one hour expires, the Administrator should specify a rekey time of 59 minutes.

Testing Assurance Activities: The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

For testing of the time-based threshold the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one hour of session time but the value used for testing shall not exceed one hour. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

(e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8).

The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one gigabyte of transferred traffic, but the value used for testing shall not exceed one gigabyte. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1(3)/Functions).

In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

a) An argument is present in the TSS section describing this hardware-based limitation and

b) All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

The evaluator configured the rekey data limit to 100 KB. The evaluator attempted to connect to the TOE using a SSH client sending data and confirmed that a rekey happened when the configured threshold was reached. Next the evaluator configured the rekey time limit to 10 minutes. The evaluator attempted to connect to the TOE using an SSH client and confirmed that a rekey happened when the configured threshold was reached.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.17 TLS Client Protocol Without Mutual Authentication - per TD0670 & TD0790 (NDcPP22e:FCS_TLSC_EXT.1)

2.2.17.1 NDcPP22e:FCS_TLSC_EXT.1.1

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

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

Section 6 (FCS_TLSC_EXT.1) in the ST states that the TSF implements TLS 1.2 conformant to RFC 5246 to provide secure TLS communication between itself and a Syslog server supporting the following ciphersuites:

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289

These cipher suites are consistent with those identified in the requirement.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

Section “TLS - Syslog” and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to conform to the description in the TSS including how to create a TLSv1.2 profile for Syslog, how to configure the supported TLS ciphersuites (consistent with the requirements in the ST), how to configure and authenticate the Root and Intermediate Trustpoint CA certificates and how to configure the reference identifier for the peer.

Testing Assurance Activities: Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites 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 HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall attempt to establish the connection using a server with a server certificate that contains the Server Authentication purpose in the extendedKeyUsage extension and verify that a connection is established. The evaluator will then verify that the client rejects an otherwise valid server certificate that lacks the Server Authentication purpose in the extendedKeyUsage field, and a connection is not established. Ideally, the two certificates should be identical except for the extendedKeyUsage field.

Test 3: The evaluator shall send a server certificate in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDSA certificate while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite). The evaluator shall verify that the TOE disconnects after receiving the server's Certificate handshake message.

Test 4: The evaluator shall perform the following 'negative tests':

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

a) The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection.

b) Modify the server's selected ciphersuite in the Server Hello handshake message to be a ciphersuite not presented in the Client Hello handshake message. The evaluator shall verify that the client rejects the connection after receiving the Server Hello.

c) [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension the evaluator shall configure the server to perform an ECDHE or DHE key exchange in the TLS connection using a non-supported curve/group (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

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

a) Change the TLS version selected by the server in the Server Hello to a non-supported TLS version and verify that the client rejects the connection.

b) [conditional]: If using DHE or ECDH, modify the signature block in the Server's Key Exchange handshake message, and verify that the handshake does not finished successfully, and no application data flows. This test does not apply to cipher suites using RSA key exchange. If a TOE only supports RSA key exchange in conjunction with TLS, then this test shall be omitted.

Test 6: The evaluator performs the following 'scrambled message tests':

a) Modify a byte in the Server Finished handshake message and verify that the handshake does not finish successfully and no application data flows.

b) Send a garbled message from the server after the server has issued the ChangeCipherSpec message and verify that the handshake does not finish successfully and no application data flows.

c) 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 ciphersuite) or that the server denies the client's Finished handshake message.

Test 1: The evaluator configured a test server to accept each ciphersuite allowed by the PP, a single ciphersuite at a time. While the test server listened to the single configured ciphersuite, the evaluator caused the TOE to attempt a connection to the test server. If a ciphersuite is supported by the TOE, the connection was successful. If a ciphersuite is not supported by the TOE, the connection was unsuccessful.

Test 2: The evaluator configured the TOE to connect to a test server and attempted two connections. During the first TLS negotiation the test server sent a valid certificate chaining to a CA known by the TOE. The certificate included the Server Authentication extended key usage (EKU) field. The evaluator observed that the connection was successful. During the second connection, the server presented a certificate chaining to a CA known by the TOE. However, the certificate did not include the Server Authentication extended key usage (EKU) field. The evaluator observed that the connection failed.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Test 3: The evaluator configured the test server to negotiate TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 but then send an ECDSA key in its certificate message. The TOE rejected the connection attempt.

Test 4: (Part a): The evaluator configured the TOE to communicate with a test server that sends only a TLS_NULL_WITH_NULL_NULL ciphersuite in the server hello. The evaluator then attempted to establish a TLS session from the TOE to the test server and observed that the TOE rejected the connection attempt.

(Part b): The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to choose a ciphersuite that the TOE did not offer in its Client Hello handshake message and observed that the TOE rejected the connection attempt.

(Part c): The evaluator configured the TOE to connect to a test server using TLS with a TOE supported ECDHE key exchange method. The evaluator also configured the test server to accept that same ECHDE key exchange method, but to require a curve that was not supported by the TOE (i.e., P-192). The evaluator observed that the TOE rejected the connection attempt.

Test 5: (Part a): The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to use a TLS version in the Server Hello that is a non-supported TLS version (version 1.4 represented by two bytes 0x0305). The evaluator observed that the TOE rejected the connection attempt.

(Part b): The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to modify the signature block in the Server's Key Exchange handshake message and observed that the TOE rejected the connection attempt.

Test 6: (Part a): The evaluator attempted a connection to the TOE where the evaluator modified a byte in the Finished handshake message (by XORing 0xff with the first byte of the calculated MAC before the packet is TLS encrypted and sent to the TOE), verified that the TOE rejected the connection attempt after receiving the modified Finished message and that the TOE sent no application data.

(Part b): The evaluator garbled a message between the TOE and its TLS peer. The modification occurred after the Server sent the ChangeCipherSpec message. The evaluator observed that the Client denies the connection. Due to the nature of the error, regardless of whether the TOE is the client or server, the client is always the first to recognize the error.

(Part c): The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to modify one byte in the server's nonce in the Server Hello handshake message. The evaluator observed that the client rejected the connection.

2.2.17.2 NDcPP22e:FCS_TLSC_EXT.1.2

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

TSS Assurance Activities: The evaluator shall ensure that the TSS describes the client's method of establishing all reference identifiers from the administrator/application-configured reference identifier, including which types of reference identifiers are supported (e.g. application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported.

Note that where a TLS channel is being used between components of a distributed TOE for FPT_ITT.1, the requirements to have the reference identifier established by the user are relaxed and the identifier may also be established through a 'Gatekeeper' discovery process. The TSS should describe the discovery process and highlight how the reference identifier is supplied to the 'joining' component. Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1 and the ST author selected attributes from RFC 5280, the evaluator shall ensure the TSS describes which attribute type, or combination of attributes types, are used by the client to match the presented identifier with the configured identifier. The evaluator shall ensure the TSS presents an argument how the attribute type, or combination of attribute types, uniquely identify the remote TOE component; and the evaluator shall verify the attribute type, or combination of attribute types, is sufficient to support unique identification of the maximum supported number of TOE components.

If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE's conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC5952 for IPv6, RFC 3986 for IPv4) is enforced.

Section 6 (FCS_TLSC_EXT.1) in the ST states that when establishing a TLS connection, the TOE supports reference identifiers of type DNS-ID and IP address and will seek a match to the DNS domain name or IP address respectively in the subjectAltName extension. If the TOE determines there is a mismatch in the presented identifier, it will not establish the TLS trusted channel connection. The TOE does not support the use of wildcards within certificates and does not support certificate pinning.

Guidance Assurance Activities: The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not, and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE includes support for IP addresses, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

Where the secure channel is being used between components of a distributed TOE for FPT_ITT.1, the SFR selects attributes from RFC 5280, and FCO_CPC_EXT.1.2 selects 'no channel'; the evaluator shall verify the guidance provides instructions for establishing unique reference identifiers based on RFC5280 attributes.

Section “Create and Configure a Certificate Map” in the **Admin Guide** provides instructions for configuring the reference identifier of the peer syslog server. The identifier (either FQDN or IP address) of the peer is specified in the SAN and the instructions explicitly state that the SAN field should be specified along with the value to match for the remote syslog server.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Testing Assurance Activities: Note that the following tests are marked conditional and are applicable under the following conditions:

a) For TLS-based trusted channel communications according to FTP_ITC.1 where RFC 6125 is selected, tests 1-6 are applicable.

or

b) For TLS-based trusted path communications according to FTP_TRP where RFC 6125 is selected, tests 1-6 are applicable

or

c) For TLS-based trusted path communications according to FPT_ITT.1 where RFC 6125 is selected, tests 1-6 are applicable. Where RFC 5280 is selected, only test 7 is applicable.

Note that for some tests additional conditions apply.

IP addresses are binary values that must be converted to a textual representation when presented in the CN of a certificate. When testing IP addresses in the CN, the evaluator shall follow the following formatting rules:

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.

- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

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

a) Test 1 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the CN.

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

b) Test 2 [conditional]: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

each supported SAN type (e.g. IPv4, IPv6, FQDN, URI). When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the SAN.

c) Test 3 [conditional]: If the TOE does not mandate the presence of the SAN extension, the evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. If the TOE does mandate the presence of the SAN extension, this Test shall be omitted.

d) Test 4 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator shall verify that the connection succeeds. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, SRV).

e) Test 5 [conditional]: The evaluator shall perform the following wildcard tests with each supported type of reference identifier that includes a DNS name (i.e. CN-ID with DNS, DNS-ID, SRV-ID, URI-ID):

1) [conditional]: The evaluator shall present a server certificate containing a wildcard that is not in the left- most label of the presented identifier (e.g. foo.*.example.com) and verify that the connection fails.

2) [conditional]: The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds if wildcards are supported or fails if wildcards are not supported. The evaluator shall configure the reference identifier without a left-most label as in the certificate (e.g. example.com) and verify that the connection fails. The evaluator shall configure the reference identifier with two left-most labels (e.g. bar.foo.example.com) and verify that the connection fails. (Remark: Support for wildcards was always intended to be optional. It is sufficient to state that the TOE does not support wildcards and observe rejected connection attempts to satisfy corresponding assurance activities.)

f) Objective: The objective of this test is to ensure the TOE is able to differentiate between IP address identifiers that are not allowed to contain wildcards and other types of identifiers that may contain wildcards.

Test 6: [conditional]If IP address identifiers are supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.1.20, CN=2001:0DB8:0000:0000:0008:0800:200C:* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

This negative test corresponds to the following section of the Application Note 64/105: 'The exception being, the use of wildcards is not supported when using IP address as the reference identifier.'

Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 6.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

(TD0790 applied, supersedes TD0670)

Test 7 [conditional]: If the secure channel is used for FPT_ITT, and RFC 5280 is selected, the evaluator shall perform the following tests. Note, when multiple attribute types are selected in the SFR (e.g. when multiple attribute types are combined to form the unique identifier), the evaluator modifies each attribute type in accordance with the matching criteria described in the TSS (e.g. creating a mismatch of one attribute type at a time while other attribute types contain values that will match a portion of the reference identifier):

1) The evaluator shall present a server certificate that does not contain an identifier in the Subject (DN) attribute type(s) that matches the reference identifier. The evaluator shall verify that the connection fails.

2) The evaluator shall present a server certificate that contains a valid identifier as an attribute type other than the expected attribute type (e.g. if the TOE is configured to expect id-atserialNumber=correct_identifier, the certificate could instead include id-at-name=correct_identifier), and does not contain the SAN extension. The evaluator shall verify that the connection fails. Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass this test.

3) The evaluator shall present a server certificate that contains a Subject attribute type that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds.

