www.GossamerSec.com

Assurance Activity Report for Aruba Mobility Controller with ArubaOS 8.10

Version 1.0
05/19/23

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-11333

Revision History

Revision	Date	Authors	Summary
Version 0.1	04/28/23	Gossamer	Initial draft
Version 1.0	05/19/23	Cummins	Doc references updated

The TOE Evaluation was Sponsored by:

Aruba, a Hewlett Packard Enterprise Company
3333 Scott Blvd
Santa Clara, CA 95054

Evaluation Personnel:

- Tyler Catterton
- Cody Cummins
- Douglas Kalmus

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

Table of Contents

1. Introduction

This document presents evaluations results of the Aruba Mobility Controller with ArubaOS 8.10 NDcPP22e/STFFW14e/VPNGW12/WLANAS10 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 Security Target identifies the models shown below. These models fall into three product Series: the Aruba 7000 Series, the Aruba 7200 Series, and the Aruba 9000 series. The difference in these series is primarily in the number of access points and users they can support. This manifests in slightly different supporting hardware for the larger number of access points and capacity available in the larger Series 7200 equipment. The 9000 series offers a smaller form factor, which translates into fewer APs and users supported.

The Security Target also identifies three models of Aruba Mobility Controller Virtual Appliances as being part of the evaluated configuration: MC-VA-50, MC-VA-250 and MC-VA-1k. These differ primarily in performance providing higher client capacity. The Virtual Appliances can execute on a set of VM Platforms all of which are running ESXi version 7.0. The difference between these VM platforms are processor and memory capacity. The same Virtual Appliance image runs on all platforms with ESXi 7.0.

The TOE includes the following hardware and virtual appliance models:

Mobility Controller Hardware Appliances

Product Model	CPU
Aruba 9004 Mobility Controller	Intel Atom C3508 (Denverton)
Aruba 9012 Mobility Controller	Intel Atom C3508 (Denverton)
Aruba 9240 Mobility Controller	Intel Atom C3508 (Denverton)
Aruba 7005 Mobility Controller	Broadcom XLP208 (MIPS64)
Aruba 7008 Mobility Controller	Broadcom XLP208 (MIPS64)
Aruba 7010 Mobility Controller	Broadcom XLP208 (MIPS64)
Aruba 7024 Mobility Controller	Broadcom XLP208 (MIPS64)
Aruba 7030 Mobility Controller	Broadcom XLP208 (MIPS64)
Aruba 7205 Mobility Controller	Broadcom XLP316 (MIPS64)
Aruba 7210 Mobility Controller	Broadcom XLP416 (MIPS64)
Aruba 7220 Mobility Controller	Broadcom XLP432 (MIPS64)
Aruba 7240 Mobility Controller	Broadcom XLP432 (MIPS64)
Aruba 7240XM Mobility Controller	Broadcom XLP432 (MIPS64)

Product Model	CPU
Aruba 7280 Mobility Controller	Broadcom XLP780 (MIPS64)

Mobility Controller Virtual Appliances
The Mobility Controller Virtual Appliances are deployed on ESXi version 7.0. The following virtual machine platforms are included in the evaluated configuration:

Name	CPU	Memory
HPE EdgeLine EL8000	Intel Xeon Gold 6212U (Cascade Lake)	128GB
Pacstar 451/3	Intel Xeon D (Coffee Lake)	32GB
Pacstar 451/3	Intel Xeon E3 (Coffee Lake)	32GB
GTS NXGEN-L11/12	Intel Core i7 (Coffee Lake)	16GB

All Aruba Mobility Controller and Virtual Mobility Controller models run the same ArubaOS 8.10 software and include the same ArubaOS Crypto Module. Regardless of the different hardware and virtual platforms, the security functionality remains the same. The differences in the platforms are in the processing speed, throughput, memory capacity, storage, physical interfaces, number of ports, etc., and are based on performance and scalability requirements. All models run the same code with the only differences being the hardware specific code for the differently scaled hardware and the virtual nature of the hardware ports on the VMs.

The evaluator tested an Aruba 7030, Aruba 7220, Aruba 9004/9012, Aruba9240, and Pacstar 451/3. Each device uses a unique software image that is compiled for the respective family of devices. For example, the software image for the Aruba7220 is designed to execute on all Aruba 72xx platforms, and the software image for the Pacstar 451/3 is designed to run on all of the VM platforms. The images all perform the same functions, and the evaluator chose the devices to cover all of the different compiled software images.

The test results make a distinction between the 9004 and 9012. Due to a hardware failure during Firewall testing, the 9012 was swapped out with the 9004 at this point in the evaluation. These devices are largely the same, utilizing the same processor (Intel Atom C3508 (Denverton)) and running on the same Aruba 90xx system image. Additionally, the device network interfaces and drivers are the same between the two models. As a result, this Test Report uses results for the two models interchangeably. The evaluation team ensured that the full range of testing was completed on the Aruba 90xx system image. The devices were both configured to use the same IP addresses, but have different MAC addresses.

Both the controllers and Access Points support wired IPsec connections. These IPsec channel leverages the same code across all devices. Therefore, IPsec was fully tested on the controllers for each unique software image, and since the channels are equivalent only a subset of a testing was performed for the WLAN IPsec connections between the controller and access point. An example IPsec connection between the Access Point (AP655) and a controller (Aruba 9240) was shown in Test Case WLANAS10:FTP_ITC.1-t1.

1.1.2 CAVP Certificate Equivalence

The TOE includes two cryptographic modules that provide supporting cryptographic functions. The ArubaOS Crypto Module version 1.0 provides all of the cryptographic functions implemented for IKEv2/IPsec, while the ArubaOS OpenSSL Module version 1.0 provides all other cryptographic functions implemented in the TOE including those for IKEv1/IPsec, TLS and SSH.

The evaluated configuration requires that the TOE be configured in FIPS mode to ensure that the CAVP tested algorithms are used. The following functions have been CAVP certified:

Requirements	Functions	Standards	Cert
	Cryptographic key generation		
FCS_CKM.1 FCS_CKM.1(1) [VPN]	RSA schemes using cryptographic key sizes of 2048-bit or greater	FIPS Pub 186-4	A2690 A2689
FCS_CKM.1 FCS_CKM.1(1) [VPN]	ECC schemes using 'NIST curves' P-256, P-384	FIPS Pub 186-4	A2690 A2689
FCS_CKM.1	FFC schemes using cryptographic key sizes of 2048-bit or greater	FIPS Pub 186-4	A2690
	Cryptographic key establishment/distribution		
FCS_CKM.2	RSA-based key establishment schemes	RFC 8017 PKCS #1	**Vendor Affirmed**
FCS_CKM.2	Elliptic curve-based key establishment schemes	NIST SP 800-56A	A2690 A2689

Requirements	Functions	Standards	Cert
FCS_CKM.2	Finite field-based key establishment schemes	NIST SP 800-56A	A2690
	Encryption/Decryption		
FCS_COP.1/ DataEncryption	AES CBC (128, 192 and 256 bits)	FIPS Pub 197 NIST SP 800-38A	A2690 A2689
FCS_COP.1/ DataEncryption	AES GCM (128 and 256 bits)	FIPS Pub 197 NIST SP 800-38D	A2690 A2689
FCS_COP.1/ DataEncryption	AES CTR (128 and 256 bits)	FIPS Pub 197 NIST SP 800-38D	A2690
	Cryptographic signature services		
FCS_COP.1/SigGen	RSA Digital Signature Algorithm (rDSA) (modulus 2048)	FIPS Pub 186-4	A2690 A2689
FCS_COP.1/SigGen	Elliptic Curve Digital Signature Algorithm (ECDSA) with an elliptical curve size of 256 or 384 bits	FIPS Pub 186-4	A2690 A2689
	Cryptographic hashing		
FCS_COP.1/Hash	SHA-1/256/384/512 (digest sizes 160, 256, 384 and 512 bits)	FIPS Pub 180-3	A2690 A2689
	Keyed-hash message authentication		
FCS_COP.1/ KeyedHash	HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512 (key and	FIPS Pub 198-1 FIPS Pub 180-3	A2690 A2689

Requirements	Functions	Standards	Cert
	output MAC sizes 160, 256, 384, and 512, respectively)		
	Random bit generation		
FCS_RBG_EXT.1	CTR_DRBG (AES) with HW based noise sources (256 bits)	NIST SP 800-90	A2690

1.2 References

The following evidence was used to complete the Assurance Activities:

- Aruba Mobility Controller with ArubaOS 8.10 Security Target, Version 0.5, 05/19/2023 **(ST)**
- Aruba OS 8.10 Supplemental Guidance (Common Criteria Configuration Guidance), Version 2.0, April 2023 **(Admin Guide)**

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 6.1 of the ST states that for the administrative task of generating/import of, changing, or deleting of cryptographic keys, the key is uniquely identified in the audit records by the name assigned to it during import.

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 "FAU_GEN.1" of the Admin Guide lists all of the auditable events and provides a sample of each required audit record. The sample audit records do not contain the year in the timestamp, however, the guide provides instructions for configuring the logging to ensure that the required information in the timestamp including year, month, day, hour, minute, and seconds is included. An example audit with the full timestamp is provided. The guidance states that all administrative actions are audited by the TOE. The TOE creates syslog entries for all commands and configuration changes that alter system behavior and includes the username of the user making the change and the location of the user. From a review of the ST, the Admin Guide and through testing, the evaluator determined that the list of auditable events includes all administrative actions related to TSF data configuration changes. 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.

The following table from the ST shows which events are audited by the TOE.

Requirement	Auditable Events	Additional Content
NDcPP22e/VPNGW12/STFFW14/WLANAS10: FAU_GEN.1	None	None
NDcPP22e:FAU_GEN.2	None	None
NDcPP22e:FAU_GEN_EXT.1	None	None
NDcPP22e:FAU_STG.1	None	None
NDcPP22e:FAU_STG_EXT.1	None	None
NDcPP22e:FCS_CKM.1	None	None
VPNGW12:FCS_CKM.1/IKE	None	None

WLANAS10:FCS_CKM.1/WPA	None	None
NDcPP22e:FCS_CKM.2	None	None
WLANAS10:FCS_CKM.2/GTK	None	None
WLANAS10:FCS_CKM.2/PMK	None	None
NDcPP22e:FCS_CKM.4	None	None
NDcPP22e:FCS_COP.1/DataEncryption	None	None
VPNGW12:FCS_COP.1/DataEncryption	None	None
WLANAS10:FCS_COP.1/DataEncryption	None	None
NDcPP22e:FCS_COP.1/Hash	None	None
NDcPP22e:FCS_COP.1/KeyedHash	None	None
NDcPP22e:FCS_COP.1/SigGen	None	None
NDcPP22e:FCS_HTTPS_EXT.1	Failure to establish a HTTPS Session.	Reason for failure.
NDcPP22e/VPNGW12:FCS_IPSEC_EXT.1	Failure to establish an IPsec SA.	Reason for failure.
	Session Establishment with peer	Entire packet contents of packets transmitted/received during session establishment
	Protocol failures. Establishment or Termination of an IPsec SA.	Reason for failure. Non-TOE endpoint of connection. Non-TOE endpoint of connection.
NDcPP22e:FCS_NTP_EXT.1	Configuration of a new time server Removal of configured time server	Identity if new/removed time server
NDcPP22e:FCS_RBG_EXT.1	None	None
NDcPP22e:FCS_SSHS_EXT.1	Failure to establish an SSH session.	Reason for failure.

NDcPP22e:FCS_TLSS_EXT.1	None	None
STFFW14e:FDP_RIP.2	None	None
STFFW14e:FFW_RUL_EXT.1	Application of rules configured with the 'log' operation	Source and destination addresses Source and destination ports Transport Layer Protocol TOE Interface
WLANAS10:FIA_8021X_EXT.1	Attempts to access the 802.1X controlled port prior to successful completion of the authentication exchange. Failed authentication attempt.	Provided client identity (e.g. Media Access Control [Media Access Control (MAC)] address). Provided client identity (e.g. MAC address).
NDcPP22e:FIA_AFL.1	Unsuccessful login attempt limit is met or exceeded.	Origin of the attempt (e.g., IP address).
NDcPP22e:FIA_PMG_EXT.1	None	None
VPNGW12:FIA_PSK_EXT.1	None	None
WLANAS10:FIA_PSK_EXT.1	None	None
VPNGW12:FIA_PSK_EXT.2	None	None
VPNGW12:FIA_PSK_EXT.3	None	None
WLANAS10:FIA_UAU.6	Attempts to re-authenticate.	Origin of the attempt (e.g., IP address).
NDcPP22e:FIA_UAU.7	None	None
NDcPP22e:FIA_UAU_EXT.2	All use of identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
NDcPP22e:FIA_UIA_EXT.1	All use of identification and authentication mechanism.	Origin of the attempt (e.g., IP address).
NDcPP22e:FIA_X509_EXT.1/Rev	Unsuccessful attempt to validate a certificate. Any addition,	Reason for failure of certificate validation

	replacement or removal of trust anchors in the TOE's trust store	Identification of certificates added, replaced or removed as trust anchor in the TOE's trust store
NDcPP22e/VPNGW12:FIA_X509_EXT.2	None	None
NDcPP22e:FIA_X509_EXT.3	None	None
NDcPP22e:FMT_MOF.1/Functions	None	None
NDcPP22e:FMT_MOF.1/ManualUpdate	Any attempt to initiate a manual update.	None
NDcPP22e:FMT_MOF.1/Services	None	None
NDcPP22e:FMT_MTD.1/CoreData	None	None
NDcPP22e:FMT_MTD.1/CryptoKeys	None	None
VPNGW12:FMT_MTD.1/CryptoKeys	None	None
NDcPP22e/VPNGW12/STFFW14/WLANAS10: FMT_SMF.1	All management activities of TSF data. All management activities of TSF data (including creation, modification and deletion of firewall rules).	None
NDcPP22e:FMT_SMR.2	None	None
WLANAS10:FMT_SMR_EXT.1	None	None
VPNGW12:FPF_RUL_EXT.1	Application of rules configured with the 'log' operation	Source and destination addresses Source and destination ports Transport Layer Protocol
NDcPP22e:FPT_APW_EXT.1	None	None
WLANAS10:FPT_FLS.1	Failure of the TSF.	Indication that the TSF has failed with

		the type of failure that occurred.
VPNGW12:FPT_FLS.1/SelfTest	None	None
NDcPP22e:FPT_SKP_EXT.1	None	None
NDcPP22e:FPT_STM_EXT.1	Discontinuous changes to time - either Administrator actuated or changed via an automated process. (Note that no continuous changes to time need to be logged. See also application note on FPT_STM_EXT.1)	For discontinuous changes to time: The old and new values for the time. Origin of the attempt to change time for success and failure (e.g., IP address).
NDcPP22e:FPT_TST_EXT.1	None	None
VPNGW12:FPT_TST_EXT.1	None	None
WLANAS10:FPT_TST_EXT.1	Execution of TSF self-test. Detected integrity violations.	None. The TSF code file that caused the integrity violation.
VPNGW12:FPT_TST_EXT.3	None	None
NDcPP22e/VPNGW12:FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure).	None
NDcPP22e:FTA_SSL.3	The termination of a remote session by the session locking mechanism.	None
NDcPP22e:FTA_SSL.4	The termination of an interactive session.	None
NDcPP22e:FTA_SSL_EXT.1	(if 'lock the session' is selected) Any attempts at unlocking of an interactive session. (if 'terminate the session' is selected) The termination of a local session by the session locking mechanism.	None
NDcPP22e:FTA_TAB.1	None	None

VPNGW12:FTA_TSE.1	None	None
WLANAS10:FTA_TSE.1	Failure of the TSF.	Indication that the TSF has failed with the type of failure that occurred.
VPNGW12:FTA_VCM_EXT.1	None	None
NDcPP22e/WLANAS10:FTP_ITC.1	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions. Failed attempts to establish a trusted channel (including IEEE 802.11). Detection of modification of channel data	Identification of the initiator and target of failed trusted channels establishment attempt. Identification of the initiator and target of channel.
WLANAS10:FTP_ITC.1/Client	None	None
VPNGW12:FTP_ITC.1/VPN	Termination of the trusted channel Failure of the trusted channel functions	Identification of the initiator and target of failed trusted channel establishment attempt
NDcPP22e:FTP_TRP.1/Admin	Initiation of the trusted path. Termination of the trusted path. Failure of the trusted path functions.	None

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 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 Security Audit Data Generation (STFFW14e:FAU_GEN.1)

2.1.2.1 STFFW14e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: No additional Evaluation Activities are specified.

No additional Evaluation Activities are specified.

Component Guidance Assurance Activities: In addition to the Evaluation Activities specified in the Supporting Document for the Base-PP, the evaluator shall check the guidance documentation to ensure that it describes the audit records specified in Table 2 of the PP-Module in addition to those required by the Base-PP. If the optional SFR FFW_RUL_EXT.2 is claimed by the TOE, the evaluator shall also check the guidance documentation to ensure that it describes the relevant audit record specified in Table 3 of the PP-Module.

See NDcPP22e:FAU_GEN.1 which identifies all audit records documented in the Admin Guide including those from the PP-Module.

Component Testing Assurance Activities: In addition to the Evaluation Activities specified in the Supporting Document for the Base-PP, the evaluator shall perform tests to demonstrate that audit records are generated for the auditable events as specified in Table 2 of the PP-Module and, if the optional SFR FFW_RUL_EXT.2 is claimed by the TOE, Table 3.

The audit records associated with STFFW14e:FFW_RUL_EXT.2 were collected during testing of that requirement. All required audits including those specified in Table 2 of the PP-Module were found during that testing.

The evaluators further verified that the audits contained the necessary information.

2.1.3 Audit Data Generation (VPN Gateway) (VPNGW12:FAU_GEN.1/VPN)

2.1.3.1 VPNGW12:FAU_GEN.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 VPNGW12:FAU_GEN.1.2/VPN

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 audit mechanisms that the TOE uses to generate audit records for VPN gateway behavior. If any audit mechanisms the TSF uses for this are not used to generate audit records for events defined by FAU_GEN.1 in the Base-PP, the evaluator shall ensure that any VPN gateway-specific audit mechanisms also meet the relevant functional claims from the Base-PP.

For example, FAU_STG_EXT.1 requires all audit records to be transmitted to the OE over a trusted channel.

This includes the audit records that are required by FAU_GEN.1/VPN. Therefore, if the TOE has an audit mechanism that is only used for VPN gateway functionality, the evaluator shall ensure that the VPN gateway related audit records meet this requirement, even if the mechanism used to generate these audit records does not apply to any of the auditable events defined in the Base-PP.

Section 6.1 of the ST indicates that the TOE generates audit records for start-up and shutdown of the TOE, all administrator actions, and for all the events identified in Table 8 Auditable Events which includes the entire packet contents for packets transmitted/received during IPsec peer session establishment. Section 6.1 of the ST further explains that the TOE can be configured to log events by including the ‘log’ keyword when defining a firewall rule. In the event that a TOE network interface is overwhelmed by traffic the TOE will drop packets and generate an audit event for every packet that is denied and dropped.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it identifies all security-relevant auditable events claimed in the ST and includes sample records of each event type. If the TOE uses multiple audit mechanisms to generate different sets of records, the evaluator shall verify that the operational guidance identifies the audit records that are associated with each of the mechanisms such that the source of each audit record type is clear.

See NDcPP22e: FAU_GEN.1 where the evaluator confirmed that the Admin Guide identifies all auditable events claimed in the ST and includes sample records.

Component Testing Assurance Activities: The evaluator shall test the audit functionality by performing actions that trigger each of the claimed audit events and verifying that the audit records are accurate and that their format is consistent with what is specified in the operational guidance. The evaluator may generate these audit events as a consequence of performing other tests that would cause these events to be generated.

The audit records associated with VPNGW12:FAU_GEN.1/VPN were collected during testing of that requirement. All required audits including those specified in Table 2 and Table 3 of the PP-Module were found during that testing.

The evaluators further verified that the audits contained the necessary information.

2.1.4 Audit Data Generation (WLANAS10:FAU_GEN.1/WLAN)

2.1.4.1 WLANAS10:FAU_GEN.1.1/WLAN

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: The evaluator will test the TOE's ability to correctly generate audit records by having the TOE generate audit records in accordance with the evaluation activities associated with the functional requirements in this PPModule. When verifying the test results, the evaluator will ensure the audit records generated during testing match the format specified in the administrative guide and that the fields in each audit record have the proper entries.

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

The audit records associated with WLANAS10:FAU_GEN.1/WLAN were collected during testing of that requirement. All required audits including those specified in Table 2 of the PP-Module were found during that testing.

The evaluators further verified that the audits contained the necessary information.

2.1.5 User identity association (NDcPP22e:FAU_GEN.2)

2.1.5.1 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.

The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for 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.

The TSS and Guidance Documentation requirements for FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for 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.

This activity was accomplished in conjunction with the testing of FAU_GEN.1.1.

2.1.6 Protected audit trail storage (NDcPP22e:FAU_STG.1)

2.1.6.1 NDcPP22e:FAU_STG.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.6.2 NDcPP22e:FAU_STG.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 amount of audit data that are stored locally and how these records are protected against unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.

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

Section 6.1 of the ST explains that the TOE protects audit records in local storage from unauthorized modification or deletion. The local logs can only be viewed - they cannot be deleted or modified. There are no CLI or GUI commands for such actions.

The TOE stores audit records locally and provides CLI and Web UI capabilities for the administrator to view the contents of the audit trail. For local storage, the maximum log file size for all processes is ARUBA_MAX_LOG_FILE_SIZE (i.e. 95,304 bytes). However, for 72xx and x86 platforms, the security.log, system.log and user-debug logs have a maximum file size given by A_MAX_SECURITY_USER_DEBUG_LOG_FILE_SIZE (i.e. 349,524 bytes). The local protected log storage operates using the first in, first out (FIFO) method, therefore audit logs are overwritten when the available space is exhausted

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it describes any configuration required for protection of the locally stored audit data against unauthorized modification or deletion.

Section “FAU_STG.1” in the Admin Guide indicates that no configuration is required. There are no CLI or GUI commands for such actions.

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

a) Test 1: The evaluator shall access the audit trail without authentication as Security Administrator (either by authentication as a non-administrative user, if supported, or without authentication at all) and attempt to modify and delete the audit records. The evaluator shall verify that these attempts 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 access the audit trail 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.

b) Test 2: The evaluator shall access the audit trail as an authorized administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted.

For distributed TOEs the evaluator shall perform test 1 and test 2 for each component that is defined by the TSS to be covered by this SFR.

Test 1 - ArubaOS does not offer any command or edit functions which allow an administrator to modify or delete audit data. The evaluator logged into an account that did not have permission to delete audit events and verified that all attempts were denied.

Test 2 - The TOE does allow deletion of files, and the evaluator attempted to delete the log file and verified that all attempts were denied.

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

2.1.7.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.3 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.

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 6.1 of the ST explains that the TOE protects audit records in local storage from unauthorized modification or deletion. The local logs can only be viewed - they cannot be deleted or modified. There are no CLI or GUI commands for such actions. The TOE stores audit records locally and provides CLI and Web UI capabilities for the administrator to view the contents of the audit trail. For local storage, the maximum log file size for all processes is ARUBA_MAX_LOG_FILE_SIZE (i.e. 95,304 bytes). However, for 72xx and x86 platforms, the security.log, system.log and user-debug logs have a maximum file size given by A_MAX_SECURITY_USER_DEBUG_LOG_FILE_SIZE (i.e. 349,524 bytes). The local protected log storage operates using the first in, first out (FIFO) method, therefore audit logs are overwritten when the available space is exhausted.

The TOE can also be configured to send audit records to a trusted third-party syslog server in the operational environment. The TOE uses IPsec to protect the communication channel between itself and the remote syslog server. If an external syslog server has been enabled, all audit logs are simultaneously written to both the local audit log and the syslog server. The local audit logs and logs sent to a remote server are identical.

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 “FAU_STG_EXT.1” in the Admin Guide provides the commands for configuring the syslog server and the protected IPsec connection between the TOE and the syslog server. It further states that local storage space for audit logs is limited on a mobility controller. The local protected log storage operates using the first in, first out (FIFO) method, therefore audit logs are overwritten when the available space is exhausted. Audit logs are simultaneously written to both the local audit log and the syslog server. The local audit logs and logs sent to a remote server are identical

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.

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 external audit server was utilizing a Rsyslogd 8.16.0. The evaluator observed the TOE initiating an IPsec tunnel to the syslog server and demonstrated that the syslog messages were successfully sent to the syslog server through the encrypted channel. No plaintext traffic was observed in the packet capture. The evaluator also verified that the logs were successfully received by viewing them on the syslog server.

Test 2: The evaluator verified that when the local audit storage was filled, the existing audit data was overwritten using the following rule: Overwrite oldest records first.

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

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

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.

Section 6.2 of the ST indicates that the TOE supports cryptographic key generation for RSA schemes using key sizes of 2048 bits or greater, ECC schemes using NIST curves P-256 and P-384 FFC schemes using key sizes of 2048 bits or greater and FFC Schemes using Diffie-Hellman group 14 that meet the following: RFC 3526, Section 3. The TOE is a TLS and SSH server, and an IPsec client and server.

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 “FCS_CKM.1” of the Admin Guide indicates that the TOE must have FIPS mode enabled so that the cryptographic requirements are met. Beyond this, no further configuration is required.

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 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 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)

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

2.2.2 Cryptographic Key Generation (for IKE Peer Authentication) - per TD0723 (VPNGW12:FCS_CKM.1/IKE)

2.2.2.1 VPNGW12:FCS_CKM.1.1/IKE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check to ensure that the TSS describes how the key-pairs are generated. In order to show that the TSF implementation complies with FIPS PUB 186-4, the evaluator shall ensure that the TSS contains the following information:

- The TSS shall list all sections of Appendix B to which the TOE complies.

- For each applicable section listed in the TSS, for all statements that are not 'shall' (that is, 'shall not', 'should', and 'should not'), if the TOE implements such options it shall be described in the TSS. If the included functionality is indicated as 'shall not' or 'should not' in the standard, the TSS shall provide a rationale for why this will not adversely affect the security policy implemented by the TOE;

- For each applicable section of Appendix B, any omission of functionality related to 'shall' or 'should' statements shall be described;

Any TOE-specific extensions, processing that is not included in the Appendices, or alternative implementations allowed by the Appendices that may impact the security requirements the TOE is to enforce shall be described.

Section 6.2 of the ST states that the TOE fulfills all of the FIPS PUB 186-4 requirements for cryptographic key generation without extensions. The TOE conforms to all shall, shall-not, should and should-not statements. For RSA key establishment, the TOE implements section B.3.6 in Appendix B.3 of FIPS PUB 186-4. For ECDSA, the TOE implements section B.4.2 in Appendix B of FIPS PUB 186-4. The TOE implementation of Diffie-Hellman group 14 (2048 MODP) meets RFC 3526, Section 3.

Component Guidance Assurance Activities: The evaluator shall check that the operational guidance describes how the key generation functionality is invoked, and describes the inputs and outputs associated with the process for each signature scheme supported. The evaluator shall also check that guidance is provided regarding the format and location of the output of the key generation process.

Section “FCS_CKM.1” of the Admin Guide indicates that the TOE must have FIPS mode enabled so that the cryptographic requirements are met. Beyond this, no further configuration is required. During regular operation of the TOE, key generation is invoked during session establishment between the TOE and external IT entities for user sessions. An administrator can invoke the use of RSA and ECDSA during generation of certificates used for X.509.

Component Testing Assurance Activities: For FFC Schemes using 'safe-prime' groups:

Testing for FFC Schemes using safe-prime groups is done as part of testing in FCS_CKM.2.

For all other selections:

The evaluator shall perform the corresponding tests for FCS_CKM.1 specified in the NDcPP SD, based on the selections chosen for this SFR. If IKE key generation is implemented by a different algorithm than the NDcPP key generation function, the evaluator shall ensure this testing is performed using the correct implementation.

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

2.2.3 Cryptographic Key Generation (Symmetric Keys for WPA2 Connections) (WLANAS10:FCS_CKM.1/WPA)

2.2.3.1 WLANAS10:FCS_CKM.1.1/WPA

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The cryptographic primitives will be verified through evaluation activities specified elsewhere in this PPModule. The evaluator will verify that the TSS describes how the primitives defined and implemented by this PP-Module are used by the TOE in establishing and maintaining secure connectivity to the wireless clients. This description will include how the GTK and PTK are generated or derived. The TSS will also provide a description of the developer's methods of assuring that their implementation conforms to the cryptographic standards; this includes not only testing done by the developing organization, but also proof of third-party testing that is performed (e.g. WPA2 certification). The evaluator will ensure that the description of the testing methodology is of sufficient detail to determine the extent to which the details of the protocol specifics are tested.