4) The evaluator shall confirm that all use of wildcards results in connection failure regardless of whether the wildcards are used in the left or right side of the presented identifier. (Remark: Use of wildcards is not addressed within RFC 5280.)

The evaluator configured the TOE to connect with a test server using TLS with the test server alternately configured with a certificate identifier as indicated in tests 1-6 in this assurance activity. The evaluator ensured the test server and TOE could connect only when the identifier fulfilled the required rules. This test was iterated using both DNS and IPv4 addresses.

Test 1: No SAN, Bad CN - the connection fails

Test 2: Bad SAN, Good CN - the connection fails.

Test 3: No SAN, Good CN - not applicable as the TOE always requires a SAN.

Test 4: Good SAN, Bad CN - the connection is successful

Test 5: As demonstrated by testing, the TOE does not support the use of wildcards.

Test 6: As demonstrated by testing, the TOE does not support the use of wildcards.

Test 7: Not applicable. The TOE does not support ITT channels or claim FPT_ITT.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.17.3 NDcPP22e:FCS_TLSC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that using an invalid certificate results in the function failing as follows:

Test 1: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds and a trusted channel can be established.

Test 2: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

Test 3 : The purpose of this test to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

Test 1: This test was performed as part of NDcPP22e:FIA_X509_EXT.1.1_Rev_t1 Test 1 which demonstrates a successful connection with a valid certificate chain.

Test 2: This test has been performed in several other test activities. Specifically, this test repeats the assurance activities as described here.

- match the reference identifier -- Corresponds to FCS_TLSC_EXT.1.2 Tests 1 through 7.
- validate certificate path -- Corresponds to FIA_X509_EXT.1/REV.1 Test 1
- validate expiration date -- Corresponds to FIA_X509_EXT.1/REV.1 Test 2
- determine the revocation status -- Corresponds to FIA_X509_EXT.2 Test 1

Test 3: Not applicable. The TOE does not support administrative override.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.2.17.4 NDcPP22e:FCS_TLSC_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that TSS describes the Supported Elliptic Curves/Supported Groups Extension and whether the required behavior is performed by default or may be configured.

Section 6 (FCS_TLSC_EXT.1) in the ST states that for TLS 1.2 connections to the Syslog server, the TSF presents secp256r1, secp384r1, and secp521r1 and no other curves in the Supported Group extension of the Client Hello. This behavior is implemented by default and is not configurable.

Guidance Assurance Activities: If the TSS indicates that the Supported Elliptic Curves/Supported Groups Extension must be configured to meet the requirement, the evaluator shall verify that AGD guidance includes configuration of the Supported Elliptic Curves/Supported Groups Extension.

Section “FIPS Mode” in the **Admin Guide** provides instructions for configuring the TOE for FIPS mode operation which ensures that the TOE is only allowed to use approved cryptography. Section “TLS - Syslog” and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to conform to the description in the TSS including how to create a TLSv1.2 profile for Syslog and how to configure the supported TLS ciphersuites (consistent with the requirements in the ST). There are no further explicit instructions required in order to configure the supported Elliptic curves.

Testing Assurance Activities: Test 1 [conditional]: If the TOE presents the Supported Elliptic Curves/Supported Groups Extension, the evaluator shall configure the server to perform ECDHE or DHE (as applicable) key exchange using each of the TOE's supported curves and/or groups. The evaluator shall verify that the TOE successfully connects to the server.

The evaluator attempted to establish a TLS session with the TOE when the evaluator's server specified only one key exchange method in the Server Hello. The evaluator observed that the connection was successful using each of the TOE’s supported curves.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.3.1.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 NDcPP22e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it contains a description, for each supported method for remote administrative actions, of how successive unsuccessful authentication attempts are detected and tracked. The TSS shall also describe the method by which the remote administrator is prevented from successfully logging on to the TOE, and the actions necessary to restore this ability.

The evaluator shall examine the TSS to confirm that the TOE ensures that authentication failures by remote administrators cannot lead to a situation where no administrator access is available, either permanently or temporarily (e.g. by providing local logon which is not subject to blocking).

Section 6 (FIA_AFL.1) in the ST states that the TOE uses an internal AAA function to detect and track failed login attempts. When an account attempting to log into an administrative interface reaches the set maximum number of failed authentication attempts, the account will not be granted access until the time period has elapsed or until the Administrator manually unblocks the account.

The TOE provides the Administrator the ability to specify the maximum number of unsuccessful authentication attempts before an offending account will be blocked. The TOE also provides the ability to specify the time period to block offending accounts.

To avoid a potential situation where password failures made by Administrators leads to no Administrator access until the defined blocking time period has elapsed, the **Admin Guide** instructs the Administrator to configure the TOE for SSH public key authentication which is not subjected to password-based brute force attacks. During the

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

block out period, the TOE provides the ability for the Administrator account to login remotely using SSH public key authentication.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the remote administrator to once again successfully log on is described for each 'action' specified (if that option is chosen). If different actions or mechanisms are implemented depending on the secure protocol employed (e.g., TLS vs. SSH), all must be described.

The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

Section “Configure Authentication Failure” in the **Admin Guide** provides instructions for specifying the value for maximum number of failed login attempts and the time period to lock the offending account.

Section “SSH Remote Administration Protocol” in the **Admin Guide** provides instructions for configuring the switch for SSH public key authentication. This is necessary to avoid a potential situation where password failures by remote Administrators lead to no Administrator access for a temporary period of time. During the defined lockout period, the Switch provides the ability for the Administrator account to login remotely using SSH public key authentication.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which remote administrators access the TOE (e.g. any passwords entered as part of establishing the connection protocol or the remote administrator application):

a) Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.

b) Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows.

If the administrator action selection in FIA_AFL.1.2 is included in the ST then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote administrator's access results in successful access (when using valid credentials for that administrator).

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

If the time period selection in FIA_AFL.1.2 is included in the ST then the evaluator shall wait for just less than the time period configured in Test 1 and show that an authorisation attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorisation attempt using valid credentials results in successful access.

Test 1 & 2: The evaluator performed lockout testing with two different values to verify that the configuration works correctly. The first case was performed with an invalid attempt threshold of 3 and a lockout time of 60 seconds. The second case was performed with an invalid attempt threshold of 5 and a lockout time of 120 seconds. In each case, the evaluator observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator also observed that the account was unlocked after the configured time period had passed. To demonstrate the manual unlocking of an account, the evaluator reused the settings of with an invalid attempt threshold of 3 and a lockout time of 60 seconds. The evaluator again observed that the use of valid credentials immediately after exceeding the limit does not result in a successful login. The evaluator then used the admin command to manually unlock the user prior to the lockout time expiring and viewed the user could again login successfully.

2.3.2 Password Management - per TD0792 (NDcPP22e:FIA_PMG_EXT.1)

2.3.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the TSS lists the supported special character(s) for the composition of administrator passwords.

The evaluator shall check the TSS to ensure that the minimum_password_length parameter is configurable by a Security Administrator.

The evaluator shall check that the TSS lists the range of values supported for the minimum_password_length parameter. The listed range shall include the value of 15.

(TD0792 applied)

Section 6 (FIA_PMG_EXT.1) in the ST states that the TOE supports the local definition of users with corresponding passwords. The passwords can be composed of any combination of upper and lower case letters, numbers, and special characters that include: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, “)” and other special characters listed in table 16. Minimum password length is settable by the Authorized Administrator, and can be configured for

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

minimum password lengths of 1 and maximum of 127 characters. A minimum password length of 8 is recommended.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it:

a) identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and

b) provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

Section "Define Password Policy" in the **Admin Guide** provides instructions for defining a common criteria policy that can be applied to each local account and that will ensure that passwords contain a minimum of 8 characters. A password lifetime can also be configured such that the password will expire after the configured time period and will prompt the user to perform a password change. This section provides instructions on setting the minimum password length and indicates that the TOE supports a minimum length from 1 to 127 characters. It is recommended to configure a password minimum length between 8 and 16 characters.

Section "Add Administrator Account" in the **Admin Guide** provides instructions for configuring a user account and specifying the common criteria password policy for that account. It also identifies the characters that may be used in passwords. It also provides the following guidance for selecting a strong password:

Note: Passwords should not use common words, repeated characters, or easily guessed patterns. Use the following guidelines while creating strong passwords:

- The password should not contain the associated username and the username should not be reversed.
- The characters in the password should not be repeated more than three times consecutively.
- The password should not be cisco, ocsic, admin, nimda, or any variant obtained by changing the capitalization of letters therein, or by substituting "1" "|" or "!" for i, and/or substituting "0" for "o", and/or substituting "$" for "s".
- The password should be composed of uppercase letters, lowercase letters, digits, and the special characters listed in table 4. The password length should be at least 8.

Component Testing Assurance Activities: The evaluator shall perform the following tests.

Test 1: The evaluator shall compose passwords that meet the requirements in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, and a minimum length listed in the requirement are supported and justify the subset of those characters chosen for testing.

Test 2: The evaluator shall compose passwords that do not meet the requirements in some way. For each password, the evaluator shall verify that the TOE does not support the password. While the evaluator is not

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that the TOE enforces the allowed characters and the minimum length listed in the requirement and justify the subset of those characters chosen for testing.

Test 1&2: The evaluator performed attempts to set passwords of varying lengths and characters to demonstrate that passwords comply with a minimum length and support the claimed set of characters.

2.3.3 Pre-Shared Key Composition (MACSEC10:FIA_PSK_EXT.1)

2.3.3.1 MACSEC10:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.3.2 MACSEC10:FIA_PSK_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it describes the process by which the bit-based pre-shared keys are generated (if the TOE supports this functionality), and confirm that this process uses the RBG specified in FCS_RBG_EXT.1.

Section 6 (FIA_PSK_EXT.1) of the ST states the TOE supports use of pre-shared keys for MACsec key agreement protocols as defined by IEEE 802.1X. The pre-shared keys are not generated by the TOE, but the TOE accepts the keys in the form of HEX strings. This is done via the CLI configuration command 'key chain test_key macsec'. The TOE accepts pre-shared keys that are 32 characters in length for AES 128-bit CMAC mode encryption and pre-shared keys that are 64 characters in length for AES 256-bit CMAC mode encryption.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on the composition of strong PSKs, and (if the selection indicates keys of various lengths can be entered) that it provides information on the range of lengths supported.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator shall confirm the operational guidance contains instructions for either entering bit-basedPSKs for each protocol identified in the requirement, or generating a bit-based pre-shared key, or both.

Section "MACSEC and MKA Configuration" in the **Admin Guide** provides instructions for configuring a keychain and a pre-shared key in the form of a HEX string via the key-string command in the key chain. When aes-128-cmac is selected, the Administrator will need to provide a 32 hex digit PSK. When aes-256-cmac is selected, the Administrator will need to provide a 64 hex digit PSK.

Component Testing Assurance Activities: The evaluator shall also perform the following tests for each protocol (or instantiation of a protocol, if performed by a different implementation on the TOE). Note that one or more of these tests can be performed with a single test case.

Test 17: (conditional, the TOE supports PSKs of multiple lengths) The evaluator shall use the minimum length, the maximum length, a length inside the allowable range, and invalid lengths beyond the supported range (both higher and lower). The minimum, maximum, and included length tests should be successful, and the invalid lengths must be rejected by the TOE.

Test 18: (conditional, the TOE does not generate bit-based PSKs) The evaluator shall obtain a bit-based PSK of the appropriate length and enter it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

Test 19: (conditional, the TOE can generate bit-based PSKs) The evaluator shall generate a bit-based PSK of the appropriate length and use it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key.