Section 6.2 of the ST states that for communications between the TOE and a wireless client, the TOE generates a symmetric key of size 128 bits in accordance with the cryptographic key generation algorithm, PRF-384, using the Random Bit Generator as specified in FCS_RBG_EXT.1 that meets the IEEE 802.11-2012 standard. The TOE supports cryptographic key distribution for 802.11 PMK key reception from an 802.1X authentication server, AES Key Wrap in EAPOL-Key frame and Group Key Handshake. Aruba uses an automated test tool (Ixia IxANVL) to test conformance with RFCs and 802.1X. Aruba also uses VeriWave test tools to test Wi-Fi performance. One feature of the VeriWave test suite is to exercise 802.1X capabilities of the product. Aruba also conducts interoperability testing through custom-built automated test beds which contain numerous client operating systems (Windows XP, Windows Vista, Windows 7, Windows 8, Mac OS X, Linux, Apple iOS, Android, etc.) connecting to Aruba Wi-Fi access points. Finally, the products are Wi-Fi certified by the Wi-Fi Alliance - conformance with 802.1X and EAP/RADIUS are requirements to pass these tests.

See Section 1.1.2 above for the certificate numbers. The WiFi Alliance maintains a website providing further information about the testing program: http://www.wi-fi.org/certification.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator will perform the following test using a packet sniffing tool to collect frames between the TOE and a wireless client:

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

Step 2: The evaluator will configure the TOE to communicate with a WLAN client using IEEE 802.11-2020 and a 256-bit (64 hex values 0-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.Step 3: The evaluator will start the sniffing tool, initiate a connection between the TOE and WLAN client, and allow the TOE to authenticate, associate, and successfully complete the four-way handshake with the client.

Step 4: The evaluator will set a timer for one minute, at the end of which the evaluator will disconnect the client from the TOE and stop the sniffer.

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

Step 6: The evaluator will select the first data frame from the captured packets that was sent between the client and TOE after the four-way handshake successfully completed and without the frame control value 0x4208 (the first two bytes are 08 42). The evaluator will use the PTK to decrypt the data portion of the packet as specified in IEEE 802.11-2020 and verify that the decrypted data contains ASCII-readable text.

Step 7: The evaluator will repeat Step 6 for the next two data frames between the TOE and client, and without frame control value 0x4208.

Additionally, the evaluator will test the PRF function using the test vectors from:

- Section 2.4 'The PRF Function - PRF (key, prefix, data, length)' of the IEEE 802.11-02/362r6 document 'Proposed Test vectors for IEEE 802.11 TGi' dated September 10, 2002

- Annex J.3 'PRF reference implementation and test vectors' of IEEE 802.11-2020

Step 1: The evaluator configured the access point and configured the sniffer to sniff the 802.11 traffic received by the AP.

Step 2: The evaluator configured the TOE for a 256-bit pre-shared key and setup the connection as described in the operational guidance.

Step 3: The evaluator started Wireshark to capture the 802.11 traffic.

Step 4: The TOE was connected and disconnected after more than one minute while a number of broadcast packets were observed using Wireshark and the capture was collected.

Step 5: The evaluator identified the 4-way handshake frames as well as data frames using Wireshark and collected the packet capture.

Step 6 and 7: After enabling decryption, the evaluator was able to see the frames, albeit properly decrypted as expected.

See Section 1.1.2 above for the CAVP and Wi-Fi alliance certificates.

2.2.4 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.2.4.1 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 6.2 of the ST indicates that the TOE supports cryptographic key generation for RSA schemes using key sizes of 2048 bits or greater, ECC schemes using NIST curves P-256 and P-384 FFC schemes using key sizes of 2048 bits or greater and FFC Schemes using Diffie-Hellman group 14 that meet the following: RFC 3526, Section 3. The TOE is a TLS and SSH server, and an IPsec client and server. The table below identifies the key exchange methods/schemes used by TOE services:

Security Function	Communication Type	Key Establishment Methods
Administration	TLS	RSA Schemes ECC Schemes FFC Schemes
Administration	SSH	DH-14
Trusted Channels for Syslog, NTP, Authentication Services	IPsec	RSA Schemes ECC Schemes DH-14

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).

Section "FCS_CKM.1" of the Admin Guide indicates that the TOE must have FIPS mode enabled so that the cryptographic requirements are met. Beyond this, no further configuration is required.

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.

2.2.5 Cryptographic Key Distribution (GTK) (WLANAS10:FCS_CKM.2/GTK)

2.2.5.1 WLANAS10:FCS_CKM.2.1/GTK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator will check the TSS to ensure that it describes how the GTK is wrapped prior to being distributed using the AES implementation specified in this PP-Module, and also how the GTKs are distributed when multiple clients connect to the TOE.

Section 6.2 of the ST states that wireless encryption between the TOE and the wireless client is implemented using WPA3/WPA2. WPA3/WPA2 uses AES in CCMP mode for authentication and data encryption. From the 802.1X authentication exchange, the client and the controller derive dynamic keys (PMK and GTK) to encrypt data transmitted on the wireless network. The PMK and the GTK are both derived during the EAP-TLS/PEAP handshake. When the RADIUS protocol is used between the TOE and the authentication server, the MS-MPPE-Recv-Key attribute (vendor-id = 17; see Section 2.4.3 in IETF RFC 2548-1999 [B30]) is used to transport the PMK to TOE. The GTK is distributed by using AES Key Wrap in an EAPOL-Key frame in accordance with 802.11-2012.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator will perform the following test using a packet sniffing tool to collect frames between a wireless client and the TOE (which may be performed in conjunction with the evaluation activity for FCS_CKM.1/PMK.

To fully test the broadcast and multicast functionality, these steps will be performed as the evaluator connects multiple clients to the TOE. The evaluator will ensure that GTKs established are sent to the appropriate participating clients.

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

Step 2: The evaluator will configure the TOE to communicate with the client using IEEE 802.11-2020 and a 256-bit (64 hex values 0-f) pre-shared key, setting up the connections as described in the operational guidance. The pre-shared key is only used for testing.

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

Step 4: The evaluator will set a timer for one minute, at the end of which the evaluator will disconnect the TOE from the client and stop the sniffer.

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

Step 6: The evaluator will select the first data frame from the captured packets that was sent between the TOE and client after the four-way handshake successfully completed, and with the frame control value 0x4208 (the first two bytes are 08 42). The evaluator will use the GTK to decrypt the data portion of the selected packet as specified in IEEE 802.11-2020 and verify that the decrypted data contains ASCII-readable text.

Step 7: The evaluator will repeat Step 6 for the next two data frames with frame control value 0x4208.

The evaluator will also perform the following testing based on the supported GTK distribution methods:

AES Key Wrap (AES-KW Tests)

Test 1: The evaluator will test the authenticated encryption functionality of AES-KW for each combination of the following input parameter lengths:

128 and 256 bit key encryption keys (KEKs)

Three plaintext lengths:

1. One of the plaintext lengths will be two semi-blocks (128 bits).

2. One of the plaintext lengths will be three semi-blocks (192 bits).

3. The third data unit length will be the longest supported plaintext length less than or equal to 64 semi-blocks (4096 bits).

For each combination, generate a set of 100 key and plaintext pairs and obtain the ciphertext that results from AES-KW authenticated encryption. To determine correctness, the evaluator will use the same key and plaintext values and encrypt them using a known good implementation of AES-KW authenticatedencryption, and ensure that the resulting respective ciphertext values are identical.

Test 2: The evaluator will test the authenticated-decryption functionality of AES-KW using the same test as for authenticated-encryption, replacing plaintext values with ciphertext values and AES-KW authenticated-encryption with AES-KW authenticated-decryption. Additionally, the evaluator will modify one byte of the ciphertext, attempt to decrypt the modified ciphertext, and ensure that a failure is returned rather than plaintext.

AES Key Wrap with Padding (AES-KWP Tests)

Test 1: The evaluator will test the authenticated-encryption functionality of AES-KWP for each combination of the following input parameter lengths:

128 and 256 bit key encryption keys (KEKs)

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

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

Test 2: The evaluator will test the authenticated-decryption functionality of AES-KWP using the same test as for AES-KWP authenticated-encryption, replacing plaintext values with ciphertext values and AES-KWP authenticated-encryption with AES-KWP authenticated-decryption. Additionally, the evaluator will modify one byte of the ciphertext, attempt to decrypt the modified ciphertext, and ensure that a failure is returned rather than plaintext.

Step 1-7: See WLANAS10:FCS_CKM.1.1/WPA where a description of testing activities is provided.

AES Key Wrap tests: the TOE has CAVP certificates as identified in section 1.1.2.

2.2.6 Cryptographic Key Distribution (PMK) (WLANAS10:FCS_CKM.2/PMK)

2.2.6.1 WLANAS10:FCS_CKM.2.1/PMK

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator will examine the TSS to determine that it describes how the PMK is transferred (that is, through what EAP attribute) to the TOE.

Section 6.2 of the ST states that wireless encryption between the TOE and the wireless client is implemented using WPA2. WPA2 uses AES in CCMP mode for authentication and data encryption. From the 802.1X authentication exchange, the client and the controller derive dynamic keys (PMK and GTK) to encrypt data transmitted on the wireless network. The PMK and the GTK are both derived during the EAP-TLS/PEAP handshake. When the RADIUS protocol is used between the TOE and the authentication server, the MS-MPPE-Recv-Key attribute (vendor-id = 17; see Section 2.4.3 in IETF RFC 2548-1999 [B30]) is used to transport the PMK to TOE. The GTK is distributed by using AES Key Wrap in an EAPOL-Key frame in accordance with 802.11-2012.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: The evaluator will establish a session between the TOE and a RADIUS server according to the configuration guidance provided. The evaluator will then examine the traffic that passes between the RADIUS server and the TOE during a successful attempt to connect a wireless client to the TOE to determine that the PMK is not exposed.

The evaluator set up a RADIUS server on a Linux machine, which is also capable of capturing packets between the TOE and the RADIUS server. The evaluator also used a mobile device that is capable of reporting the PMK value in the WPA handshake process. The packet capturer was enabled and the mobile device connected so that the evaluator collected both the RADIUS packets for the connection and the PMK. The evaluator then opened up the packet capture in HxD (a hex editor) and searched for the PMK value. The evaluator confirmed that the PMK is not exposed.

2.2.7 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.7.1 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.

Section 6.2 of the ST explains that the TOE is designed to zeroize secret and private keys when they are no longer required by the TOE. Zeroization is accomplished by overwriting the secret or private key with all zeroes. The ST contains a table that identifies all secret and private keys and Critical Security Parameters (CSPs), the related zeroization procedures and whether any interface is available to view the plaintext key.

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 “FCS_CKM.4” of the Admin Guide states that no configuration is required. During runtime, all CSPs will be zeroized automatically when no longer needed. To erase all CSPs stored in flash memory (as well as software images and configuration files), the command ‘zeroize-tpm-keys’ (for hardware) and ‘wipe out flash’ (for virtual) should be issued. These commands will overwrite the entire flash with an alternating pattern. The controller must be restored through TFTP after this process. In addition, files in the flash can be zeroized using the ‘write erase all’ command. There are no configurations or circumstances that do not strictly conform to the key destruction requirement, nor does the Admin Guide identify any situations where key destruction is delayed at the physical layer.

Component Testing Assurance Activities: None Defined

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

2.2.8.1 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 6.2 of the ST contains Table 10 Cryptographic Functions which identifies the following key size(s) and mode(s) for data encryption/decryption:

AES CBC (128, 192 an d 256 bits)

AES GCM (128 and 256 bits)

AES CTR (128 and 256 bits)

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 "FCS_COP.1" in the Admin Guide explains that the Advanced Cryptography License must installed and FIPS mode must be enabled to ensure that the evaluated cryptographic algorithms will be used and the cryptographic requirements are met.

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].

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:

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.

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.

2.2.9 Cryptographic Operation (AES Data Encryption/Decryption) (VPNGW12:FCS_COP.1/DataEncryption)

2.2.9.1 VPNGW12:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to require the ST author to make certain selections, but these selections are all part of the original definition of the SFR so no new behavior is defined by the PP-Module.

See NDcPP22e: FCS_COP.1/DataEncryption

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.10 Cryptographic Operation (AES Data Encryption/Decryption) (WLANAS10:FCS_COP.1/DataEncryption)

2.2.10.1 WLANAS10:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There are no additional TSS evaluation activities for this component beyond what the NDcPP requires.

See NDcPP22e: FCS_COP.1/DataEncryption

Component Guidance Assurance Activities: There are no additional guidance evaluation activities for this component beyond what the NDcPP requires.

There are no additional guidance evaluation activities for this component beyond what the NDcPP requires.

Component Testing Assurance Activities: In addition to the tests required by the NDcPP, the evaluator will perform the following testing:

AES-CCM Tests

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

- 128 bit and 256 bit keys

- Two payload lengths. One payload length will be the shortest supported payload length, greater than or equal to zero bytes. The other payload length will be the longest supported payload length, less than or equal to 32 bytes (256 bits).

- Two or three associated data lengths. One associated data length will be 0, if supported. One associated data length will be the shortest supported payload length, greater than or equal to zero bytes. One associated data length will be the longest supported payload length, less than or equal to 32 bytes (256 bits). If the implementation supports an associated data length of 216 bytes, an associated data length of 216 bytes will be tested.

- Nonce lengths. All supported nonce lengths between 7 and 13 bytes, inclusive, will be tested.

- Tag lengths. All supported tag lengths of 4, 6, 8, 10, 12, 14, and 16 bytes will be tested.

Due to the restrictions that IEEE 802.11 specifies for this mode (nonce length of 13 and tag length of 8), it is acceptable to test a subset of the supported lengths as long as the selections fall into the ranges specified above. In this case, the evaluator will ensure that these are the only supported lengths. To test the generation-encryption functionality of AES-CCM, the evaluator will perform the following four tests:

Test 1: For each supported key and associated data length and any supported payload, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Test 2: For each supported key and payload length and any supported associated data, nonce and tag length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values, and obtain the resulting ciphertext.

Test 3: For each supported key and nonce length and any supported associated data, payload, and tag length, the evaluator will supply one key value and 10 associated data, payload and nonce value 3-tuples, and obtain the resulting ciphertext.

Test 4: For each supported key and tag length and any supported associated data, payload, and nonce length, the evaluator will supply one key value, one nonce value and 10 pairs of associated data and payload values and, obtain the resulting ciphertext.