Test 17: The evaluator attempted to establish a connection using pre-shared keys of valid and invalid lengths and confirmed that pre-shared keys with valid lengths resulted in successful connections, while the attempt to configure pre-shared keys of invalid lengths was unsuccessful.

Test 18: The evaluator entered a pre-shared key according to instructions in the guide and demonstrated a successful connection.

Test 19: Not applicable. The TOE does not generate bit-based keys.

2.3.4 PROTECTED AUTHENTICATION FEEDBACK (NDcPP22e:FIA_UAU.7)

2.3.4.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

There are no preparatory steps needed to ensure authentication data is not revealed. Section 6 (FIA_UAU.7) in the ST states that when a user enters their password at the local console or via a remote session, the TOE does not echo any characters as the password is entered.

Component Testing Assurance Activities: The evaluator shall perform the following test for each method of local login allowed:

a) Test 1: The evaluator shall locally authenticate to the TOE. While making this attempt, the evaluator shall verify that at most obscured feedback is provided while entering the authentication information.

Test 1- This test was performed as part of the tests for FIA_UIA_EXT.1-t1 where the evaluator observed that passwords are not echoed back at the console login.

2.3.5 Password-based Authentication Mechanism (NDcPP22e:FIA_UAU_EXT.2)

2.3.5.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See FIA_UIA_EXT.1

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component Guidance Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See FIA_UIA_EXT.1

Component Testing Assurance Activities: Evaluation Activities for this requirement are covered under those for FIA_UIA_EXT.1. If other authentication mechanisms are specified, the evaluator shall include those methods in the activities for FIA_UIA_EXT.1.

See FIA_UIA_EXT.1

2.3.6 User Identification and Authentication (NDcPP22e:FIA_UIA_EXT.1)

2.3.6.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.6.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a 'successful logon'.

The evaluator shall examine the TSS to determine that it describes which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

For distributed TOEs the evaluator shall examine that the TSS details how Security Administrators are authenticated and identified by all TOE components. If not all TOE components support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the TSS shall describe how the overall TOE functionality is split between TOE components including how it is ensured that no unauthorized access to any TOE component can occur.

For distributed TOEs, the evaluator shall examine the TSS to determine that it describes for each TOE component which actions are allowed before user identification and authentication. The description shall cover authentication and identification for local and remote TOE administration. For each TOE component that does not support authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2 the TSS shall describe any unauthenticated services/services that are supported by the component.

Section 6 (FIA_UIA_EXT.1) in the ST states that the TOE requires all users to be successfully identified and authenticated before allowing any TSF mediated actions to be performed. Prior to being granted access, a login warning banner is displayed.

Administrative access to the TOE is facilitated through a local password-based authentication and SSH public key authentication mechanisms on the TOE through which all Administrator actions are mediated. Once a potential (unauthenticated) administrative user attempts to access the TOE through an interactive administrative interface, the TOE prompts the user for a user name and password or SSH public key authentication. No access is allowed to the administrative functionality of the TOE until the administrator is successfully identified and authenticated.

The process for authentication is the same for administrative access whether administration is occurring via a directly connected console or remotely via SSHv2 secured connection. At initial login, the administrative user is prompted to provide a username. After the user provides the username, the user is prompted to provide the administrative password associated with the user account. The TOE then either grants administrative access (if the combination of username and password is correct) or indicates that the login was unsuccessful. The TOE does not provide a reason for failure in the cases of a login failure.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-shared keys, tunnels, certificates, etc.) to logging in are described. For each supported the login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the allowed services.

Section “Operational Environment” in the **Admin Guide** describes the local console which is directly connected to the TOE via the Serial console port. Section “Switch- Initial Configuration” in the **Admin Guide** provides manual steps for the initial configuration of the TOE (via the CLI on the local console) including configuring the Enable secret password, providing an initial configuration for an Out of Band management interface and configuring the console to require username and password authentication.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Section "Preparative Procedures and Operational Guidance for the TOE" in the **Admin Guide** describes how to access the CLI by connecting the RJ-45 console port or USB console port of the switch to your PC or workstation and accessing the switch through a terminal emulation program. It provides the necessary setting to use to get a console connection.

Section "Configure Local Authentication" in the **Admin Guide** provides the steps to enable the authentication, authorization and accounting access control model, set the default authentication at login to use local authentication and the default authorization method to use local credentials.

Section "Define Password Policy" in the **Admin Guide** provides instructions for defining a common criteria policy that can be applied to each local account and that will ensure that passwords contain a minimum of 8 characters. This section provides instructions on setting the minimum password length and indicates that the valid minimum password lengths supported are from 1 to 127 characters and it is recommended to set the minimum length to between 8 and 16 characters.

Section "Add Administrator Account" in the **Admin Guide** provides instructions for configuring an administrator account and specifying the common criteria password policy for that account. It also identifies the characters that may be used in passwords.

Section "Access Banner" in the **Admin Guide** provides instructions for configuring the warning banner that will display on the CLI and SSH interface prior to allowing any administrative access.

Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for configuring remote administration using SSH. This includes SSH client authentication using either password or SSH public key.

Section "Access CLI Over SSH" in the **Admin Guide** provides instructions for initiating a connection using SSH from a remote management workstation. Successful login will result in privileged administrator access denoted by the 'hashtag' symbol.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

b) Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

c) Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.

d) Test 4: For distributed TOEs where not all TOE components support the authentication of Security Administrators according to FIA_UIA_EXT.1 and FIA_UAU_EXT.2, the evaluator shall test that the components authenticate Security Administrators as described in the TSS.

The TOE offers the following user interfaces where authentication is provided.

- Local console using local authentication with password
- SSH CLI using local authentication with password
- SSH CLI using local authentication with public key

Test 1 - Using each interface, the evaluator performed an unsuccessful and successful logon of each type using bad and good credentials respectively. Results for administrator login demonstrating valid and invalid SSH public key were performed as part of FCS_SSHS_EXT.1.2-t1.

Test 2 - Using each interface, the evaluator observed that there are no services available nor any configuration options offered to administrators to control services available prior to authentication other than viewing of the warning banner. After TOE configuration, the evaluator performed an nmap scan of the TOE and confirmed that there were no additional network services offered by the TOE that were not identified in the Security Target.

Test 3 - This test was performed as part of test 1 & 2. Using each interface, the evaluator found that the TOE does not allow any activity prior to login locally or remotely except the operations specified in the requirement.

Test 4 - Not applicable. The TOE is not a distributed TOE.

2.3.7 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/Rev)

2.3.7.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

a) Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOEs trust store, or by setting up the trust store in a way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store).

Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE's trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

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

c) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates - conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the peer certificate and revocation of the peer intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore the revoked certificate(s) used for testing shall not be a trust anchor.

d) Test 4: If OCSP 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.

e) 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.)

f) 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.)

g) Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

h) The following tests are run when a minimum certificate path length of three certificates is implemented.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall conduct the following tests:

Test 8a: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a valid chain of EC certificates (terminating in a trusted CA certificate), 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 TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a chain of EC certificates (terminating in a trusted CA certificate), where the intermediate certificate in the certificate chain uses an explicit format version of the Elliptic Curve parameters in the public key information field, and 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 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

(TD0527 12/2020 update applied)

Test 1a & 1b: The evaluator configured the TOE to have the trusted root CA used on the test server to anchor all of its certificates. The evaluator then attempted to connect the TLSC TOE client to the test server and saw a successful connection. The evaluator then removed the trusted root CA from the truststore. The evaluator then attempted to connect the TLSC TOE client to the test server and saw an unsuccessful connection as expected.

Test 2: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid and 2) that is expired and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the TOE to the test server and observed that the connection succeeded only if there were no expired certificates.

Test 3: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that is revoked, and 3) issued by an intermediate CA that is revoked. In each case, the evaluator then attempted to connect the TOE to the test server and confirmed that the connection only succeeded if there were no revoked certificates. This test was executed using CRL. The TOE does not support OCSP.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Test 4: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) issued by an intermediate CA referring to a CRL revocation server where the signer lacks cRLSign, and 3) issued by an intermediate CA whose issuer CA refers to a CRL revocation server where the signer lacks cRLSign. In each case, the evaluator then attempted to connect the TOE to the test server and confirmed that the connection only succeeded if all retrieved CRLs were signed using certificates with cRLSign.

Test 5: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator attempted to connect the TOE to the test server and verified that the connection only succeeded if the certificate was not modified/corrupted.

Test 6: This test was performed as part of Test 5.

Test 7: This test was performed as part of Test 5.

Test 8 (all parts): Not applicable. The TOE does not support ECDSA certificates.

2.3.7.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation). For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

a) Test 1: The evaluator shall ensure that at least one of the CAs in the chain does not contain the basicConstraints extension. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

without the basicConstraints extension to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

b) Test 2: The evaluator shall ensure that at least one of the CA certificates in the chain has a basicConstraints extension in which the CA flag is set to FALSE. The evaluator confirms that the TOE rejects such a certificate at one (or both) of the following points: (i) as part of the validation of the leaf certificate belonging to this chain; (ii) when attempting to add a CA certificate with the CA flag set to FALSE to the TOE's trust store (i.e. when attempting to install the CA certificate as one which will be retrieved from the TOE itself when validating future certificate chains).

The evaluator shall repeat these tests for each distinct use of certificates. Thus, for example, use of certificates for TLS connection is distinct from use of certificates for trusted updates so both of these uses would be tested. But there is no need to repeat the tests for each separate TLS channel in FTP_ITC.1 and FTP_TRP.1/Admin (unless the channels use separate implementations of TLS).

Test 1: For this test, the evaluator alternately configured a test server to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server and observed that the connection was rejected in each case.

Test 2: This was performed as part of Test 1.

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

The TSS shall describe when revocation checking is performed and on what certificates. If the revocation checking during authentication is handled differently depending on whether a full certificate chain or only a leaf certificate is being presented, any differences must be summarized in the TSS section and explained in the Guidance.

Section 6 (FIA_X509_EXT.1/Rev) of the ST states the TOE uses X.509v3 certificates to support authentication for TLS connections. The TSF determines the validity of certificates at the time of authentication by ensuring that the certificate and the certificate path are valid in accordance with RFC 5280. The certificate path is validated by ensuring that all the CA certificates have the basicConstraints extension and the CA flag is set to TRUE and the certificate path must terminate with a trusted CA certificate.

CRL revocation checking is supported by the TOE. Revocation checking is performed on the leaf and intermediate certificate(s) when authenticating a certificate chain provided by the remote peer. There are no functional differences if a full certificate chain or only a leaf certif-icate is presented.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

Section "Enable Remote Syslog Server" of the **Admin Guide** states that The TOE uses X.509v3 certificates to support authentication for TLS connections to a Syslog audit server. The TOE determines the validity of certificates by ensuring that the certificate and the certificate path are valid in accordance with RFC 5280. The certificate path is validated by ensuring that all the CA certificates have the basicConstraints extension and the CA flag is set to TRUE and the certificate path must terminate with a trusted CA certificate. The TOE will also verify the extendedKeyUsage field of the TLS peer certificate contains the Server Authentication purpose. OCSP is not supported; therefore the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) is trivially satisfied by the TOE. Revocation checking is performed on the leaf and intermediate certificate(s) when authenticating a certificate chain provided by the remote peer.

Section “TLS - Syslog” and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog, this includes configuring a trustpoint to perform revocation checking using CRL.

Component Testing Assurance Activities: None Defined

2.3.8 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2)

2.3.8.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 NDcPP22e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it describes how the TOE chooses which certificates to use, and any necessary instructions in the administrative guidance for configuring the operating environment so that the TOE can use the certificates.

The evaluator shall examine the TSS to confirm that it describes the behaviour of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed.

Section 6 (FIA_X509_EXT.2) in the ST states that the TOE determines which certificate to use based upon the trust point configured. The instructions for configuring trust points are provided in the **Admin Guide**. In the event that a network connection cannot be established to verify the revocation status of a certificate for an external peer, the connection will be rejected.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the configuration required in the operating environment so the TOE can use the certificates. The guidance documentation shall also include any required configuration on the TOE to use the certificates. The guidance document shall also describe the steps for the Security Administrator to follow if the connection cannot be established during the validity check of a certificate used in establishing a trusted channel.

Section “TLS - Syslog” in the Admin Guide states that the TOE requires an Audit (syslog) Server in the IT Environment to which the TOE transmits syslog messages over TLS. The TOE will validate the X.509 certificate presented by the remote syslog server but does not require a X.509 certificate for the TOE itself. The Administrator will need to ensure the remote syslog server is properly configured with a valid X.509 certificate and the CDP (Certificate Distribution Point) for CRL revocation checking is available on the network. If the CDP for CRL revocation checking is unavailable or the remote syslog server is not properly configured with a valid X.509 certificate, the TOE will not establish the connection to the Syslog server. In this case, the Administrator should troubleshoot and resolve the issue before proceeding.

Section “TLS - Syslog” and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog, how to configure the supported TLS ciphersuites (consistent with the requirements in the ST), how to configure and authenticate the Root and Intermediate Trustpoint CA certificates and how to configure the reference identifier for the peer.

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.

Test: The evaluator alternately configured a test peer to send an authentication certificate with valid/accessible revocation servers and an authentication certificate with revocation information referring to an inaccessible revocation server. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server expecting the connection to be successful when the revocation server is accessible and when the revocation server is not accessible only if that behavior is claimed for the TOE. The evaluator observed the certificate validation checking behavior in each case and confirmed that it was consistent with the actions selected in FIA_X509_EXT.2.2 in the ST.

2.4 Security management (FMT)

2.4.1 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate)

2.4.1.1 NDcPP22e:FMT_MOF.1.1/ManualUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs it is required to verify the TSS to ensure that it describes how every function related to security management is realized for every TOE component and shared between different TOE components. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

There are no specific requirements for non-distributed TOEs.

The TOE is not distributed.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary steps to perform manual update are described. The guidance documentation shall also provide warnings regarding functions that may cease to operate during the update (if applicable).

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

For distributed TOEs the guidance documentation shall describe all steps how to update all TOE components. This shall contain description of the order in which components need to be updated if the order is relevant to the update process. The guidance documentation shall also provide warnings regarding functions of TOE components and the overall TOE that may cease to operate during the update (if applicable).

Section "Update TOE Software" in the **Admin Guide** provides instructions to manually update the TOE and to ensure that the image's digital signature is verified. Since the process involves rebooting before an upgrade can be completed, the entire device will cease to pass traffic during the update.

Component Testing Assurance Activities: The evaluator shall try to perform the update using a legitimate update image without prior authentication as Security Administrator (either by authentication as a user with no administrator privileges or without user authentication at all - depending on the configuration of the TOE). The attempt to update the TOE should fail.

The evaluator shall try to perform the update with prior authentication as Security Administrator using a legitimate update image. This attempt should be successful. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

The set of functions available to administrators prior to login do not include TOE update as specified in NDcPP22e:FIA_UIA_EXT.1-t2. The successful update of the TOE by an authorized administrator was demonstrated in NDcPP22e:FPT_TUD_EXT.1-t1.

2.4.2 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData)

2.4.2.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are accessible through an interface prior to administrator log-in are identified. For each of these functions, the evaluator shall also confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

If the TOE supports handling of X.509v3 certificates and implements a trust store, the evaluator shall examine the TSS to determine that it contains sufficient information to describe how the ability to manage the TOE's trust store is restricted.

Section 6 (FMT_MTD.1/CoreData) in the ST states that prior to authentication the TOE may be configured by the Administrator to display a customized login banner. No administrative functionality is available prior to administrative login. Only Security Administrators can access the TOE's trust store. This section also states that the TOE provides the ability for Authorized Administrators to access TOE data, such as audit data, configuration data, security attributes, session thresholds, cryptographic keys, and updates. Each of the predefined and administratively configured privilege levels has a set of permissions that will grant access to the TOE data, though with some privilege levels, the access is limited.

Component Guidance Assurance Activities: The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

If the TOE supports handling of X.509v3 certificates and provides a trust store, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to configure and maintain the trust store in a secure way. If the TOE supports loading of CA certificates, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to securely load CA certificates into the trust store. The evaluator shall also review the guidance documentation to determine that it explains how to designate a CA certificate a trust anchor.

The evaluator reviewed the guidance documentation while performing the guidance assurance activities in this AAR. The evaluator identified the TSF data manipulating functions and referenced the guidance documentation to demonstrate that it contained the corresponding configuration information. The evaluator documented these "Guidance Assurance Activities" throughout this AAR with the requirements to which they apply.

Section "Add Administrator Account" in the **Admin Guide** provides instructions for configuring an administrator with privilege level 15. Administrative users with privilege level 15 have full access to all TOE security functions including importing X.509v3 certificates to the TOE's trust store and otherwise maintaining the trust store securely.

See NDcPP22e:FIA_X509_EXT.2.2 which identifies the sections in the **Admin Guide** which describe how the administrator role can configure and maintain the trust store including loading of CA certificates.

Component Testing Assurance Activities: No separate testing for FMT_MTD.1/CoreData is required unless one of the management functions has not already been exercised under any other SFR.

All management functions are exercised under other SFRs.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.4.3 Management of TSF Data (NDcPP22e:FMT_MTD.1/CryptoKeys)

2.4.3.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see chapter 2.4.1.1.

For non-distributed TOEs, the evaluator shall ensure the TSS lists the keys the Security Administrator is able to manage to include the options available (e.g. generating keys, importing keys, modifying keys or deleting keys) and how that how those operations are performed.

Section 6.1 (FMT_MTD.1/CryptoKeys) in the ST states that only Security Administrators can access the TOE's trust store. This section describes that the Authorized Administrator generates RSA key pairs to be used in the TLS and SSH protocols. Zeroization of these keys is described in Section 6.1, Table 20 in the ST. The TOE Administrators can control (generate/delete) RSA Key Pairs and SSH RSA Key Pairs by following the instructions in the Admin Guide.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation lists the keys the Security Administrator is able to manage to include the options available (e.g. generating keys, importing keys, modifying keys or deleting keys) and how that how those operations are performed.

Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for generating an RSA 2048 bit or 3072 bit key for SSH, configuring the SSH server key exchange algorithm (diffie-hellman group 14 sha1), configuring the host key and user public key algorithms and configuring ssh rekey settings.

Section "TLS – Syslog" and associated sub-sections in the **Admin Guide** provide instructions for configuring TLS to establish a trusted channel to the audit server including how to create a TLSv1.2 profile for Syslog, how to configure the supported TLS ciphersuites (consistent with the requirements in the ST) and how to import and authenticate the Root and Intermediate Trustpoint CA certificates.

Section "MACSEC and MKA Configuration" in the **Admin Guide** provides instructions for configuring pre-shared keys for use with MACsec.

Section "Zeroize Private Key" in the **Admin Guide** provides instructions for deleting the private key stored in NVRAM that is generated for SSH. Other keys stored in SDRAM are zeroized when no longer in use, zeroized with a new value of the key, or zeroized on power-cycle.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component Testing Assurance Activities: The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

The set of functions available to administrators prior to login do not include performing any cryptographic functions as specified by NDcPP22e:FIA_UIA_EXT.1-t2. The evaluator logged in as administrator and successfully issued commands to generate and delete a crypto key.

2.4.4 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1)

2.4.4.1 NDcPP22e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_SSL_EXT.1, FTA_SSL.3, FTA_TAB.1, FMT_MOF.1(1)/ManualUpdate, FMT_MOF.1(4)/AutoUpdate (if included in the ST), FIA_AFL.1, FIA_X509_EXT.2.2 (if included in the ST), FPT_TUD_EXT.1.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(2)/Services, and FMT_MOF.1(3)/Functions (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

(containing also requirements on Guidance Documentation and Tests)

The evaluator shall examine the TSS, Guidance Documentation and the TOE as observed during all other testing and shall confirm that the management functions specified in FMT_SMF.1 are provided by the TOE. The evaluator shall confirm that the TSS details which security management functions are available through which interface(s) (local administration interface, remote administration interface).

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator shall examine the TSS and Guidance Documentation to verify they both describe the local administrative interface. The evaluator shall ensure the Guidance Documentation includes appropriate warnings for the administrator to ensure the interface is local.

For distributed TOEs with the option 'ability to configure the interaction between TOE components' the evaluator shall examine that the ways to configure the interaction between TOE components is detailed in the TSS and Guidance Documentation. The evaluator shall check that the TOE behaviour observed during testing of the configured SFRs is as described in the TSS and Guidance Documentation.

See the other requirements in this AAR as referenced. All security management functions and the corresponding configuration information are identified or referenced throughout this AAR with the requirement to which they apply.

Section 6 (FMT_SMF.1) in the ST states that the TOE provides all capabilities necessary to securely manage the TOE and the services provided by the TOE. The management functionality of the TOE is provided through the TOE CLI. The Authorized Administrator can perform all management functions by accessing the TOE directly via connected console cable or remote administration via SSHv2 secure connection. The specific management capabilities listed in this section are consistent with those identified in the requirement.

Section 1.3 in the ST defines the Local Console as any IT Environment Console that is directly connected to the TOE via the Serial Console Port and is used by the TOE Administrator to support TOE administration.

Similarly, section “Operational Environment” in the **Admin Guide** defines the Local Console as any IT environment console that is directly connected to the TOE via the serial console port and is used by the TOE administrator to support TOE administration. This is sufficient guidance for the administrator to ensure that the interface is local.

Component Guidance Assurance Activities: See TSS Assurance Activities

See TSS Assurance Activities.

Component Testing Assurance Activities: The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

All of the management functions were demonstrated throughout the course of testing as part of testing all the SFRs.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.4.5 Specification of Management Functions (MACsec) - per TD0748 (MACSEC10:FMT_SMF.1/MACSEC)

2.4.5.1 MACSEC10:FMT_SMF.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the ability of the TOE to provide the management functions defined in this SFR.

Section 6 (FMT_SMF.1/MACSEC) of the ST describes the specific management capabilities available from the TOE. For MACSEC this includes:

- The ability to manage the trusted public keys database
- The ability to manage the Key Server and associated MKA participants
- The ability to manage a PSK and install in the CAK cache
- The ability to specify the lifetime of a CAK and to enable, disable or delete a PSK in the CAK cache of a device

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides instructions on how to perform each of the management functions defined in this SFR.

As addressed in the relevant SFRs throughout this document, section “MACSEC and MKA Configuration” in the **Admin Guide** provides instructions for how to perform the management functions relevant to this SFR. This includes instructions for configuring and enabling a PSK based CAK using the ‘key chain’ command, configuring the key-string and the lifetime of a CAK and configuring the key-server priority in the MKA policy to ensure that the TOE can act as the Key Server when connecting with MACsec peers.

Component Testing Assurance Activities: The evaluator shall set up an environment where the TOE can connect to two other MACsec devices, identified as devices B and C, with the ability of PSKs to be distributed between them. The evaluator shall configure the devices so that the TOE will be elected key server and principal actor, i.e., has highest key server priority.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator shall follow the relevant operational guidance to perform the tests listed below. Note that if the TOE claims multiple management interfaces, the tests should be performed for each interface that supports the functions.

Test 20: The evaluator shall connect to the PAE of the TOE and install a PSK. The evaluator shall then specify a CKN and that the PSK is to be used as a CAK.