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

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

Additionally, the evaluator will use tests from the IEEE 802.11-02/362r6 document 'Proposed Test vectors for IEEE 802.11 TGi', dated September 10, 2002, Section 2.1 AES-CCMP Encapsulation Example and Section 2.2 Additional AES-CCMP Test Vectors to verify further the IEEE 802.11-2020 implementation of AES-CCMP.

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

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

2.2.11.1 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 6.2 of the ST states that the TOE supports SHA-1/256/384/512 (digest sizes 160, 256, 384, and 512 bits) for cryptographic hashing. The TOE implements SHA-1, SHA-256 and SHA-384 in support of TLS v1.2 in conjunction with AES (CBC and GCM) 128- or 256-bit ciphers and RSA and ECDSA and SHA-256 and SHA-512 in support of SSH. The TOE stores passwords in flash using a SHA-1 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 “FCS_COP.1” in the Admin Guide explains that the Advanced Cryptography License must installed and FIPS mode must be enabled to ensure that the evaluated cryptographic algorithms will be used and the cryptographic requirements are met.

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

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.

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

2.2.12.1 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 6.2 of the ST states that the TOE supports HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC_SHA-512 (key and output MAC sizes 160, 256, 384, and 512, respectively) for keyed-hash message authentication. The TOE implements HMAC-SHA-1, HMAC-SHA-1-96 and diffie-hellman-group 14-sha1 in support of SSHv2 and HMAC-SHA-1 in support of IPsec..

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

Section "FCS_COP.1" in the Admin Guide explains that the Advanced Cryptography License must installed and FIPS mode must be enabled to ensure that the evaluated cryptographic algorithms will be used and the cryptographic requirements are met.

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.

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

2.2.13.1 NDcPP22e:FCS_COP.1.1/SigGen

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 specifies the cryptographic algorithm and key size supported by the TOE for signature services.

Section 6.2 of the ST states that the TOE supports rDSA (modulus 2048) and ECDSA with elliptical curve sizes of 256 or 384 bits for signature generation and verification.

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 "FCS_COP.1" in the Admin Guide explains that the Advanced Cryptography License must installed and FIPS mode must be enabled to ensure that the evaluated cryptographic algorithms will be used and the cryptographic requirements are met.

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

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.

2.2.14 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.2.14.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.3 NDcPP22e:FCS_HTTPS_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 and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Section 6.2 of the ST states that the TOE supports TLS/HTTPS web-based secure administrator sessions in compliance with RFC 2818. The TOE supports TLSv1.2. Any other SSL/TLS versions are not supported by the TOE and such connection attempts will be rejected. If an invalid peer certificate is presented, the TOE will not establish the connection.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to configure TOE for use as an HTTPS client or HTTPS server.

Section "FCS_HTTPS_EXT.1" in the Admin Guide explains that when in FIPS mode, the TOE does not need any configuration to operate using HTTPS with compliance to RFC 2818.

Component Testing Assurance Activities: This test is now performed as part of FIA_X509_EXT.1/Rev testing.

Tests are performed in conjunction with the TLS evaluation activities.

If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

The HTTPS server implementation was tested as part of NDcPP22E:FCS_TLSS_EXT.1. The TOE does not use X.509 certificates for authentication of remote administrators.

2.2.15 IPsec Protocol - per TD0633 (NDcPP22e:FCS_IPSEC_EXT.1)

2.2.15.1 NDcPP22e:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet. The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule. The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in

RFC 4301.

As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA.

Section 6.2 of the ST explains that the TOE implements an SPD to determine what traffic gets protected with IPsec, what gets bypassed, and what gets dropped. The SPD is achieved via the routing table and firewall policies. For client-to-gateway VPN links, the firewall policies are in control of how traffic is forwarded. For site-to-site VPN, the routing table is in control. In each case, additional routing or firewall rules may be applied. The TOE administrator implicitly configures the IPsec SPD via the routing table and firewall policies and includes a final rule that causes the network packet to be discarded if no other rules are matched.

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases â€“ a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

Section “FCS_IPSEC_EXT.1” of the Admin Guide explains that RFC 4301 references an explicit Security Policy Database (SPD) with rules for DISCARD, BYPASS, and PROTECT. ArubaOS does not implement an explicit SPD, but equivalent behavior may be obtained through the use of firewall policies and “routing” ACLs. The Admin Guide then provides examples for how to create the ACLs. The access control lists used by the TOE are read in hierarchical order. When traffic enters or exits the TOE, the first applicable rule in the ACL is applied. Any additional rules that include applicable traffic below the initially triggered rule are not applied. Note that if an access rule is applied, a duplicate cannot be entered. If the administrator applied a permit rule and then enters a deny rule with the same parameters, the deny rule will replace the permit rule and vice versa

Testing Assurance Activities: The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

a) Test 1: The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The selectors used in the construction of the rule shall be different such that the evaluator can generate a packet and send packets to the gateway with the appropriate fields (fields that are used by the rule - e.g., the IP addresses, TCP/UDP ports) in the packet header. The evaluator performs both positive and negative test cases for each type of rule (e.g. a packet that matches the rule and another that does not match the rule). The evaluator observes via the audit trail, and packet captures that the TOE exhibited the expected behaviour: appropriate packets were dropped, allowed to flow without modification, encrypted by the IPsec implementation.

b) Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and guidance documentation. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation.

Test 1 - The evaluator created an access-list with a bypass rule to allow plaintext traffic, a discard rule to discard SSH traffic, protect (encrypt) rules to allow encrypted traffic to flow through the IPsec tunnel and a default deny rule. The evaluator sent network traffic to the TOE to cause the TOE to protect, bypass, and discard. The evaluator verified that the TOE processed the packets accordingly.

Test 2 - This test was performed as part of test 1.

2.2.15.2 NDcPP22e:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1.

The evaluator uses the guidance documentation to configure the TOE to carry out the following tests:

The evaluator shall configure the SPD such that there is a rule for dropping a packet, encrypting a packet, and allowing a packet to flow in plaintext. The evaluator may use the SPD that was created for verification of FCS_IPSEC_EXT.1.1. The evaluator shall construct a network packet that matches the rule to allow the packet to flow in plaintext and send that packet. The evaluator should observe that the network packet is passed to the proper destination interface with no modification. The evaluator shall then modify a field in the packet header; such that it no longer matches the evaluator-created entries (there may be a 'TOE create'' final entry that discards packets that do not match any previous entries). The evaluator sends the packet and observes that the packet was dropped.

This test was performed as a part of NDcPP21: FCS_IPSEC_EXT.1.1, test 1.

2.2.15.3 NDcPP22e:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: The evaluator checks the TSS to ensure it states that the VPN can be established to operate in transport mode and/or tunnel mode (as identified in FCS_IPSEC_EXT.1.3).

Section 6.2 of the ST states that the TOE supports IPsec in both transport and tunnel mode. This is consistent with FCS_IPSEC_EXT.1.3.

Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions on how to configure the connection in each mode selected.

Section "FCS_IPSEC_EXT.1.3" in the Admin Guide states that IPsec tunnel and transport mode is supported In ArubaOS. It then provides an example of how to establish a site-to-site VPN tunnel.

Testing Assurance Activities: The evaluator shall perform the following test(s) based on the selections chosen:

Test 1: If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode.

Test 2: If transport mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in transport mode and also configures a VPN peer to operate in transport mode. The evaluator configures the TOE and the VPN peer to use any of the allowed cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator then initiates a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the transport mode.

Test 1: The TOE supports tunnel mode. The evaluator configured a VPN peer to require only tunnel mode. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and observed via logs and a packet capture that the connection was successful.

Test 2: The TOE supports transport mode. The evaluator configured a VPN peer to require only transport mode. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and observed via logs and a packet capture that the connection was successful.

2.2.15.4 NDcPP22e:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: The evaluator shall examine the TSS to verify that the algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1(4)/KeyedHash Cryptographic Operations (for keyed-hash message authentication) and if the SHA-based HMAC function truncated output is utilized it must also be described.

Section 6.2 in the ST states that the IPsec ESP protocol is implemented in conjunction with AES-CBC-128, AES-CBC192, AES-CBC-256 (as specified by RFC 3602) and AES-GCM-128 and AES-GCM-256 (as specified by RFC 4106) together with the HMAC-SHA-1 algorithm. HMAC-SHA-1 is specified in FCS_COP.1/KeyedHash.

Guidance Assurance Activities: The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected.

Section "FCS_IPSEC_EXT.1.4" of the Admin Guide explains how to configure the ciphers. IPsec cipher suites are configured using transform-sets. These are ordered lists of ciphers - the controller will attempt each one in order until one is successfully negotiated with the peer. The Admin Guide then provides examples for how to set each claimed cipher.

Testing Assurance Activities: The evaluator shall configure the TOE as indicated in the guidance documentation configuring the TOE to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds.

Test - The evaluator configured the TOE for AES-CBC-128, AES-CBC-192, AES-CBC-256, AES-GCM-128 and AESGCM-256 and verified via logs and packet captures that the connection was successfully established for each algorithm. This was repeated for IKEv1 and IKEv2.

2.2.15.5 NDcPP22e:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented.

For IKEv1 implementations, the evaluator shall examine the TSS to ensure that, in the description of the IPsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option.

Section 6.2 in the ST states that the TOE implements both IKEv1, as defined in RFCs 2407, 2408, 2409, and RFC 4109; and IKE2, with support for NAT traversal, as defined in RFC 5996 and RFC 4868 for hash functions (HMAC-SHA-1). For IKEv1, aggressive mode is not used, only main mode is supported.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and how to configure the TOE to perform NAT traversal for the following test (if selected).

If the IKEv1 Phase 1 mode requires configuration of the TOE prior to its operation, the evaluator shall check the guidance documentation to ensure that instructions for this configuration are contained within that guidance.

Section "FCS_IPSEC_EXT.1.5" of the Admin Guide states that IKEv1 and IKEv2 are both supported, and both may be configured simultaneously. NAT Traversal (NAT-T) is supported for both. NAT-T transports packets over UDP port 4500 rather than using IPsec native encapsulation. For inbound connections where the controller is the IKE responder, NAT-T is supported by default. To disable, a firewall rule that blocks UDP 4500 can be configured. The command for selecting IKEv1 or IKEv2 is provided.

Testing Assurance Activities: Tests are performed in conjunction with the other IPsec evaluation activities.

a) Test 1: If IKEv1 is selected, the evaluator shall configure the TOE as indicated in the guidance documentation and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported.

b) Test 2: If NAT traversal is selected within the IKEv2 selection, the evaluator shall configure the TOE so that it will perform NAT traversal processing as described in the TSS and RFC 5996, section 2.23. The evaluator shall initiate an IPsec connection and determine that the NAT is successfully traversed.

Test 1: The evaluator configured the TOE for IKEv1. The evaluator attempted to connect in aggressive mode but the TOE rejected the connection. The TOE accepted the main mode connection.

Test 2: The evaluator configured the connection between the TOE and test peer so that NAT was required - both devices were on separate class C networks with a NAT router bridging the two networks. The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to be successful regardless of the NAT routing. The evaluator initiated an IPsec connection and observed that the TOE correctly negotiated the NAT connection to establish a protected IPsec connection.

2.2.15.6 NDcPP22e:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms chosen in the selection of the requirement are included in the TSS discussion.

Section 6.2 in the ST states that the TOE uses the AES-CBC-128, AES-CBC-192 and AES-CBC-256 algorithms as specified in RFC 3602 to encrypt the IKEv1 or IKEv2 payload. This matches what is claimed in the requirement.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement.

Section "FCS_IPSEC_EXT.1.6" of the Admin Guide describes how to set the IKE algorithms for IKEv1 and IKEv2 (for AES-CBC-128, AES-CBC-192 and AES-CBC-256).

Testing Assurance Activities: The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation.

Test - The evaluator configured IKEv1 and IKEv2 (for AES-CBC-128, AES-CBC-192, AES-CBC-256) on the TOE. The evaluator then confirmed that the TOE could establish a session with each algorithm and that the tunnel successfully established with the selected algorithm

2.2.15.7 NDcPP22e:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 1 SA lifetime and/or the IKEv2 SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Section 6.2 of the ST states that the TOE supports SA lifetime limits based on length of time for both IKEv1 and IKEv2. Lifetimes for IKEv1 SAs (both Phase 1 and Phase 2) and IKEv2 SAs are established during administrator configuration of the IKE policies. In the case of IKEv1, lifetimes can be configured based on length of time within 1-24 hours for phase 1 and within 1-8 hours for phase 2 by specifying the number of seconds for the SA lifetime. In the case of IKEv2, lifetimes for the IKEv2 IKE_SA can be configured by specifying the number of seconds within 1-24 hours. For the IKEv2 IKE_CHILD SA, lifetimes can be configured by specifying the number of seconds within 1-8hrs for the SA lifetime. This matches the selections in the requirements.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g. configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

Section “FCS_IPSEC_EXT.1.7/8” of the Admin Guide describes how to set time-based lifetime values for both phase 1 and phase 2 in the IKE policies.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If 'number of bytes' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish a SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 1 negotiation.

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the Phase 1 SA lifetime on the TOE.

Test 1: This test is not applicable as 'number of bytes' is not selected for IKE/Phase 1 exchanges.

Test 2: The evaluator configured the TOE to have a 24 hour IKE and 8 hour ESP limits and the VPN test peer was configured to have 25 hour IKE and 9 hour ESP limits. The evaluator then connected the IPsec VPN between the test peer and the TOE. The evaluator observed through logs and packet captures that the connection was successful and that the TOE rekeyed each time the configured time limit was reached.

2.2.15.8 NDcPP22e:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 2 SA lifetime and/or the IKEv2 Child SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.5.

Section 6.2 of the ST states that the TOE supports SA lifetime limits based on length of time for both IKEv1 and IKEv2. Lifetimes for IKEv1 SAs (both Phase 1 and Phase 2) and IKEv2 SAs are established during administrator configuration of the IKE policies. In the case of IKEv1, lifetimes can be configured based on length of time within 1-24 hours for phase 1 and within 1-8 hours for phase 2 by specifying the number of seconds for the SA lifetime. In the case of IKEv2, lifetimes for the IKEv2 IKE_SA can be configured by specifying the number of seconds within 1-24 hours. For the IKEv2 IKE_CHILD SA, lifetimes can be configured by specifying the number of seconds within 1-8hrs for the SA lifetime. This matches the selections in the requirements.

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g. configure a time value of 7h 45min to ensure the actual rekey is performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement.

Section “FCS_IPSEC_EXT.1.7/8” of the Admin Guide describes how to set time-based lifetime values for both phase 1 and phase 2 in the IKE policies.

Testing Assurance Activities: When testing this functionality, the evaluator needs to ensure that both sides are configured appropriately. From the RFC 'A difference between IKEv1 and IKEv2 is that in IKEv1 SA lifetimes were negotiated. In IKEv2, each end of the SA is responsible for enforcing its own lifetime policy on the SA and rekeying the SA when necessary. If the two ends have different lifetime policies, the end with the shorter lifetime will end up always being the one to request the rekeying. If the two ends have the same lifetime policies, it is possible that both will initiate a rekeying at the same time (which will result in redundant SAs). To reduce the probability of this happening, the timing of rekeying requests SHOULD be jittered.'

Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

Test 1: If 'number of bytes' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime in terms of the number of bytes allowed following the guidance documentation. The evaluator shall configure a test peer with a byte lifetime that exceeds the lifetime of the TOE. The evaluator shall establish a SA between the TOE and the test peer, and determine that once the allowed number of bytes through this SA is exceeded, a new SA is negotiated. The evaluator shall verify that the TOE initiates a Phase 2 negotiation.

Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE.

Test 1 - The TOE does not claim support for "number of bytes".

Test 2 - This test was performed as part of FCS_IPSEC_EXT.1.7 test 2.

2.2.15.9 NDcPP22e:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating 'x'. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of 'x' meets the stipulations in the requirement.

Section 6.2 in the ST explains that the TOE generates the secret value x used in the IKEv1/IKEv2 Diffie-Hellman key exchange ('x' in gx mod p) using the FIPS validated RBG specified in FCS_RBG_EXT.1 and having possible lengths of 224, 256 or 384 bits (for DH Groups 14, 19, and 20, respectively).

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.10 NDcPP22e:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

Section 6.2 in the ST states that the TOE supports the following PRF hash functions: PRF_HMAC_SHA1, PRF_HMAC_SHA256 and PRF_HMAC_SHA384 and generates nonces used in the IKEv1 and IKEv2 exchanges of at least 128 bits in size and at least half the output size of the negotiated pseudorandom function (PRF) hash. Nonces are generated using the function RANDOM_numberGenerator() which generates numbers that meet the requirements specified in FCS_RBG_EXT.1 for random bit generation.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Each of the following tests shall be performed for each version of IKE selected in the FCS_IPSEC_EXT.1.5 protocol selection:

a) Test 1: If the first selection is chosen, the evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

b) Test 2: If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement.

See TSS assurance activity above.

2.2.15.11 NDcPP22e:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer.

Section 6.2 in the ST states that Diffie-Hellman (DH) groups 14, 19 and 20 are supported for both IKEv1 and IKEv2.

This is consistent with the DH groups specified in the requirement. In the IKEv1 phase 1 and phase 2 and IKEv2 IKE_SA and IKE_CHILD exchanges, the TOE and peer will agree on the best DH group both can support. When the TOE initiates the IKE negotiation, the DH group is sent in order according to the peer's configuration. When the TOE receives an IKE proposal, it will select the first match and the negotiation will fail if there is no match.

Guidance Assurance Activities: The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement.

Section "FCS_IPSEC_EXT.1.11" of the Admin Guide explains that ArubaOS supports DH groups 14, 19, and 20 and provides the commands to set the group value

Testing Assurance Activities: For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group.

Test 1 - The evaluator made an IPsec connection to an IPsec peer using each of the claimed DH groups. The evaluator was able to capture each DH group using a packet capture.

2.2.15.12 NDcPP22e:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation.

Section 6.2 in the ST indicates that the TOE implements the IPsec ESP protocol using AES-CBC-128, AES-CBC-192, AES-CBC-256, AES-GCM-128 and AES-GCM-256. The TOE uses AES-CBC-128, AES-CBC 192 and AES-CBC-256 algorithms to encrypt the IKEv1 or IKEv2 payload.

Section 6.2 in the ST states that the Administrator is responsible for ensuring that IKE/IPsec policies are configured so that the strength of the negotiated symmetric algorithm (in terms of the number of bits in the key) in the IKEv1 Phase 2 SA or IKEv2 CHILD_SA is less than or equal to the strength of the IKEv1 Phase 1 SA or IKEv2 IKE_SA. Administrators should configure IKE/IPsec policies so that the strength of the IKE association is greater than or equal to the strength of the IPsec tunnel (for example, by always using AES-256). However, if a misconfiguration is made, the TOE will reject the security association along with generating an audit log message.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator simply follows the guidance to configure the TOE to perform the following tests.

a) Test 1: This test shall be performed for each version of IKE supported. The evaluator shall successfully negotiate an IPsec connection using each of the supported algorithms and hash functions identified in the requirements.

b) Test 2: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP that selects an encryption algorithm with more strength than that being used for the IKE SA (i.e., symmetric algorithm with a key size larger than that being used for the IKE SA). Such attempts should fail.

c) Test 3: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish an IKE SA using an algorithm that is not one of the supported algorithms and hash functions identified in the requirements. Such an attempt should fail.

d) Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail.

Test 1 - The evaluator made an IPsec connection to an IPsec peer using each of the claimed hash functions identified in the requirements. The evaluator verified via packet capture and logs that the connection was successful with each of the claimed functions.

Test 2 - The evaluator configured a test peer to use a 128 bit key size for IKE and a 256 bit key size for ESP. The evaluator then attempted to connect the IPSec VPN between the test peer and the TOE and confirmed that the connection was rejected by the TOE. This was repeated for both IKEv1 and IKev2.

Test 3 - The evaluator attempted to establish a connection with an unsupported algorithm/hash combination. The connection attempt failed. This was repeated for both IKEv1 and IKev2.

Test 4 - This test was performed as part of test 3.

2.2.15.13 NDcPP22e:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: The evaluator ensures that the TSS identifies RSA and/or ECDSA as being used to perform peer authentication. The description must be consistent with the algorithms as specified in FCS_COP.1(2)/SigGen Cryptographic Operations (for cryptographic signature).

If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of IPsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

Section 6.2 in the ST indicates that both IKEv1 and IKEv2 can perform authentication using RSA and ECDSA certificates and pre-shared keys. This is consistent with the algorithms specified in FCS_COP.1/SigGen.

Pre-shared keys used for IPsec can be constructed of essentially any alphabetic character (upper and lower case), numerals, and special characters (e.g., “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”) and can be anywhere from 6-160 characters in length (e.g., 22 characters). The TOE requires suitable keys to be entered by an authorized administrator using a Web GUI or CLI function (i.e., keys are entered and not generated).

Guidance Assurance Activities: The evaluator ensures the guidance documentation describes how to set up the TOE to use certificates with RSA and/or ECDSA signatures and public keys.

The evaluator shall check that the guidance documentation describes how pre-shared keys are to be generated and established. The description in the guidance documentation shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

The evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA and ensure a valid certificate for that CA is loaded into the TOE and marked 'trusted'.

Section "FCS_IPSEC_EXT.1.13" in the Admin Guide explains that ArubaOS supports both RSA and ECDSA certificates. The Advanced Cryptography License must be installed to make use of ECDSA. Loading of certificates onto the controller for both authentication to peers and for verification of other peers is described in the User Guide for which links are provided in the Introduction section of the Admin Guide. Minimally, both a "server certificate" and a "trusted root CA" certificate must be loaded onto the controller in order to perform IPsec operations. Once these certificates are loaded on the controller, they can be configured for use with IPsec. Commands are provided for specifying the certificate and configuring an IKE policy to authenticate RSA or ECDSA certificates. This section also provides the commands for configuring the IPsec connection to use text or bit based pre-shared keys.

Section "FIA_PSK_EXT.1 (VPNGW)" in the Admin Guide provides further guidance for pre-shared keys including how the administrator can create an IKE policy that uses pre-shared keys by entering the pre shared key mapped by client IP address. When configuring the pre-shared key, the administrator must ensure that the PSK is at least 22 characters, contains at least one uppercase character, one lowercase character, one special character, and one digit. If the PSK is configured as a bit-based key, the 'key-hex' field should be used instead.

Testing Assurance Activities: For efficiency sake, the testing is combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1.

Testing in NDcPP22e_FCS_IPSEC_EXT.1.3-t1 demonstrates that a successful IPsec connection using an ECDSA certificate can be established.

Testing in NDcPP22e_FIA_X509_EXT.1_Rev.1-t2 demonstrates that a successful IPsec connection using an RSA certificate can be established.

Testing in VPNGW12_FIA_PSK_EXT.1-t1 demonstrates that a successful IPsec connection using PSK can be established.

2.2.15.14 NDcPP22e:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN). If the TOE simultaneously supports the same identifier type in the CN and SAN, the TSS shall describe how the TOE prioritizes the comparisons (e.g. the result of comparison if CN matches but SAN does not). If the location (e.g. CN or SAN) of non-DN identifier types must explicitly be configured as part of the reference identifier, the TSS shall state this. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate, including what field(s) are compared and which fields take precedence in the comparison.

Section 6.2 in the ST explains the TOE will only establish a trusted IPsec channel if the presented identifier in the received certificate matches the configured reference identifier, where the presented and reference identifiers are of the following type: Distinguished Name (DN). Fields within the DN are not individually selectable; the DN must be an exact match for the entire DN string

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 does not guarantee unique identifiers, 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.

Section "FIA_X509_EXT.1" in the Admin Guide explains how certificate checking is performed and describes how to configure the DN for site-to-site VPNs and client VPNs. The site-to-site discussion incudes a warning that Attempting to establish an IPsec tunnel while examining the log file (possibly after enabling "logging level debugging security") will generally show the exact DN string that must be configured, once it is received from the peer. For VPN clients instructions are provided for setting the client and revocation checking. The section also states that the TOE does not support the SAN extension.

Testing Assurance Activities: In the context of the tests below, a valid certificate is a certificate that passes FIA_X509_EXT.1 validation checks but does not necessarily contain an authorized subject.

The evaluator shall perform the following tests:

Test 1: (conditional) For each CN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes CN checking over SAN (through explicit configuration of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the SAN so it contains an incorrect identifier of the correct type (e.g. the reference identifier on the TOE is example.com, the CN=example.com, and the SAN:FQDN=otherdomain.com) and verify that IKE authentication succeeds.

Test 2: (conditional) For each SAN/identifier type combination selected, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the field in the peer's presented certificate and shall verify that the IKE authentication succeeds. If the TOE prioritizes SAN checking over CN (through explicit specification of the field when specifying the reference identifier or prioritization rules), the evaluator shall also configure the CN so it contains an incorrect identifier formatted to be the same type (e.g. the reference identifier on the TOE is DNS-ID; identify certificate has an identifier in SAN with correct DNS-ID, CN with incorrect DNS-ID (and not a different type of identifier)) and verify that IKE authentication succeeds.

Test 3: (conditional) For each CN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with the CN so it contains the valid identifier followed by ''. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the evaluator shall configure the SAN so it matches the reference identifier.

b) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the CN without the '' and verify that IKE authentication fails.

Test 4: (conditional) For each SAN/identifier type combination selected, the evaluator shall:

a) Create a valid certificate with an incorrect identifier in the SAN. The evaluator shall configure a string representation of the correct identifier in the DN. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the addition/modification shall be to any non-CN field of the DN. Otherwise, the addition/modification shall be to the CN.

b) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the correct identifier (expected in the SAN) and verify that IKE authentication fails.

Test 5: (conditional) If the TOE supports DN identifier types, the evaluator shall configure the peer's reference identifier on the TOE (per the administrative guidance) to match the subject DN in the peer's presented certificate and shall verify that the IKE authentication succeeds.

Test 6: (conditional) If the TOE supports DN identifier types, to demonstrate a bit-wise comparison of the DN, the evaluator shall create the following valid certificates and verify that the IKE authentication fails when each certificate is presented to the TOE:

a) Duplicate the CN field, so the otherwise authorized DN contains two identical CNs.

b) Append '' to a non-CN field of an otherwise authorized DN.

Test 1 - This test is not applicable as the TOE only supports DN.

Test 2 - This test is not applicable as the TOE only supports DN.

Test 3 - This test is not applicable as the TOE only supports DN.

Test 4 - This test is not applicable as no SAN type identifiers are claimed.

Test 5 - The evaluator configured a test peer to send an authentication certificate with an authorized DN and confirmed that the connection succeeded.

Test 6 - a) The evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate but with a DN containing a duplicate CN. The evaluator attempted to establish an IPsec connection and confirmed that the TOE to rejected the certificate containing a duplicate CN.

Test 6 - b) The evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate in which the Organization field of the DN has a trailing null character (71) appended. In each case, the evaluator attempted to establish an IPsec connection, and confirmed that the TOE rejected the certificate containing a DN Organization with an appended null character.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.16 IPsec Protocol - per TD0657 (VPNGW12:FCS_IPSEC_EXT.1)

2.2.16.1 VPNGW12:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.2 VPNGW12:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.3 VPNGW12:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.4 VPNGW12:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.5 VPNGW12:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.6 VPNGW12:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.7 VPNGW12:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.8 VPNGW12:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.9 VPNGW12:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.10 VPNGW12:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.11 VPNGW12:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.12 VPNGW12:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.13 VPNGW12:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.16.14 VPNGW12:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: All existing activities regarding 'Pre-shared keys' apply to all selections including pre-shared keys. If any selection with 'Pre-shared keys' is included, the evaluator shall check to ensure that the TSS describes how the selection works in conjunction with the authentication of IPsec connections.

Section 6.10 of the ST states that the TOE uses the IPsec/IKE protocol with pre-shared keys or certificates to establish a trusted channel between itself and the external authentication server, syslog server, NTP server, and other VPN Gateways (ie. site-to-site VPN).

Component Guidance Assurance Activities: If any selection with 'Pre-shared Keys' is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

Section "FIA_PSK_EXT.1 (VPNGW)" of the Admin Guide describes how to configure the TOE to use Pre-shared Keys for authentication and additionally how to configure them on the TOE. The section describes these steps for both text-based and hex-based Pre-shared Keys.

Component Testing Assurance Activities: None Defined

2.2.17 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1)

2.2.17.1 NDcPP22e:FCS_NTP_EXT.1.1

TSS Assurance Activities: The evaluator shall examine the TSS to ensure it identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained. The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp.

Section 6.2 of the ST states that the TOE supports NTPv4.

Section 6.10 of the ST states that the TOE uses the IPsec/IKE protocol with pre-shared keys or certificates to establish a trusted channel between itself and the NTP server.

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the version of NTP supported, how to configure multiple NTP servers for the TOE's time source and how to configure the TOE to use the method(s) that are selected in the ST.