Repeat this test for both 128-bit and 256-bit key sizes.

Repeat this test for a CKN of valid length (1-32 octets), and observe success.

Repeat this test again for CKN of invalid lengths zero and 33, and observe failure.

Test 21: (conditional, "Cause key server to generate a new group CAK..." is selected) The evaluator shall test the ability of the TOE to enable and disable MKA participants using the management function specified in the ST. The evaluator shall install PSKs in devices B and C, and take any necessary additional steps to create corresponding MKA participants. The evaluator shall disable the MKA participant on device C, then observe that the TOE can communicate with B but neither the TOE nor B can communicate with device C. The evaluator shall re-enable the MKA participant of device C and observe that the TOE is now able to communicate with devices B and C. (TD0748 applied)

Test 22: For TOEs using only PSKs, the TOE should be the key server in both tests and only one peer (B) needs to be tested. The tests are:

Test 22.1: (Conditional: The TOE supports MKA keychains with multiple CKN/CAKs) Switch to unexpired CKN: TOE and Peer B have CKN1(10 minutes) and CKN2. CKN2 can either be configured with a longer overlapping lifetime (20 minutes) or be configured with a lifetime starting period of more than 10 minutes after the CKN1 start. The TOE and Peer B start using CKN1 and after 10 minutes, verify that the TOE expires SAK1. This can be verified by either 1) seeing the TOE immediately distribute a new SAK to the peer if the lifetime of CKN2 overlaps CKN1, or 2) by terminating the connection with CKN1 and distributing a new SAK once the lifetime period of CKN2 begins.

Test 22.2: Reject CA with expired CKN: TOE has CKN1 (10 minutes). Peer B has CKN1 (20 minutes). TOE and Peer B start using CKN1 and after 10 minutes, verify that the TOE rejects (or ignores) peer's request to use (or distribute) a SAK using CKN1.

Test 23: (conditional, 'Cause key server to generate a new group CAK...' is selected) The evaluator shall connect to the PAE of the TOE, set the management function specified in the ST (e.g., set ieee8021XKayCreateNewGroup to true), and observe that the TOE distributes a new group CAK.

Test 20: The evaluator first configured a CKN with a valid length and a 128-bit CAK and established a successful connection. The evaluator repeated this for a 256-bit CAK. The evaluator then attempted to configure CKNs with lengths below the minimum and above the maximum and viewed the attempts failed as expected.

Test 21: Not applicable. Group CAKs are not supported by the TOE.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Test 22.1: The evaluator configured the TOE with CKN1, which expires in 10 minutes, and CKN2, which does not expire. The tester set up a valid MACsec channel between the TOE and the peer using CKN1. After 10 minutes, the evaluator analyzed the TOE logs and packet capture. The evaluator determined that a new SAK was distributed using CKN2.

Test 22.2: The evaluator configured the TOE with CKN1, which expires in 10 minutes, and CKN2, which expires in 20 minutes. The tester set up a valid MACsec channel between the TOE and the peer using CKN1. After 10 minutes, the evaluator analyzed the TOE logs and packet capture. The evaluator determined that a new SAK was distributed using CKN2.

Test 23: Not applicable. Group CAKs are not supported by the TOE.

2.4.6 Restrictions on Security Roles (NDcPP22e:FMT_SMR.2)

2.4.6.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.6.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.6.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Section 6 (FMT_SMR.2) in the ST states that the TOE maintains privileged and semi-privileged Administrator roles.

The TOE performs role-based authorization, using TOE platform authorization mechanisms, to grant access to TOE functions. For the purposes of this evaluation, the privileged role is equiv-alent to full administrative access to the CLI, which is the default access for IOS-XE privilege level (PL) 15. Semi-privileged roles are assigned a PL of 0 - 14. PL 0 and 1 are defined by de-fault and are customizable, while PL 2-14 are undefined by default and are also customizable. Note: Levels 0 - 14 are a subset of PL 15 and the levels are not hierarchical.

The term “Authorized Administrator” refers to any user which has been assigned to a privilege level that is permitted to perform the relevant action; therefore, has the appropriate privileges to perform the requested functions.

The privilege level determines the functions the user can perform, hence the Authorized Ad-ministrator with the appropriate privileges.

The TOE can and shall be configured to authenticate all access to the command line interface using a username and password.

Component Guidance Assurance Activities: The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration.

See FIA_UIA_EXT.1 which identifies the instructions in the **Admin Guide** for administering the TOE both locally and remotely.

Component Testing Assurance Activities: In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

Throughout the course of testing, the TOE was administered both via local CLI and SSH.

2.5 Protection of the TSF (FPT)

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.5.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1)

2.5.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Section 6 (FPT_APW_EXT.1) in the ST states that the TOE is designed specifically to not disclose any passwords stored in the TOE. All passwords are stored using a SHA-2 hash. 'Show' commands display only the hashed password. This section also explains that the CC Configuration Guide instructs the Administrator to use the algorithm-type scrypt sub-command when passwords are created or updated. The scrypt is password type 9 and uses a SHA-2 hash.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.2 Protection of CAK Data (MACSEC10:FPT_CAK_EXT.1)

2.5.2.1 MACSEC10:FPT_CAK_EXT.1.1

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how CAKs are stored and that they are unable to be viewed through an interface designed specifically for that purpose. If these values are not stored in plaintext, the TSS shall describe how they are protected or obscured.

Section 6 (FPT_CAK_EXT.1) of the ST states a CAK value is specified in the configuration file by the Administrator using a bit-based (hex) format. The interface specifically implemented in the TSF for viewing the configuration file is the "show running-config" or "show startup-config "CLI commands. When the TOE is operating in the evaluated configuration, and the Administrator executes the "show running-config" or "show startup-config" CLI commands, the CAK data will not be displayed. This protects the CAK data from unauthorized disclosure.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.3 Failure with Preservation of Secure State - per TD0816 (MACSEC10:FPT_FLS.1)

2.5.3.1 MACSEC10:FPT_FLS.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it indicates that the TSF will attain a secure/safe state (e.g., shutdown) if a self-test failure is detected. For TOEs with redundant failover capability, the evaluator shall examine the TSS to determine that it indicates that the failed components will attain a secure/safe state (e.g., shutdown) if a self-test failure is detected. (TD0816 applied)

Section 6 (FPT_FLS.1) of the ST states whenever a failure occurs (power-on self-tests, integrity check of the TSF executable image and/or the noise source health-tests) within the TOE, the TOE will attain a secure/safe state by disabling its interfaces to prevent the unintentional flow of any information to or from the TOE and reloads.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

If the failures persist, the TOE will continue to reload in an attempt to correct the failure. This functionally prevents any failure from causing an unauthorized information flow. There are no failures that circumvent this protection. If the rebooting continues, the Authorized Administrator must contact Cisco Technical Assistance Center (TAC).

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it describes the behavior of the TOE following a self-test failure and actions that an administrator should take if it occurs.

Section "Cryptographic Self-Tests" in the **Admin Guide** states that the TOE runs a suite of self-tests during initial start-up to verify correct operation of cryptographic modules. If any component reports failure for the POST, the system crashes and appropriate information is displayed on the local console. All ports are blocked from moving to forwarding state during the POST. If all components of all modules pass the POST, the system is placed in FIPS PASS state and ports are allowed to forward data traffic. If any of the tests fail, a message is displayed to the local console and the TOE component will automatically reboot. If the Administrator observes a cryptographic self-test failure, they should contact Cisco Technical Support. Refer to the Contact Cisco section of this document.

If the Administrator needs to execute cryptographic self-tests for the Switch after the image is loaded the following command can be used: SWITCH# test crypto self-test. If any of the tests fail, a message is displayed to the local console and the TOE will automatically reboot.

Component Testing Assurance Activities: The following test may require the vendor to provide access to a test platform that provides the evaluator with the ability to modify the TOE internals in a manner that is not provided to end customers:

Test 24: For each failure mode specified in the ST that can be deliberately induced, the evaluator shall ensure that the TOE attains a secure state (e.g., shutdown) after initiating each failure mode type. For TOEs with redundant failover capability, the evaluator shall determine that the failed components attain a secure state and the behavior of the TOE is consistent with the operational guidance. For each component, the evaluator shall repeat each type of self-test that can be deliberately induced to fail. (TD0816 applied)

The evaluator used special builds provided by the vendor to cause failure of the integrity check of the image and failure of each of the relevant power-on self-tests. In each case the TOE rebooted as expected due to the self-test failure.

2.5.4 Replay Detection - per TD0746 & 0869 (MACSEC10:FPT_RPL.1)

2.5.4.1 MACSEC10:FPT_RPL.1.1

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.4.2 MACSEC10:FPT_RPL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes how replay is detected for MKPDUs and MPDUs and how replayed MKPDUs and MPDUs are handled by the TSF.

Section 6 (FPT_RPL.1) of the ST states replayed data is discarded by the TOE and the attempt to replay data is logged.

The TOE ensures MPDUs are replay protected by ensuring the received 32-bit PN in the SecTAG of the frame is not less than the lowest acceptable 32-bit PN for the SA. With Extended Packet Numbering (XPN) which uses a 64-bit PN, the TOE enforces replay detection by ensuring the received 64-bit PN in the SecTAG of the frame is not less than the lowest acceptable 64-bit PN for the SA.

If the PN is less that the lowest acceptable PN for the SA, the MPDU will be dropped and not processed further. The Replay Protection Window Size determines the lowest acceptable PN for the SA and must be enabled in the evaluated configuration. The Replay Protection Window Size may be set to zero to enforce strict replay protection.

The TOE protects against replayed MKPDUs by ensuring if a MKPDU contains a duplicate Member Number (MN) and not the most current MN in the Basic Parameter set, then the MKPDU will be dropped and not processed further.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Before performing each test the evaluator shall successfully establish a MACsec channel between the TOE and a MACsec-capable peer in the operational environment sending enough traffic to see it working and verify the MN and PN values increase for each direction.

Test 25: The evaluator shall set up a MACsec connection with an entity in the operational environment. The evaluator shall then capture traffic sent from this remote entity to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the remote entity where the PN values in the SecTag of these

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

packets are less than the lowest acceptable PN for the SA. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded. (TD0746 applied)

Test 26: The evaluator will capture frames during an MKA session and record the lowest MN from the Basic Parameter set observed in a particular time range. The evaluator shall then send a frame with a lower MN, and then verify that this frame is dropped. The evaluator will verify that the device logged this event.

Test 25: The evaluator ensured replay protection was enabled on the TOE and set up a successful MACsec connection between the TOE and a test system. The evaluator captured MACsec traffic sent from the test system to the TOE and then attempted to send the same traffic, which contains an old packet number (PN). The TOE successfully detected the invalid PN and dropped the traffic along with reporting an audit log of the event. The evaluator then attempted the same test, only this time the evaluator sent MKA traffic that was already sent. The evaluator noted that the TOE successfully detected the invalid MN, dropped the traffic, and reported the error in an audit log.

Test 26: This was performed as part of test 1 above.

2.5.5 Replay Detection for XPN - per TD0728 (MACSEC10:FPT_RPL_EXT.1)

2.5.5.1 MACSEC10:FPT_RPL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.2 MACSEC10:FPT_RPL_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it includes XPN in the description of how replay is detected for MPDUs and how replayed MPDUs are handled by the TSF.