Section “FPT_STM_EXT.1” in the Admin Guide provides instructions for configuring NTPv4 servers. Multiple time servers can be configured with the use of the ‘ntpserver’ command. The TOE supports configuration of 3 NTP servers. If a remote NTP server is used, the administrator must ensure that the connection is protected via IPsec.

Testing Assurance Activities: The version of NTP selected in element 1.1 and specified in the ST shall be verified by observing establishment of a connection to an external NTP server known to be using the specified version(s) of NTP. This may be combined with tests of other aspects of FCS_NTP_EXT.1 as described below.

This test was performed as part of FCS_NTP_EXT.1.4 where a valid connection with an NTP server was established.

2.2.17.2 NDcPP22e:FCS_NTP_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: For each of the secondary selections made in the ST, the evaluator shall examine the guidance document to ensure it instructs the Security Administrator how to configure the TOE to use the algorithms that support the authenticity of the timestamp and/or how to configure the TOE to use the protocols that ensure the integrity of the timestamp.

Assurance Activity Note:

Each primary selection in the SFR contains selections that specify a cryptographic algorithm or cryptographic protocol. For each of these secondary selections made in the ST, the evaluator shall examine the guidance documentation to ensure that the documentation instructs the administrator how to configure the TOE to use the chosen option(s).

Section “FPT_STM_EXT.1” in the Admin Guide provides instructions for configuring NTPv4 servers. If a remote NTP server is used, the administrator must ensure that the connection is protected via IPsec.

Section “FCS_IPSEC_EXT.1” in the Admin Guide provides instructions for configuring the TOE to use IPsec.

Testing Assurance Activities: The cryptographic algorithms selected in element 1.2 and specified in the ST will have been specified in an FCS_COP SFR and tested in the accompanying Evaluation Activity for that SFR. Likewise, the cryptographic protocol selected in in element 1.2 and specified in the ST will have been specified in an FCS SFR and tested in the accompanying Evaluation Activity for that SFR.

[Conditional] If the message digest algorithm is claimed in element 1.2, the evaluator will change the message digest algorithm used by the NTP server in such a way that the new value does not match the configuration on the TOE and confirms that the TOE does not synchronize to this time source.

The evaluator shall use a packet sniffer to capture the network traffic between the TOE and the NTP server. The evaluator uses the captured network traffic, to verify the NTP version, to observe time change of the TOE and uses the TOE's audit log to determine that the TOE accepted the NTP server's timestamp update.

The captured traffic is also used to verify that the appropriate message digest algorithm was used to authenticate the time source and/or the appropriate protocol was used to ensure integrity of the timestamp that was transmitted in the NTP packets.

This test was performed as part of FCS_IPSEC_EXT.1 protocol testing.

2.2.17.3 NDcPP22e:FCS_NTP_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the TOE to not accept broadcast and multicast NTP packets that would result in the timestamp being updated.

Section “FPT_STM_EXT.1” in the Admin Guide states that the TOE, by default, does not accept broadcast and multicast NTP packets.

Testing Assurance Activities: The evaluator shall configure NTP server(s) to support periodic time updates to broadcast and multicast addresses. The evaluator shall confirm the TOE is configured to not accept broadcast and multicast NTP packets that would result in the timestamp being updated. The evaluator shall check that the time stamp is not updated after receipt of the broadcast and multicast packets.

The evaluator configured the NTP server to support periodic time updates to broadcast and multicast addresses and then verified that the NTP server sent broadcast and multicast packets and that the TOE timestamp did not change to the NTP server time.

2.2.17.4 NDcPP22e:FCS_NTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test 1: The evaluator shall confirm the TOE supports configuration of at least three (3) NTP time sources. The evaluator shall configure at least three NTP servers to support periodic time updates to the TOE. The evaluator shall confirm the TOE is configured to accept NTP packets that would result in the timestamp being updated from each of the NTP servers. The evaluator shall check that the time stamp is updated after receipt of the NTP packets. The purpose of this test to verify that the TOE can be configured to synchronize with multiple NTP servers. It is up to the evaluator to determine that the multi- source update of the time information is appropriate and consistent with the behaviour prescribed by the RFC 1305 for NTPv3 and RFC 5905 for NTPv4.

Test 2: (The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers).

The evaluator shall confirm that the TOE would not synchronize to other, not explicitly configured time sources by sending an otherwise valid but unsolicited NTP Server responses indicating different time from the TOE's current system time. This rogue time source needs to be configured in a way (e.g. degrade or disable valid and configured NTP servers) that could plausibly result in unsolicited updates becoming a preferred time source if they are not discarded by the TOE. The TOE is not mandated to respond in a detectable way or audit the occurrence of such unsolicited updates. The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers. It is up to the evaluator to craft and transmit unsolicited updates in a way that would be consistent with the behaviour of a correctly-functioning NTP server.

(TD0528 applied)

Test 1: The evaluator first started by using the TOE's CLI to configure 3 different NTP servers on the TOE and use them for time synchronization. The evaluator then started a packet capture and confirmed that the TOE synchronizes its' time through one of the NTP servers as can be seen in the full packet capture below. The evaluator also used the TOE's audit log to confirm that the TOE synchronized its' time through the NTP servers.

Test 2: The evaluator configured the TOE with 3 unreachable NTP connections and observed the current time on the TOE and NTP server. The evaluator set the time on the NTP server ahead by 1 hour and observed that NTP servers were unreachable The evaluator collected network traffic while monitoring the time on the TOE while an untrusted NTP server was configured to broadcast to the TOE. As can be seen in this packet capture, the TOE ignored the broadcast and attempted to update time using traditional authenticated updates with the unreachable NTP servers.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.18 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.18.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.18.2 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 6.2 in the ST states that the TOE uses a software based random bit generator that complies with AES-256 CTR_DRBG when operating in FIPS mode. AES-256 is used in conjunction with a minimum of 256 bits of entropy accumulated from two hardware based noise sources: a hardware random number generator provided by a Trusted Platform Module chip (TPM RNG Entropy source) and the network processor that serves as the main CPU for the Mobility Controller (CPU RNG Entropy source). The third entropy source used for virtual controllers is the Jitter RNG Entropy source. The jitterentropy implementation is based upon noise gathered by measuring the jitter in CPU execution time. The timing for the execution of a fixed piece of code is unrepeatable in any machine. The Intel DRNG kernel replenishes the entropy pool by both rngd and jitterentropy daemons. There is a fourth entropy source that is software based and supplied by the Linux Kernel RNG, however, Aruba does not view this as a useful source of entropy as the only sources of software/kernel entropy events used in the Aruba modules are messages between the dataplane and control plane, and flash memory access. While they do feed the Linux Entropy pool, they do not significantly contribute to it. Aruba's use of the Linux kernel RNG is therefore primarily for building up and storing entropy from hardware noise sources.

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) document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section "FCS_RBG_EXT.1" of the Admin Guide states no configuration is necessary for the 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 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 section entitled, "CAVP Certificate Justification" earlier in this

document.

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

2.2.19.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.19.2 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 6.2 in the ST explains that SSHv2 supports both public-key and password-based authentication and can be configured. The TOE supports SSHv2 with AES (CBC and CTR) 128 or 256 bit ciphers, in conjunction with HMACSHA-

1 and HMAC-SHA1-96 and RSA using the following key exchange methods: diffie-hellman-group14-sha1. The TOE supports SSH_RSA for server authentication which is consistent with FCS_SSHS_EXT.1.5.

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 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: This test has been performed in FIA_UIA_EXT.1-t1 where the evaluator demonstrated a login using public keys. The TOE claims support for RSA public keys only.

Test 2: The evaluator attempted to connect to the TOE using a SSH client using a pubkey not configured on the TOE. The evaluator found that an unconfigured pubkey would not be used to establish an SSH session and the TOE would revert back to password authentication.

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 test was performed as part of Test 3 as described above.

2.2.19.3 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 6.2 in the ST states that the TOE limits packets to 256k bytes. As SSH packets are being received, the TOE uses a buffer to build all packet information. Once complete, the packet is checked to ensure it can be appropriately decrypted. However, if it is not complete when the buffer becomes full (256k bytes) the packet will be dropped.

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 of 256k bytes.

The evaluator observed that the TOE rejected the packet and the connection was closed.

2.2.19.4 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 6.2 in the ST states that the TOE supports SSHv2 with AES (CBC and CTR) 128 or 256 bit ciphers, in conjunction with HMAC-SHA-1 and HMAC-SHA1-96 and RSA using the following key exchange methods: diffiehellman-group14-sha1, ecdh-sha2-nistp256, and ecdh-sha2-nistp384. 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 “FCS_SSHS_EXT.1” of the Admin Guide states that no configuration is needed to specify the permitted algorithms after ‘fips enable’ has been set. The controller will attempt negotiations using AES128-CBC, AES256

CBC, AES128-CTR, and AES256-CTR. This set of algorithms is consistent with the requirement and the description in the TSS.

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.

The evaluator attempted to establish an SSH connection with each of the following SSH algorithms to encrypt the session: AES128-CBC and AES256-CBC, AES128-CTR, AES256-CTR. The evaluator captured packets associated with each of the connection attempts and observed that using these algorithms the evaluator was able to successfully connect to the TOE.

2.2.19.5 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)

Section 6.2 in the ST states that the TOE supports SSH_RSA for server authentication. 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).

Section “FCS_SSHS_EXT.1” of the Admin Guide provides instructions for disabling dsa and configuring ssh-rsa as the public key algorithm. This is consistent with the requirement and the TSS which indicates that ssh-rsa is supported for server authentication.

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.

(TD0631 applied)

Test 1: The evaluator attempted to connect to the TOE using a SSH client alternately using each of the authentication algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator confirmed that the TOE supports the ssh-rsa algorithms as claimed. The evaluator followed up with a disallowed authentication algorithm and confirmed that it was not accepted.

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

2.2.19.6 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 6.2 in the ST states the TOE supports SSHv2 with AES (CBC) 128 or 256 bit ciphers, in conjunction with HMAC-SHA-1,HMAC-SHA1-96, and HMAC-SHA2-256 and RSA (and RSA 256 and 512) for integrity. 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 “FCS_SSHS_EXT.1” of the Admin Guide states that no configuration is needed to specify the permitted algorithms after ‘fips enable’ has been set. The controller will attempt negotiations using AES128-CBC, AES256-

CBC, AES128-CTR, and AES256-CTR in conjunction with HMAC-SHA-1 and HMAC-SHA1-96 and RSA using the following key exchange method: diffie-hellman-group14-sha1.

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.

Test 1: The evaluator attempted to connect to the TOE using a SSH client alternately using each of the MAC algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator followed up with a disallowed MAC algorithm to ensure it was not accepted. The evaluator observed that only the claimed hmac-sha1 and hmac-sha2-256 algorithms were able to successfully connect to the TOE.

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

2.2.19.7 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 6.2 in the ST states the TOE supports SSHv2 using the following key exchange methods: diffie-hellman-group14-sha1, 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 “FCS_SSHS_EXT.1” of the Admin Guide states that no configuration is needed to specify the permitted algorithms after ‘fips enable’ has been set. The controller will attempt negotiations using AES128 CBC, AES256-

CBC, AES128-CTR, and AES256-CTR in conjunction with HMAC-SHA-1 and HMAC-SHA1-96 and RSA using the following key exchange method: diffie-hellman-group14-sha1.

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. The evaluator attempted to connect to the TOE using DiffieHellman-Group1. The TOE rejected the attempt as expected.

Test 2 - The evaluator attempted to connect to the TOE using a SSH client alternately using an unallowable key exchange algorithm which fails and then each of the key exchange algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator attempted to connect to the TOE using an SSH client with the claimed key exchange method: DH group 14 and confirmed that the connection was successful.

2.2.19.8 NDcPP22e:FCS_SSHS_EXT.1.8

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

a) Both thresholds are checked by the TOE.

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

Section 6.2 of the ST states that SSHv2 connections are rekeyed after a period of no longer than one hour or no more than one gigabyte of transmitted data. Both of these thresholds are checked by the TOE and rekeying is performed upon reaching the threshold that is hit first.

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 “FCS_SSHS_EXT.1” of the Admin Guide asserts that SSH rekey intervals are non-configurable and are set to a maximum time interval of one (1) hour or 512M.

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 (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.

Test - The evaluator attempted to connect to the TOE using an SSH client generating 1GB of data and confirmed via logs and packet captures that the rekey happened before the threshold was reached. The evaluator then attempted to connect to the TOE using a SSH client waiting an hour and confirmed via logs and packet captures that the rekey happened before the 1 hour threshold was reached.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.20 TLS Server Protocol Without Mutual Authentication - per TD0635 (NDcPP22e:FCS_TLSS_EXT.1)

2.2.20.1 NDcPP22e:FCS_TLSS_EXT.1.1

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 are identical to those listed for this component.

Section 6.2 of the ST states that the TOE supports TLSv1.2. Any other SSL/TLS versions are not supported by the TOE and such connection attempts will be rejected. Remote administration via the Web UI is protected using TLS/HTTPS. The following cipher suites are implemented by the TOE, and other than identifying the cipher no configuration is needed to support any of these ciphers:

TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,

TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA as defined in RFC 4492,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

This list matches those 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 (for instance, the set of ciphersuites advertised by the TOE may have to be restricted to meet the requirements).

Section "FCS_TLSS_EXT.1" in the Admin Guide provides a list of the supported TLSv1.2 ciphersuites and states that no configuration is required to set the permitted cipher suites once 'fips enable' has been entered on the controller. The list of cipher suites is consistent with those claimed in the ST.

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 send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

Test 3: The evaluator shall perform the following modifications to the traffic:

a) Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.

b) (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)

The evaluator shall use one of the claimed ciphersuites to complete a successful handshake and observe transmission of properly encrypted application data. The evaluator shall verify that no Alert with alert level Fatal (2) messages were sent.

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

Test 1: The evaluator attempted to connect to the TOE using each of the claimed ciphersuites. A packet capture was obtained for each connection attempt. The evaluator confirmed that each connection was successful.

Test 2: The evaluator sent a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verified that the server denied all such connection attempts. Additionally, the evaluator sent a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verified that the server denied the connection.

Test 3: (a, b): The evaluator made connection attempts from a client to the TOE. The client implementation of the TLS protocol was modified as stated in parts a and b of the assurance activity. The evaluator observed in packet captures that the connections are rejected for each scenario. Also, the evaluator established a TLS connection from a test server to the TOE, while capturing packets associated with that connection. The evaluator located the ChangeCipherSpec message (Content type hexadecimal 14) in the packet capture and found it to be followed by an EncryptedHandshakeMessage (Content type hexadecimal 16). The evaluator examined the payload of the EncryptedHandshakeMessage to determine if it contained a cleartext or encrypted version of the required Finished message. A Cleartext version would begin with a Content type hexadecimal value of 14, while an encrypted version would (for at least one of three test messages) not contain a Content type hexadecimal value of 14.