Section 6 (FPT_RPL_EXT.1) states replayed data is discarded by the TOE and the attempt to replay data is logged.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The TOE ensures MPDUs are replay protected by ensuring the received 32-bit PN in the SecTAG of the frame is not less than the lowest acceptable 32-bit PN for the SA. With Extended Packet Numbering (XPN) which uses a 64-bit PN, the TOE enforces replay detection by ensuring the received 64-bit PN in the SecTAG of the frame is not less than the lowest acceptable 64-bit PN for the SA. Extended Packet Numbering (XPN) applies to the Catalyst 9600X series models. It does not apply to the 9400CX series.

If the PN is less that the lowest acceptable PN for the SA, the MPDU will be dropped and not processed further. The Replay Protection Window Size determines the lowest acceptable PN for the SA and must be enabled in the evaluated configuration. The Replay Protection Window Size may be set to zero to enforce strict replay protection.

Component Guidance Assurance Activities: If the use of XPN or the XPN cipher suites used by the TOE are configurable, the evaluator shall examine the guidance documentation to determine that it describes how this is configured.

Section "MACSEC and MKA Configuration" in the **Admin Guide** provides instructions for configuring whether XPN is enabled with gcm-aes-xpn-128 or gcm-aes-xpn-256. Only the 9600x supports the XPN ciphers. The 9400x does not support XPN.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

Test 29: The evaluator shall establish a MACsec connection between the TOE and a test system using the GCM-AES-XPN-128 cipher suite if selected, otherwise use GCM-AES-XPN-256. The evaluator shall write or obtain a script to send a small frame with a known payload (such as five bytes of all zeroes) to the TOE. The evaluator shall activate a packet capture tool on the connection between the TOE and the test system and then use the test system to send this frame to the TOE 4,294,967,267 (2^32 + 1) times. The evaluator shall use the packet capture tool to verify that for the first and last frames sent, the least significant 32 bits are the same. This means the most significant bits should have been incremented during this test. Since the IV is different the two encrypted frames should be different.

Note that if traffic is sent to the TOE at a rate of 10 GB/s, this will take approximately 5 minutes as per IEEE 802.1AE-2018.

Test 30: If both cipher suites were selected, then the evaluator shall reconfigure the TOE using the second cipher suite and rerun Test 29 to demonstrate support for both cipher suites. (TD0728 applied)

Note: This testing was only performed on the 9600x.

Test 29: The evaluator established a MACsec connection w/ the TOE as one peer while collecting a packet capture of the MACsec traffic. This connection used GCM-AES-XPN-256. The evaluator sent a small frame 2^32+1 times to the TOE. The evaluator observed that the 32 bit Packet Number was the same in the first frame and in the last frame. This is the 32 least significant bits of the Extended Packet Number (XPN). The most significant bits are not sent in a MACsec packet. The evaluator confirmed the most significant bit of the packet number had changed because the ICV (which uses the packet number in its calculation) was different between the start and end encrypted frames despite the unencrypted data being sent being identical.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Test 30: The evaluator repeated the prior test using GCM-AES-XPN-128 and observed the same correct behavior.

2.5.6 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1)

2.5.6.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

Section 6 (FPT_SKP_EXT.1) in the ST states that the TOE is designed specifically to not disclose any keys stored in the TOE. The TOE stores all private keys in a secure directory that cannot be viewed or accessed, even by the Administrator. The TOE stores symmetric keys only in volatile memory. Pre-shared keys (MACsec CAKs) may be specified in the configuration file by the Administrator using a bit-based (hex) format. While, only the Administrator may view the configuration file, the CAKs can be configured so they are excluded from display when viewing the configuration file via “show running-config” or “show startup-config “CLI commands.

Table 20 in Section 6.1 of the ST further details the relevant keys and where they are stored.

Section 6 (FTP_CAK_EXT.1) further states a CAK value is specified in the configuration file by the Administrator using a bit-based (hex) format. The interface specifically implemented in the TSF for viewing the configuration file is the “show running-config” or “show startup-config “CLI commands. When the TOE is operating in the evaluated configuration, and the Administrator executes the “show running-config” or “show startup-config” CLI commands, the CAK data will not be displayed. This protects the CAK data from unauthorized disclosure.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.7 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1)

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.5.7.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time, and that it provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

If 'obtain time from the underlying virtualization system' is selected, the evaluator shall examine the TSS to ensure that it identifies the VS interface the TOE uses to obtain time. If there is a delay between updates to the time on the VS and updating the time on the TOE, the TSS shall identify the maximum possible delay. (TD0632 applied)

Section 6 (FPT_STM_EXT.1) in the ST states that the TSF implements a clock function to provide a source of date and time. The clock function is reliant on the system clock provided by the underlying hardware. All Switch models have a real-time clock (RTC) with battery to maintain time across reboots and power loss.

The TOE relies upon date and time information for the following security functions:

- To monitor local and remote interactive administrative sessions for inactivity (FTA_SSL_EXT.1, FTA_SSL.3);
- Validating X.509 certificates to determine if a certificate has expired (FIA_X509_EXT.1);
- To determine when SSH session keys have expired and to initiate a rekey (FCS_SSHS_EXT.1);
- To provide accurate timestamps in audit records (FAU_GEN.1.2).

Component Guidance Assurance Activities: The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the guidance documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

If the TOE supports obtaining time from the underlying VS, the evaluator shall verify the Guidance Documentation specifies any configuration steps necessary. If no configuration is necessary, no statement is necessary in the Guidance Documentation. If there is a delay between updates to the time on the VS and updating the time on the TOE, the evaluator shall ensure the Guidance Documentation informs the administrator of the maximum possible delay. (TD0632 applied)

Section "Configure Time and Date" in the **Admin Guide** provides instructions for the administrator to manually set the time on the TOE.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: If the TOE supports direct setting of the time by the Security Administrator then the evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

b) Test 2: If the TOE supports the use of an NTP server; the evaluator shall use the guidance documentation to configure the NTP client on the TOE, and set up a communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

c) Test 3: [conditional] If the TOE obtains time from the underlying VS, the evaluator shall record the time on the TOE, modify the time on the underlying VS, and verify the modified time is reflected by the TOE. If there is a delay between the setting the time on the VS and when the time is reflected on the TOE, the evaluator shall ensure this delay is consistent with the TSS and Guidance. (TD0632 applied)

Test 1: The evaluator issued commands from the local console to change the time and then observed via the console display that the time was set as intended.

Test 2: Not applicable. The TOE does not support the use of an NTP server.

Test 3: Not applicable. The TOE does not use an underlying VS.

2.5.8 TSF Testing (NDcPP22e:FPT_TST_EXT.1)

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.5.8.1 NDcPP22e:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

For distributed TOEs the evaluator shall examine the TSS to ensure that it details which TOE component performs which self-tests and when these self-tests are run.

Section 6 (FPT_TST_EXT.1) in the ST states that the TOE runs a suite of self-tests during initial start-up to verify correct operation of the cryptographic module. All ports are blocked from moving to forwarding state during the Power on Self-Test (POST). If all components of all modules pass the POST, the system is placed in FIPS PASS state and ports are allowed to forward data traffic. If any of the tests fail, the system halts and a message is displayed to the local console. These tests include:

- AES Known Answer Test: For the encrypt test, a known key is used to encrypt a known plain text value resulting in an encrypted value. This encrypted value is compared to a known encrypted value. If the encrypted texts match, the test passes; otherwise, the test fails. The decrypt test is just the opposite. In this test a known key is used to decrypt a known encrypted value. The resulting plaintext value is compared to a known plaintext value. If the decrypted texts match, the test passes; otherwise, the test fails.

- RSA Signature Known Answer Test (both signature/verification): This test takes a known plaintext value and Private/Public key pair and used the public key to encrypt the data. This value is compared to a known encrypted value. If the encrypted values, the test passes; otherwise, the test fails. The encrypted data is then decrypted using the private key. This value is compared to the original plaintext value. If the decrypted values match, the test passes; otherwise, the test fails.

- RNG/DRBG Known Answer Test: For this test, known seed values are provided to the DRBG implementation. The DRBG uses these values to generate random bits. These random bits are compared to known random bits. If the random bits match, the test passes; otherwise, the test fails.

- HMAC Known Answer Test: For each of the hash values listed, the HMAC implementation is fed known plaintext data and a known key. These values are used to generate a MAC. This MAC is compared to a known MAC. If the MAC values match, the test passes; otherwise, the test fails.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

- Software Integrity Test: The Software Integrity Test uses HMAC-SHA256 (IC2M) and HMAC-SHA-1 (CiscoSSL FOM) verification to con-firm the cryptographic module has maintained its integrity. The Software Integrity Test is run automatically when the module is loaded.

- SHA-256/384/512 Known Answer Test: For each of the values listed, the SHA implementation is fed known data and a key. These values are used to generate a hash. This hash is compared to a known value. If the hash values match, the test passes; otherwise, the test fails.

Section 6 (FPT_TST_EXT.1) in the ST further describes that if any component reports failure for the POST, the system crashes. Appropriate information is displayed on the screen and saved in the crashinfo file. All ports are blocked during the POST. If all components pass the POST, the system is placed in FIPS PASS state and ports can forward data traffic. If an error occurs during the self-test, a SELF_TEST_FAILURE system log is generated.

Section 6 (FPT_TST_EXT.1) in the ST concludes that these tests are sufficient to verify that the correct version of the TOE software is running as well as that the cryptographic operations are all performing as expected because any deviation in the TSF behaviour will be identified by the failure of a self-test.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

For distributed TOEs the evaluator shall ensure that the guidance documentation describes how to determine from an error message returned which TOE component has failed the self-test.

Section “Cryptographic Self-Tests” in the **Admin Guide** states that the TOE runs a suite of self-tests during initial start-up to verify correct operation of cryptographic modules. If any component reports failure for the POST, the system crashes and appropriate information is displayed on the local console. All ports are blocked from moving to forwarding state during the POST. If all components of all modules pass the POST, the system is placed in FIPS PASS state and ports are allowed to forward data traffic. If any of the tests fail, a message is displayed to the local console and the TOE component will automatically reboot. If the Administrator observes a cryptographic self-test failure, they should contact Cisco Technical Support. Refer to the Contact Cisco section of this document.

If the Administrator needs to execute cryptographic self-tests for the Switch after the image is loaded the following command can be used: SWITCH# test crypto self-test. If any of the tests fail, a message is displayed to the local console and the TOE will automatically reboot.

Component Testing Assurance Activities: It is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE

b) Verification of the correct operation of the cryptographic functions necessary to fulfill any of the SFRs.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Although formal compliance is not mandated, the self-tests performed should aim for a level of confidence comparable to:

a) FIPS 140-2, chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

b) FIPS 140-2, chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions. Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

The evaluator shall either verify that the self tests described above are carried out during initial start-up or that the developer has justified any deviation from this.

For distributed TOEs the evaluator shall perform testing of self-tests on all TOE components according to the description in the TSS about which self-test are performed by which component.

The relevant self-tests were all demonstrated in MACsec10:FPT_FLS.1-t1 where the evaluator used special builds provided by the vendor to cause failure of the integrity check of the image and failure of each of the relevant power-on self-tests. In each case the TOE rebooted as expected due to the self-test failure.

2.5.9 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.5.9.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.9.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.9.3 NDcPP22e:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describe how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the TSS needs to describe how and when the inactive version becomes active. The evaluator shall verify this description.

The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system firmware and software (for simplicity the term 'software' will be used in the following although the requirements apply to firmware and software). The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

If the options 'support automatic checking for updates' or 'support automatic updates' are chosen from the selection in FPT_TUD_EXT.1.2, the evaluator shall verify that the TSS explains what actions are involved in automatic checking or automatic updating by the TOE, respectively.

For distributed TOEs, the evaluator shall examine the TSS to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component. Alternatively, this description can be provided in the guidance documentation. In that case the evaluator should examine the guidance documentation instead.

If a published hash is used to protect the trusted update mechanism, then the evaluator shall verify that the trusted update mechanism does involve an active authorization step of the Security Administrator, and that download of the published hash value, hash comparison and update is not a fully automated process involving no active authorization by the Security Administrator. In particular, authentication as Security Administration according to FMT_MOF.1/ManualUpdate needs to be part of the update process when using published hashes.

Section 6 (FPT_TUD_EXT.1) in the ST states that an Authorized Administrator can query the software version running on the TOE and can initiate updates to (replacements of) software images. The current active version can be verified by executing the “show version” command from the TOE’s CLI. When software updates are made available by Cisco, an Administrator can obtain, verify the integrity of, and install the updates. Trusted updates can be installed on the TOE in one shot or as a multi-stage process with a delayed activation. The inactive version will become active when the Administrator responds ‘y’ at the reboot prompt. The updates can be downloaded from software.cisco.com.

The TOE will authenticate the image using a digital signature verification check to ensure it has not been modified since distribution using the following process: Prior to being made publicly available, the software image is hashed

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

using a SHA512 algorithm and then digitally signed. The digital signature is embedded to the image (hence the image is signed The TOE uses a Cisco public key to validate the digital signature to obtain the SHA512 hash during the first stage in the install process. The TOE then computes its own hash of the image using the same SHA512 algorithm and verifies the computed hash against the embedded hash. If they match the image has not been modified or tampered since distributed from Cisco meaning the software is authenticated and the image is ready to be activated automatically in the one stage upgrade or by the administrator in the multistage upgrade. If they do not match the image will not install.

The TOE does not support automatic checking for updates, and is not distributed. Published hash is not used.

Component Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the guidance documentation needs to describe how to query the loaded but inactive version.

The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

If a published hash is used to protect the trusted update mechanism, the evaluator shall verify that the guidance documentation describes how the Security Administrator can obtain authentic published hash values for the updates.

For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT_TUD_EXT.1, how all TOE components are updated, and the error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. The guidance documentation only has to describe the procedures relevant for the Security Administrator; it does not need to give information about the internal communication that takes place when applying updates.

If this was information was not provided in the TSS: For distributed TOEs, the evaluator shall examine the Guidance Documentation to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component.

If this was information was not provided in the TSS: If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the Guidance Documentation contains a description of how the certificates are contained on the device. The evaluator also ensures that the Guidance Documentation describes how the certificates are installed/updated/selected, if necessary.

Section “Verify TOE Software” in the **Admin Guide** provides the “show version” command which is used to view the currently active version.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Section "Update TOE Software" in the **Admin Guide** provides instructions for updating the TOE software including where to obtain the updated software image and how to upload the image to the switch to make it available for installation and activation or alternatively performing all three steps in one stage using the "install add file [tftp | ftp | sftp://<IP Address of TFTP/FTP/SFTP server>] <image name.bin> activate commit"   command.

The TOE will automatically verify the integrity of the stored image when loaded for execution. The TOE uses a Cisco public key to validate the digital signature to obtain an embedded SHA512 hash that was generated prior to the image being distributed from Cisco. The TOE then computes its own hash of the image using the same SHA512 algorithm and verifies the computed hash against the embedded hash. If they match the image is authenticated and has not been modified or tampered with and can proceed with installation. If they do not match the image will not boot or execute. After boot, the authorized administrator can also manually verify the digital signature by executing the 'verify bootflash:<image or package name> ' command.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: The evaluator performs the version verification activity to determine the current version of the product. If a trusted update can be installed on the TOE with a delayed activation, the evaluator shall also query the most recently installed version (for this test the TOE shall be in a state where these two versions match). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

b) Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below, and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

1) A modified version (e.g. using a hex editor) of a legitimately signed update

2) An image that has not been signed

3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

4) If the TOE allows a delayed activation of updates the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

c) Test 3 [conditional]: If the TOE itself verifies a hash value over an image against a published hash value (i.e. reference value) that has been imported to the TOE from outside such that the TOE itself authorizes the installation of an image to update the TOE, the following test shall be performed (otherwise the test shall be omitted). If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

1) The evaluator obtains or produces an illegitimate update such that the hash of the update does not match the published hash. The evaluator provides the published hash value to the TOE and calculates the hash of the update either on the TOE itself (if that functionality is provided by the TOE), or else outside the TOE. The evaluator confirms that the hash values are different, and attempts to install the update on the TOE, verifying that this fails because of the difference in hash values (and that the failure is logged). Depending on the implementation of the TOE, the TOE might not allow the Security Administrator to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE.

2) The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value to the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to the TOE, e.g. if the hash value needs to be handed over to the TOE as a parameter in a command line message and the syntax check of the command prevents the execution of the command without providing a hash value. In that case the mechanism that prevents the execution of this check shall be tested accordingly, e.g. that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE.

3) If the TOE allows delayed activation of updates, the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs. Depending on the point in time when the attempted update is rejected, the most recently installed version might or might not be updated. The evaluator shall verify that the TOE

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 3 shall be skipped.

The evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all methods supported (manual updates, automatic checking for updates, automatic updates).

For distributed TOEs the evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all TOE components.

Test 1: The evaluator verified the current TOE version and then followed guidance procedures to load a new install image to the TOE but not yet activate it. At this point, the evaluator issued the “show install summary” command to show that the current version matches the active image, and the update version is shown as an inactive image. The evaluator then proceeded to activate the image which required a reboot to finish the installation process. Upon installation completion and successfully reboot, the evaluator verified that the TOE’s version was updated successfully.

Test 2: The evaluator attempted to install the following forms of illegitimate updates:

1. Corrupted Image/Valid Signature - A byte in the update file is modified via a hex editor

2. No Signature - Image is missing a signature

3. Invalid Signature – A byte in the image’s signature is modified via a hex editor

Attempts to update with each of these modified images failed as expected. The evaluator first verified the current TOE version. The evaluator then attempted to load an invalid image onto the TOE. An error message was output to the console and the image failed to load. The evaluator verified that the TOE version remained unchanged. The evaluator also verified that the TOE’s installed version matches the current running version of the TOE.

Test 3: Not applicable. The TOE does not verify the integrity of updates using published hashes.

2.6 TOE ACCESS (FTA)

2.6.1 TSF-INITIATED TERMINATION (NDcPP22e:FTA_SSL.3)

2.6.1.1 NDcPP22e:FTA_SSL.3.1

TSS Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the administrative remote session termination and the related inactivity time period.

Section 6.(FTA_SSL.3) in the ST states that the Administrator can configure maximum inactivity times individually for both local and remote administrative sessions. If either the local or remote administrative sessions are inactive for a configured period of time, the session will be terminated and will require re-authentication. An Authorized Administrator can configure the maximum inactivity times using the "exec-timeout" setting applied to the console and virtual terminal (vty) lines. The allowable inactivity timeout range is from is <0-35791> minutes. A value of 0 means there is no inactivity timeout enforced.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

Section "Session Termination" in the **Admin Guide** provides instructions for configuring the inactivity time period for remote administrative session termination. A value of 0 means there is no inactivity timeout enforced.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

The evaluator configured an inactivity timeout of 1 minute on the TOE, and after one minute of no activity, the TOE disconnected as expected. The evaluator then repeated this with an inactivity timeout of 2 minutes, and again the TOE disconnected due to inactivity at the expected time.

2.6.2 User-initiated Termination (NDcPP22e:FTA_SSL.4)

2.6.2.1 NDcPP22e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how the local and remote administrative sessions are terminated.

Section 6 (FTA_SSL.4) in the ST states that an Authorized Administrator can exit out of both local and remote administrative sessions by issuing the 'exit' or 'logout' command.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Throughout the **Admin Guide** there are numerous examples of the administrator using the 'exit' command to exit out of both local and remote administrative sessions.

The "SSH Remote Administration Protocol" section in the **Admin Guide** indicates that logging out from the CLI session can be performed using either the "exit or "logout" command. Throughout the **Admin Guide** there are numerous examples of the administrator using the 'exit' command to exit out of both local and remote administrative sessions. During testing the 'logout' command was also demonstrated.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following tests:

a) Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

b) Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

Test 1 & Test 2: The evaluator logged into the TOE via an interactive local session and proceeded to issue a logout command, after which the TOE logged out of the session as expected. The evaluator then performed the same actions via a remote SSH session, and once again observed the session terminating as expected.

2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.6.3.1 NDcPP22e:FTA_SSL_EXT.1.1

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details whether local administrative session locking or termination is supported and the related inactivity time period settings.

local administrative session locking or termination is supported and the related inactivity time period settings.

Section 6 (FTA_SSL_EXT.1) in the ST states that the Administrator can configure maximum inactivity times individually for both local and remote administrative sessions. If either the local or remote administrative sessions are inactive for a configured period of time, the session will be terminated and will require re-authentication. The local interactive session terminates and does not lock. An Authorized Administrator can configure the maximum inactivity times using the “exec-timeout” setting applied to the console and virtual terminal (vty) lines. The allowable inactivity timeout range is from is <0-35791> minutes. A value of 0 means there is no inactivity timeout enforced.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states whether local administrative session locking or termination is supported and instructions for configuring the inactivity time period.

Section “Session Termination” in the **Admin Guide** provides instructions for configuring local admin session termination after a specified time period of inactivity by using the exec-timeout command applied to the console lines. A value of 0 means there is no inactivity timeout enforced.

Component Testing Assurance Activities: The evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that reauthentication is needed when trying to unlock the session.

Test 1: The evaluator configured an idle timeout of 1 minute and observed the TOE terminate the session after the expected time had elapsed. The evaluator then repeated this with an idle timeout of 2 minutes and again observed the TOE terminate the session at the expected time.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.6.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.6.4.1 NDcPP22e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access, and might be configured during initial configuration (e.g. via configuration file).

Section 6 (FTA_TAB.1) in the ST states that the Administrator can configure an access banner that describes restrictions of use, legal agreements, or any other appropriate information to which users consent by accessing the TOE. The banner will display on the local console port and SSH interfaces prior to allowing any administrative access. Please refer to the TSS assurance activities in FIA_UIA_EXT.1 where these administrative methods of access are described in detail.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Section “Access Banner” in the **Admin Guide** provides instructions for configuring the banner message.

Component Testing Assurance Activities: The evaluator shall also perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

The evaluator configured a banner and verified that the banner was displayed appropriately for console and SSH CLI logins.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.7 Trusted path/channels (FTP)

2.7.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1)

2.7.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 NDcPP22e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Section 6 (FTP_ITC.1) of the ST states the TOE uses secure protocols to provide trusted communications between itself and authorized IT entities

For Syslog Server the TOE acts as a client, with communication secured by TLS and the non-TSF endpoint identification is done using X.509 certificates. This aligns with the SFR claims of FCS_TLSC_EXT.1 and FIA_X509_EXT.1/.2.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

Section "TLS - Syslog" in the **Admin Guide** provides instructions for configuring and establishing TLS connections between the TOE and a remote audit server.

Section "TLS Syslog Server Interruption and Recovery" in the **Admin Guide** provides instructions for if the TLS connection to the Syslog Server is unexpectedly interrupted. The TLS session will be reestablished once communication to the Syslog Server is available again.

Component Testing Assurance Activities: The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

c) Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

d) Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Further assurance activities are associated with the specific protocols.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public- facing document or report.

Test 1-3: The evaluator followed the guidance documentation to configure a secure TLS connection between the TOE and an external test server that served as the syslog a server. The evaluator observed and confirmed via the packet captures that the TOE initiated the connection to the syslog server in order to transmit audit records. The evaluator also observed that the channel data was encrypted and no channel data was sent in plaintext.