2.2.20.2 NDcPP22e:FCS_TLSS_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS contains a description of how the TOE technically prevents the use of old SSL and TLS versions.

Section 6.2 of the ST states the TOE supports TLSv1.2. Any other SSL/TLS versions are not supported by the TOE and such connection attempts will be rejected.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Section “FCS_TLSS_EXT.1” in the Admin Guide provides the commands used to configure the TLS web-server profile including the command ‘ssl-protocol’ used to configure TLSv1.2.

Testing Assurance Activities: The evaluator shall send a Client Hello requesting a connection for all mandatory and selected protocol versions in the SFR (e.g. by enumeration of protocol versions in a test client) and verify that the server denies the connection for each attempt.

The evaluator attempted to establish a TLS session on the TOE with each of the unsupported versions of SSL and TLS. The evaluator used a network sniffer to capture the session negotiation and observed that the expected protocol and version were offered and rejected during negotiation.

2.2.20.3 NDcPP22e:FCS_TLSS_EXT.1.3

TSS Assurance Activities: If using ECDHE and/or DHE ciphers, the evaluator shall verify that the TSS lists all EC Diffie-Hellman curves and/or Diffie-Hellman groups used in the key establishment by the TOE when acting as a TLS Server. For example, if the TOE supports TLS_DHE_RSA_WITH_AES_128_CBC_SHA cipher and Diffie-Hellman parameters with size 2048 bits, then list Diffie-Hellman Group 14.

Section 6.2 in the ST states the TOE performs RSA key establishment with key sizes 2048 bits and generates DH parameters over NIST curve secp256r1.

Guidance Assurance Activities: The evaluator shall verify that any configuration necessary to meet the requirement must be contained in the AGD guidance.

Section “FCS_TLSS_EXT.1” in the Admin Guide provides a list of the supported TLSv1.2 ciphersuites and states that no configuration is required to set the permitted cipher or the associated key agreement parameters once ‘fips enable’ has been entered on the controller. The TOE performs RSA key establishment with key sizes 2048 bits and generates DH parameters over NIST curve secp256r1.

Testing Assurance Activities: Test 1: [conditional] If ECDHE ciphersuites are supported:

a) The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single supported elliptic curve specified in the Elliptic Curves Extension. The Evaluator shall verify (though a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange message and successfully establishes the connection.

b) The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single unsupported elliptic curve (e.g. secp192r1 (0x13)) specified in RFC4492, chap. 5.1.1. The evaluator shall verify that the TOE does not send a Server Hello message and the connection is not successfully established.

Test 2: [conditional] If DHE ciphersuites are supported, the evaluator shall repeat the following test for each supported parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use a supported Diffie-Hellman parameter size. The evaluator shall attempt a connection using a supported DHE ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a Server Key Exchange Message where p Length is consistent with the message are the ones configured Diffie-Hellman parameter size(s).

Test 3: [conditional] If RSA key establishment ciphersuites are supported, the evaluator shall repeat this test for each RSA key establishment key size. If any configuration is necessary, the evaluator shall configure the TOE to perform RSA key establishment using a supported key size (e.g. by loading a certificate with the appropriate key size). The evaluator shall attempt a connection using a supported RSA key establishment ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a certificate whose modulus is consistent with the configured RSA key size.

Test - The TOE supports connections with 2048-bit RSA certs; as well as with P-256 curves. The evaluator attempted to establish a TLS session with the TOE when the evaluator's client specified only one key exchange method in the Client Hello. The evaluator observed the connection succeeded only when the TOE was configured for the various key exchange methods attempted.

The evaluator also attempted a connection using ECDHE w/ P-192 curve and observed the TOE rejected TLS negotiation.

2.2.20.4 NDcPP22e:FCS_TLSS_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes if session resumption based on session IDs is supported (RFC 4346 and/or RFC 5246) and/or if session resumption based on session tickets is supported (RFC 5077).

If session tickets are supported, the evaluator shall verify that the TSS describes that the session tickets are encrypted using symmetric algorithms consistent with FCS_COP.1/DataEncryption. The evaluator shall verify that the TSS identifies the key lengths and algorithms used to protect session tickets.

If session tickets are supported, the evaluator shall verify that the TSS describes that session tickets adhere to the structural format provided in section 4 of RFC 5077 and if not, a justification shall be given of the actual session ticket format.

Section 6.2 of the ST states that TLS session resumption is supported for TLS connection of the HTTPS/TLS server. The session tickets are encrypted according to the TLS negotiated symmetric encryption algorithm. This is consistent with FCS_COP.1/DataEncryption claims in this ST – AES used in CBC and GCM modes and key sizes of 128 and 256 bits. The session tickets adhere to the structural format provided in section 4 of RFC 5077.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test Objective: To demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption).

Test 1 [conditional]: If the TOE does not support session resumption based on session IDs according to RFC4346 (TLS1.1) or RFC5246 (TLS1.2) or session tickets according to RFC5077, the evaluator shall perform the following test:

a) The client sends a Client Hello with a zero-length session identifier and with a SessionTicket extension containing a zero-length ticket.

b) The client verifies the server does not send a NewSessionTicket handshake message (at any point in the handshake).

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps: Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

d) The client completes the TLS handshake and captures the SessionID from the ServerHello.

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

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

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

a) The evaluator shall conduct a successful handshake and capture the TOE-generated session ID in the Server Hello message. The evaluator shall then initiate a new TLS connection and send the previously captured session ID to show that the TOE resumed the previous session by responding with ServerHello containing the same SessionID immediately followed by ChangeCipherSpec and Finished messages (as shown in Figure 2 of RFC 4346 or RFC 5246).

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

Test 3 [conditional]: If the TOE supports session tickets according to RFC5077, the evaluator shall carry out the following steps (note that for each of these tests, it is not necessary to perform the test case for each supported version of TLS):

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with a ServerHello with an empty SessionTicket extension, NewSessionTicket, ChangeCipherSpec and Finished messages (as seen in figure 2 of RFC 5077).

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

Test 1: Not applicable. The TOE claims to support session resumption using session tickets.

Test 2: Not applicable. The TOE claims to support session resumption using session tickets.

Test 3a: The evaluator established a successful TLS handshake with the TOE and captured the TOE-generated session ticket. The evaluator then attempted to establish a TLS session with the previously generated session ticket. The evaluator verified that the TOE correctly reused the client's proposed session ticket and the session was successfully resumed.

Test 3b: The evaluator repeated the steps of the previous test, but this time attempted to resume the TLS session with a modified session ticket. The evaluator confirmed that the TOE implicitly rejected the session ticket by performing a full handshake, and sent a new session ticket.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 User data protection (FDP)

2.3.1 Full Residual Information Protection (STFFW14e:FDP_RIP.2)

2.3.1.1 STFFW14e:FDP_RIP.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: 'Resources' in the context of this requirement are network packets being sent through (as opposed to 'to', as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs.

Section 6.3 explains network packets are received in memory buffers pre-allocated at boot time. The buffers are populated in the network interface receive ring. When the CPU receives network packets from the network interface, the CPU allocates a free buffer from the pre-allocated pool and replenishes the receive ring. When the CPU has finished packet processing, the CPU adds the memory buffer associated with this network packet to the free buffer pool. Packets read from the buffers are always the same size as those written, so no explicit zeroing or overwriting of buffers on allocation is required (i.e., packet content is always overwritten before being read).

Each pre-allocated memory buffer is fixed at 1792 bytes and is sufficient to hold a standard Ethernet frame. When a frame is received by the network interface, a proprietary header is prepended onto the frame indicating its length, among other parameters. The frame is then sent to the CPU. The CPU reads the length out of the proprietary header and uses this to process the frame out of the buffer. For standard Ethernet frames, only a single frame is placed into a memory buffer - buffers do not contain multiple frames. Jumbo packets are supported - these must be split across multiple memory buffers. The proprietary header ensures that the packet is reconstructed correctly.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4 Firewall (FFW)

2.4.1 Stateful Traffic Filtering (STFFW14e:FFW_RUL_EXT.1)

2.4.1.1 STFFW14e:FFW_RUL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.2 STFFW14e:FFW_RUL_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS describes a stateful packet filtering policy and the following attributes are identified as being configurable within stateful traffic filtering rules for the associated protocols:

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

o Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

The evaluator shall verify that each rule can identify the following actions: permit or drop with the option to log the operation. The evaluator shall verify that the TSS identifies all interface types subject to the stateful packet filtering policy and explains how rules are associated with distinct network interfaces.

Section 6.4 in the ST explains that the TOE performs stateful packet filtering. Filtering rules may be applied to appliance Ethernet interfaces and to user-roles (wireless clients connecting through APs are placed into userroles). The TOE is comprised of a data plane and a control plane. Network packets are processed in the data plane, which is the first component that is initialized. Network interfaces are not brought 'up' until initialization is complete and the data plane operating system is fully initialized. All packet level enforcement is performed within the data plane. In case of system error, packets are dropped by default. The control plane operating system (management interfaces) boots simultaneously.

The TOE supports stateful processing of the following protocols:

RFC 792 (ICMPv4)

RFC 4443 (ICMPv6)

RFC 791 (IPv4)

RFC 2460 (IPv6)

RFC 793 (TCP)

RFC 768 (UDP)

The TOE allows the definition of a stateful packet filtering policy based on the following attributes that are used to

define rules (based on the actions: permit, deny or log) for the associated protocols:

Note: ICMPv4 Code 134 is an unsolicited router advertisement which is discarded/dropped by the TOE. All other

traffic is handled based upon access control lists configured on the TOE.

ICMPv4

o Source Address

o Destination Address

o Type

o Code1

ICMPv6

o Source Address

o Destination Address

o Type

o Code

IPv4

o Source address

o Destination Address

o Transport Layer Protocol

IPv6

o Source address

o Destination Address

o Transport Layer Protocol

o IPv6 Extension header type

TCP

o Source Address

o Destination Address

o Source Port

o Destination Port

o Sequence number

o Flags

UDP

o Source Address

o Destination Address

o Source Port

o Destination Port

Guidance Assurance Activities: The evaluators shall verify that the guidance documentation identifies the following attributes as being configurable within stateful traffic filtering rules for the associated protocols:

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

o Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

The evaluator shall verify that the guidance documentation indicates that each rule can identify the following actions: permit, drop, and log.

The evaluator shall verify that the guidance documentation explains how rules are associated with distinct network interfaces.

Section "FFW_RUL_EXT.1.2/3/4" of the Admin Guide lists each protocol and their attributes from the requirement and provides an example for how to set a firewall rule for each protocol including setting a permit, deny and log rule.

Section "FFW_RUL_EXT.1.5" in the Admin Guide provides the commands and examples for assigning rules to an interface.

Testing Assurance Activities: Test 1: The evaluator shall use the instructions in the guidance documentation to test that stateful packet filter firewall rules can be created that permit, drop, and log packets for each of the following attributes:

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

o Transport Layer Protocol and where defined by the ST author, Extension Header Type, Extension Header Fields

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

Test 2: Repeat the test evaluation activity above to ensure that stateful traffic filtering rules can be defined for each distinct network interface type supported by the TOE.

Note that these test activities should be performed in conjunction with those of FFW_RUL_EXT.1.9 where the effectiveness of the rules is tested. The test activities for FFW_RUL_EXT.1.9 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfil the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

Test 1: The evaluator configured firewall rules for testing of the other FFW_RUL_EXT.1 tests (including FFW_RUL_EXT.1.9) using instructions provided within the administrative guidance and found all necessary instructions were provided accurately.

Test 2: The evaluator configured firewall rules for testing of the other FFW_RUL_EXT.1 tests (including FFW_RUL_EXT.1.9) using instructions provided within the administrative guidance and found all necessary instructions were provided accurately.

2.4.1.3 STFFW14E:FFW_RUL_EXT.1.3

TSS Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Guidance Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Testing Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2.

2.4.1.4 STFFW14E:FFW_RUL_EXT.1.4

TSS Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Guidance Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Testing Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

2.4.1.5 STFFW14E:FFW_RUL_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS identifies the protocols that support stateful session handling. The TSS shall identify TCP, UDP, and, if selected by the ST author, also ICMP.

The evaluator shall verify that the TSS describes how stateful sessions are established (including handshake processing) and maintained.

The evaluator shall verify that for TCP, the TSS identifies and describes the use of the following attributes in session determination: source and destination addresses, source and destination ports, sequence number, and individual flags.

The evaluator shall verify that for UDP, the TSS identifies and describes the following attributes in session determination: source and destination addresses, source and destination ports.

The evaluator shall verify that for ICMP (if selected), the TSS identifies and describes the following attributes in session determination: source and destination addresses, other attributes chosen in FFW_RUL_EXT.1.5.

The evaluator shall verify that the TSS describes how established stateful sessions are removed. The TSS shall describe how connections are removed for each protocol based on normal completion and/or timeout conditions.

The TSS shall also indicate when session removal becomes effective (e.g., before the next packet that might match the session is processed).

Section 6.4 in the ST explains the TOE allows the definition of a stateful packet filtering policy based on the following attributes that are used to define rules (based on the actions: permit, deny or log) for the associated protocols:

ICMPv4

o Source Address

o Destination Address

o Type

o Code2

ICMPv6

o Source Address

o Destination Address

o Type

o Code

IPv4

o Source address

o Destination Address

o Transport Layer Protocol

IPv6

o Source address

o Destination Address

o Transport Layer Protocol

o IPv6 Extension header type

TCP

o Source Address

o Destination Address

o Source Port

o Destination Port

o Sequence number

o Flags

UDP

o Source Address

o Destination Address

o Source Port

o Destination Port

The TOE allows the stateful packet filtering rules to be assigned to each distinct network interface. The interfaces can be viewed through the CLI command "show interface". Stateful session tracking is established and maintained by the data plane operating system thorough inspection of network packets, including handshake transactions.

The algorithm applied to incoming packets is as follows:

Check for IP fragments and assemble.

Parse and identify protocol in the IP packet.

Perform length checks and apply default rules.

Enforce interface access-lists (ACLs) if configured.

Lookup session. If exists, then apply corresponding session / stateful ACLs.

Add session if it doesn't exist. (stateful if the protocol warrants, as listed above.). If stateful also open reverse ports for returning/stateful session. The protocol attributes identified above are used in session determination and will include IP protocol attributes for higher level

protocols. TCP processing is further described below. Generate log message, if the 'log'

keyword is configured in the rule.