Test 4: The evaluator started a packet capture and established a TLS connection between the TOE and the external test server. The evaluator then initiated the physical interruption of the connection for roughly 1 minute and then restored the connection. The packet capture shows no traffic between the TOE and external server during the disruption. Upon reconnection the TOE continued to use the TLS sessions and did not establish a new TLS handshake between the TOE and the test server. The evaluator repeated this disruption test for a period of roughly 6 minutes. Upon reconnection this time, the evaluator observed that the TOE initiated a new TLS handshake to establish a new TLS session between the TOE and the test server.

2.7.2 Inter-TSF Trusted Channel (MACsec Communications) (MACSEC10:FTP_ITC.1/MACSEC)

2.7.2.1 MACSEC10:FTP_ITC.1.1/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.2.2 MACSEC10:FTP_ITC.1.2/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

2.7.2.3 MACSEC10:FTP_ITC.1.3/MACSEC

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.7.3 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin)

2.7.3.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.3.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.3.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

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

Section 6 (FTP_TRP.1/Admin) in the ST states that all remote administrative communications take place over a secure encrypted SSHv2 session. The SSHv2 session is encrypted using AES encryption. The remote users (Authorized Administrators) can initiate SSHv2 communications with the TOE.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Section "SSH Remote Administration Protocol" in the **Admin Guide** provides instructions for establishing the remote administrative sessions using SSH.

Component Testing Assurance Activities: The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: The evaluator shall ensure, for each communication channel, the channel data is not sent in plaintext.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

The successful testing of the remote administration channel and the demonstration of its encryption can be found in FCS_SSHS_EXT.1. The evaluator verified that the results from the FCS_SSHS_EXT.1 tests were using the correct protocol and that there was no channel data being sent in plaintext.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent is that these interfaces will be adequately tested, and

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

It should be noted that there may be some SFRs that do not have an interface that is explicitly 'mapped' to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

For this PP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

3.2 GUIDANCE DOCUMENTS (AGD)

3.2.1 OPERATIONAL USER GUIDANCE (AGD_OPE.1)

Assurance Activities: The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps (per TD0536):

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

In addition, the evaluator performs the EAs specified below.

The evaluator shall ensure the Operational guidance documentation is distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

In addition the evaluator shall ensure that the following requirements are also met.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes instructions that describe at least one method of validating the hash/digital signature.

c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

The **Admin Guide** provides instructions for configuring FIPS mode such that only approved cryptographic algorithms and parameters are available. FIPS mode of operation must be enabled in order for the TOE to be

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

operating in its evaluated configuration. There are other areas throughout the **Admin Guide** that define which functions are allowed and which are not allowed in the evaluated configuration. Section “Excluded Functionality” makes it clear that Non-FIPS mode is excluded and states that use of other cryptographic engines beyond what is required for the TOE was not evaluated or tested during the CC evaluation. Section “Disable Unused Protocols” provides instructions for disabling protocols that are not included in the evaluated configuration. All sections in the **Admin Guide** which describe configuring TOE security functions include only those settings that should be enabled and can be used in the evaluated configuration.

The process for updating the TOE is described above in NDcPP22e:FPT_TUD_EXT.1.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

Preparative procedures are distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

In addition, the evaluator performs the EAs specified below.

The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

In addition the evaluator shall ensure that the following requirements are also met.

The preparative procedures must

a) include instructions to provide a protected administrative capability; and

b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

The evaluation team had the following documents to use when configuring the TOE:

- Cisco Catalyst 9400X/9600X Series Switches 17.12 CC Configuration Guide, Version 0.3, September 11, 2024 (**Admin Guide)**

In some instances, the document referenced general Cisco manuals which the evaluation could find on the Cisco web site. The completeness of the documentation is addressed by their use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a 'hard label' (e.g., stamped into the metal, paper label) or a 'soft label' (e.g., electronically presented when queried).

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator performs the CEM work units associated with ALC_CMC.1.

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

See section 3.3.1 above for an explanation of how all CM items are addressed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes 'evaluated configuration' instructions). The focus of the testing is to confirm that the requirements specified in Section 5.1.7 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2, 3 and 4.

The evaluator should consult Appendix B when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section B.4.3.1.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The test configuration consisted of the following TOE platforms along with supporting products.

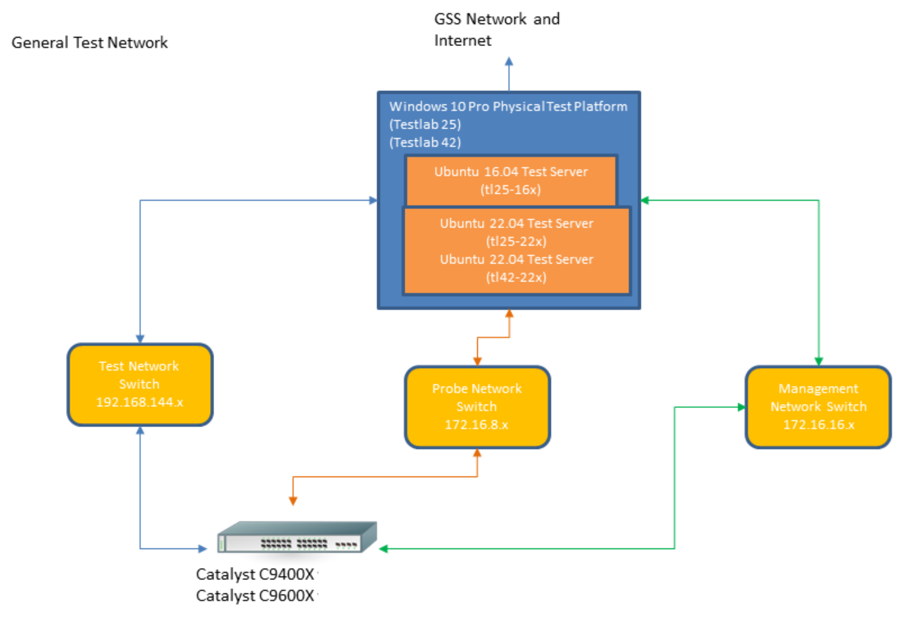
General Test Network
GSS Network and Internet
Windows 10 Pro Physical Test Platform (Testlab 25) (Testlab 42)
Ubuntu 16.04 Test Server (tl25-16x)
Ubuntu 22.04 Test Server (tl25-22x)
Ubuntu 22.04 Test Server (tl42-22x)
Test Network Switch 192.168.144.x
Probe Network Switch 172.16.8.x
Management Network Switch 172.16.16.x
Catalyst C9400X
Catalyst C9600X

Figure 1 General Test Setup

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

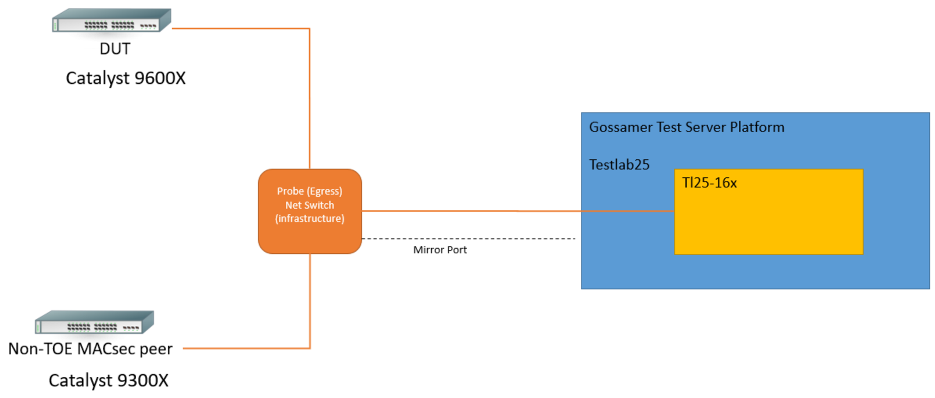

Figure 2 XPN Test Setup for FPT_RPL_EXT.1

TOE Platforms:

- Catalyst 9600X (C9606R)
- Catalyst 9400X (C9407R)

Supporting Products:

- Catalyst 9300X (used as a Macsec peer during the FPT_RPL_EXT.1 tests for throughput capability)

- Testlab25 (Windows 10 Pro)
- Tl25-16x (Ubuntu 16.04)
- Tl25-22x (Ubuntu 22.04)
- Tl42-16x (Ubuntu 16.04)
- Tl42-22x (Ubuntu 22.04)

Supporting Software:

- SSH Client – Putty version 6.2
- SSH Client – SecureCRT version 5.1.2
- Big Packet Putty version 6.2
- Wireshark version 4.0
- Nmap version 7.01
- Wpa_supplicant v2.10
- Stunnel4 3.5.0
- Scapy version 2.4.5
- Rsyslog version 8.16.0
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Intel DPDK version 24.07 (for XPN testing)
- PacketBatch-DPDK (for XPN testing)

3.5 Vulnerability Assessment (AVA)

Commented [A1]: update

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A, while an 'outline' of the assurance activity is provided below.

In addition to the activities specified by the CEM in accordance with Table 2, the evaluator shall perform the following activities.

The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic libraries, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis. (TD0547 applied)

If the TOE is a distributed TOE then the developer shall provide:

a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]

b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuzz testing. None of the public search for vulnerabilities, or the fuzz testing uncovered any residual vulnerability.

The evaluation team performed a public search on 9/11/24 for vulnerabilities in order to ensure there are no publicly known and exploitable vulnerabilities in the TOE from the following sources:

- National Vulnerability Database (https://web.nvd.nist.gov/vuln/search)
- Vulnerability Notes Database (http://www.kb.cert.org/vuls/)
- Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities)
- Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search)
- Offensive Security Exploit Database (https://www.exploit-db.com/)

Each site was searched using the following terms:

- SSH
- TLS
- Cisco IOS XE
- IOS-XE 17.12
- Cisco Catalyst
- Catalyst 9400X
- Catalyst 9600X
- C9404R
- C9407R
- C9410R
- C9606R
- Intel Xeon D-1548
- Intel Xeon D-1573N
- Intel Broadwell
- IOS Common Cryptographic Module
- IC2M
- IC2M Rel5a
- CiscoSSL FIPS Object Module
- CiscoSSL FOM 7.2a

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.

- CDR5M PHY
- MACsec
- MACsec Controller
- MSC
- UADP 2.0
- Unified Access Data Plane

3.5.2 Additional Flaw Hypothesis (AVA_VAN.1)

Assurance Activities: The following additional tests shall be performed:1.) [Conditional]: If the TOE is a TLS server and supports ciphersuites that use RSA transport (e.g. supporting TLS_RSA_WITH_* ciphers) the following test shall be performed. Where RSA Key Establishment schemes are claimed and especially when PKCS#1 v1.5* padding is used, the evaluators shall test for implementation flaws allowing Bleichenbacher and Klima et al. style attacks, including Bock et al's ROBOT attacks of 2017 in the flaw analysis. Even though Bleichenbacher's original paper is two decades old, Bock et al. found these attacks to still be effective in weakening the security of RSA key establishment in current products. Bleichenbacher and Klima et al. style attacks are complex and may be difficult to detect, but a number of software testing tools have been created to assist in that process. The iTC strongly recommends that at least one of the tools mentioned in Bock et al's ROBOT attacks of 2017 webpage or paper, as effective to detect padding oracle attacks, be used to test TOE communications channels using RSA based Key Establishment (related sources: http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf, https://eprint.iacr.org/2003/052, https://robotattack.org/). Network Device Equivalency Consideration.

The TOE does not support a TLS server implementation and therefore is not subject to Bleichenbacher attacks.

© 2024 Gossamer Security Solutions, Inc.
All rights reserved.