Derive role for the user and apply role based ACLs. If no role ACLs, then apply default ACLs (deny).

Perform bandwidth contract enforcement.

Perform NATing if required.

During TCP session establishment, the session is identified by source IP, destination IP, source port, and destination port. By default, the three-way handshake is not enforced before data is allowed. This behavior can be changed using the configuration “firewall enforce-tcp-handshake”. Once the session has been established in the datapath session table, future packets which match the source IP, destination IP, source port, and destination port are processed by the “fast path” which was previously programmed during session establishment.

By default, TCP sequence numbers are ignored. This behavior can be changed using the configuration “firewall enforce-tcp-sequence”. Where the controller is used as a wired firewall, sequence number checks do provide some value and in this case the feature should be enabled. TCP flags are largely ignored by the firewall, with the obvious exception of SYN/ACK during session establishment, and FIN/RST during session teardown.

Sessions are removed if the relevant protocol frame is received (e.g. TCP RST) or aged out after a configurable timeout of inactivity (whichever occurs first). Session removal is immediate. Audit log messages are not generated when a session is removed.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes stateful session behaviours. For example, a TOE might not log packets that are permitted as part of an existing session.

Sections “FFW_RUL_EXT.1.2/3/4” and “FFW_RUL_EXT.1.5” in the Admin Guide describe stateful session behavior and configuration. The TOE provides stateful session firewalls for communication sent through its interfaces. The Aruba Policy Enforcement Firewall provides context-based controls to enforce application-layer security and prioritization. The TOE can enforce network access policies that specify who may access the network, with what device, and what area of the network they may access.

The TOE implements a full stateful firewall instance around every user, tightly controlling what the user is permitted to do and providing separation between user classes. When session access control lists are configured with logging specified, traffic sent through the session will be logged in syslog and recorded within statistical counters that can be viewed by an administrator.

Testing Assurance Activities: Test 1: The evaluator shall configure the TOE to permit and log TCP traffic. The evaluator shall initiate a TCP session. While the TCP session is being established, the evaluator shall introduce session establishment packets with incorrect flags to determine that the altered traffic is not accepted as part of the session (i.e., a log event is generated to show the ruleset was applied). After a TCP session is successfully established, the evaluator shall alter each of the session determining attributes (source and destination addresses, source and destination ports, sequence number, flags) one at a time in order to verify that the altered packets are not accepted as part of the established session.

Test 2: The evaluator shall terminate the TCP session established per Test 1 as described in the TSS. The evaluator shall then immediately send a packet matching the former session definition in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 3: The evaluator shall expire (i.e., reach timeout) the TCP session established per Test 1 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 4: The evaluator shall configure the TOE to permit and log UDP traffic. The evaluator shall establish a UDP session. Once a UDP session is established, the evaluator shall alter each of the session determining attributes (source and destination addresses, source and destination ports) one at a time in order to verify that the altered packets are not accepted as part of the established session.

Test 5: The evaluator shall expire (i.e., reach timeout) the UDP session established per Test 4 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 6: If ICMP is selected, the evaluator shall configure the TOE to permit and log ICMP traffic. The evaluator shall establish a session for ICMP as defined in the TSS. Once an ICMP session is established, the evaluator shall alter each of the session determining attributes (source and destination addresses, other attributes chosen in FFW_RUL_EXT.1.5) one at a time in order to verify that the altered packets are not accepted as part of the established session.

Test 7: If applicable, the evaluator shall terminate the ICMP session established per Test 6 as described in the TSS. The evaluator shall then immediately send a packet matching the former session definition in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 8: The evaluator shall expire (i.e., reach timeout) the ICMP session established per Test 6 as described in the TSS. The evaluator shall then send a packet matching the former session in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 1: The evaluator configured the TOE to allow and log network traffic for a valid TCP session. The evaluator started transmitting packets from a test server to establish a valid TCP session. The packets being sent were constructed such that they would be permitted to pass through the TOE. During the TCP session negotiation, packets with invalid flags are injected at various points in the exchange. The evaluator ensured by examining the TOE logs and viewing captured packets, that the packets with incorrect flags were discarded as not being part of the session. The evaluator also sent packets with incorrect source and destination addresses, incorrect source and destination ports, and incorrect flags and sequence numbers. The evaluator confirmed (through logs and packet captures) that all non-matching packets are not treated as part of the established session.

Test 2: The evaluator configured the TOE to allow and log network traffic for a valid TCP session. The evaluator started transmitting packets from a test server to establish a valid TCP session. The evaluator then terminated the TCP session, and attempted to send additional packets using the same TCP session information. The evaluator ensured by examining the TOE logs and viewing captured packets, that the packet sent after the session termination, was not passed by the TOE as part of the session.

Test 3: Repeated Test 2, expiring the session rather than explicitly terminating the session. The evaluator observed that no packets sent after a session expiration were treated by the TOE as part of the original TCP session.

Test 4: The evaluator configured the TOE to allow and log network traffic for a valid UDP session. The evaluator started transmitting packets from a test server to establish a valid UDP session. The packets being sent were constructed such that they would be permitted to pass through the TOE. The evaluator also sent packets with incorrect source and destination addresses, incorrect source and destination ports, and incorrect flags and sequence numbers. The evaluator confirmed (through logs and packet captures) that all nonmatching packets are not treated as part of the established session.

Test 5: The evaluator performed test 3 (using an expired session) using a UDP session rather than a TCP session.

Test 6: The evaluator configured the TOE to permit and log ICMP traffic and established a session for ICMP as defined in the TSS. Once an ICMP session was established, the evaluator altered each of the sessions determining attributes (source and destination addresses, other attributes chosen in FFW_RUL_EXT.1.5) one at a time and verified that the altered packets were not accepted as part of the established session.

Test 7: The evaluator terminated the ICMP session established per Test 6 as described in the TSS. The evaluator then immediately sent a packet matching the former session definition in order to ensure it is not forwarded through the TOE without being subject to the ruleset.

Test 8: The evaluator expired (i.e., reach timeout) the ICMP session established per Test 6 as described in the TSS. The evaluator then sent a packet matching the former session and verified it is not forwarded through the TOE without being subject to the ruleset.

2.4.1.6 STFFW14E:FFW_RUL_EXT.1.6

TSS Assurance Activities: The evaluator shall verify that the TSS identifies the following as packets that will be automatically dropped and are counted or logged:

a) Packets which are invalid fragments, including a description of what constitutes an invalid fragment

b) Fragments that cannot be completely re-assembled

c) Packets where the source address is defined as being on a broadcast network

d) Packets where the source address is defined as being on a multicast network

e) Packets where the source address is defined as being a loopback address

f) The TSF shall reject and be capable of logging network packets where the source or destination address of the network packet is defined as being unspecified (i.e. 0.0.0.0) or an address 'reserved for future use' (i.e. 240.0.0.0/4) as specified in RFC 5735 for IPv4;

g) The TSF shall reject and be capable of logging network packets where the source or destination address of the network packet is defined as an 'unspecified address' or an address 'reserved for future definition and use' (i.e. unicast addresses not in this address range: 2000::/3) as specified in RFC 3513 for IPv6;

h) Packets with the IP options: Loose Source Routing, Strict Source Routing, or Record Route specified

i) Other packets defined in FFW_RUL_EXT.1.6 (if any).

Section 6.4 explains that the TOE enforces the default stateful traffic filtering rules specified in FFW_RUL_EXT.1.6 and any administrator defined rules. These rules, called access-lists (ACLs), are in the sequence specified in the packet processing algorithm described in the assurance activities for FFW_RUL_EXT.1.5 above. Only a single access-list may be applied to an Ethernet interface. Multiple access-lists may be applied to a user-role. If multiple access-lists are applied, they are processed in order from top to bottom. The first match found is selected and no further processing takes place. If no match is found, the default action is to deny.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes packets that are discarded and potentially logged by default. If applicable protocols are identified, their descriptions need to be consistent with the TSS. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets.

Section “FFW_RUL_EXT.1.6” in the Admin Guide describes how to ensure proper handling of traffic with addresses identifies as ‘reserved for future use’ and also details the protocols blocked by default by the TOE. All other traffic is handled based upon access control lists configured on the TOE.

Section “FFW_RUL_EXT.1.7” in the Admin Guide describes how to configure the TOE with a default access control list to ensure that the packets identified by the FFW_RUL_EXT.1.6 and FFW_RUL_EXT.1.7 requirements are dropped and logged.

Testing Assurance Activities: Both IPv4 and IPv6 shall be tested for items a), b), c), d), and e) of the SFR element. Both IPv4 and IPv6 shall be tested for item i) unless the rule definition is specific to IPv4 or IPv6. Note: f), g), and h) are specific to IPv4 or IPv6 and shall be tested accordingly.

Test 1: The evaluator shall test each of the conditions for automatic packet rejection in turn. In each case, the TOE should be configured to allow all network traffic and the evaluator shall generate a packet or packet fragment that is to be rejected. The evaluator shall use packet captures to ensure that the unallowable packet or packet fragment is not passed through the TOE.

Test 2: For each of the cases above, the evaluator shall use any applicable guidance to enable dropped packet logging or counting. In each case above, the evaluator shall ensure that the rejected packet or packet fragment was recorded (either logged or an appropriate counter incremented).

Test 1: The evaluator sent a series of packets to the TOE, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator configured firewall rules to allow network traffic and enabled rejected packet logging. The evaluator generated the following types of packets which the TOE rejected and logged.

- Invalid Fragment IPv4 and IPv6
- Incomplete Fragment IPv4 and IPv6
- Invalid Broadcast Source Address IPv4 and IPv6
- Invalid Multicast Source Address IPv4 and IPv6
- Invalid Loopback Source Address IPv4 and IPv6
- Invalid Future Source and Destination Address
- Invalid Loose and Strict Source Routing and Record Route

In addition to the PP required default rules, the security target identifies several additional rules which were also tested.

1. The evaluator ensured that the TOE rejected and was capable of logging IPv4 packets with destination IP address equal to 0.0.0.0.

2. The evaluator ensured that the TOE rejected and was capable of logging IPv4 packets with source IP address equal to 0.0.0.0.

3. The evaluator ensured that the TOE rejected and was capable of logging packets with the first octet of the source IP address equal to zero.

4. The evaluator ensured that the TOE rejected and was capable of logging packets with the network part of the source IP address equal to all 0’s.

5. The evaluator ensured that the TOE rejected and was capable of logging packets with the network part of the source IP address equal to all 1’s.

6. The evaluator ensured that the TOE rejected and was capable of logging packets with the host part of the source IP address equal to all 1's.

7. The evaluator ensured that the TOE rejected and was capable of logging packets with the host part of the source IP address equal to all 0's.

8. The evaluator ensured that the TOE rejected and was capable of logging ICMP error packets when the ICMP error messages are not related to any session already established in the TOE.

9. The evaluator ensured that the TOE rejected and was capable of logging network packets when the appliance is not able to find any established connection related to the frame embedded in the ICMPv6 error message.

10. The evaluator ensured that the TOE rejected and was capable of logging network packets when an ICMP echo request/reply packet was received with a malformed code(non-zero).

Test 2: The evaluator observed that the TOE correctly blocked each of the packets directed to it and confirmed that the TOE correctly logged all rejected/dropped/blocked packets.

2.4.1.7 STFFW14E:FFW_RUL_EXT.1.7

TSS Assurance Activities: The evaluator shall verify that the TSS explains how the following traffic can be dropped and counted or logged:

a) Packets where the source address is equal to the address of the network interface where the network packet was received

b) Packets where the source or destination address of the network packet is a link-local address

c) Packets where the source address does not belong to the networks associated with the network interface where the network packet was received, including a description of how the TOE determines whether a source address belongs to a network associated with a given network interface

Section 6.4 in the ST explains the TOE is capable of dropping and logging network packets according to the rules specified in FFW_RUL_EXT.1.7. The administrator must configure a default Access Control List in order to ensure that this traffic is dropped and logged

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how the TOE can be configured to implement the required rules. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets.

Section “FFW_RUL_EXT.1.7” in the Admin Guide describes how to configure the TOE with a default access control list to ensure that the packets identified by the FFW_RUL_EXT.1.6 and FFW_RUL_EXT.1.7 requirements are dropped and logged.

Section “FFW_RUL_EXT.1.2/3/4” in the Admin Guide provides many examples of how to create rules with logging enabled.

Testing Assurance Activities: Test 1: The evaluator shall configure the TOE to drop and log network traffic where the source address of the packet matches that of the TOE network interface upon which the traffic was received. The evaluator shall generate suitable network traffic to match the configured rule and verify that the traffic is dropped and a log message generated.

Test 2: The evaluator shall configure the TOE to drop and log network traffic where the source IP address of the packet fails to match the network reachability information of the interface to which it is targeted, e.g. if the TOE believes that network 192.168.1.0/24 is reachable through interface 2, network traffic with a source address from the 192.168.1.0/24 network should be generated and sent to an interface other than interface 2. The evaluator shall verify that the network traffic is dropped and a log message generated.

The tests were performed using both IPv4 and IPv6.

Test 1: The evaluator configured the TOE to drop and log network traffic where the source address of the packet matches that of the TOE network interface upon which the traffic was received. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator confirmed via packet capture and logs that the traffic is dropped and a log message is generated.

Test 2: The evaluator configured the TOE to drop and log network traffic where the source IP address of the packet fails to match the network reachability information of the interface to which it is targeted. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator confirmed via packet capture and logs that the traffic is dropped and a log message is generated

2.4.1.8 STFFW14E:FFW_RUL_EXT.1.8

TSS Assurance Activities: The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset.

Section 6.4 in the ST explains the packet processing algorithm. The algorithm applied to incoming packets is as

follows:

Check for IP fragments and assemble.

Parse and identify protocol in the IP packet.

Perform length checks and apply default rules.

Enforce interface access-lists (ACLs) if configured.

Lookup session. If exists, then apply corresponding session / stateful ACLs.

Add session if it doesn't exist. (stateful if the protocol warrants, as listed above.). If stateful also open reverse ports for returning/stateful session. The protocol attributes identified above are used in session determination and will include IP protocol attributes for higher level protocols. TCP processing is further described below. Generate log message, if the 'log' keyword is configured in the rule.

Derive role for the user and apply role based ACLs. If no role ACLs, then apply default ACLs

(deny).

Perform bandwidth contract enforcement.

Perform NATing if required.

During TCP session establishment, the session is identified by source IP, destination IP, source port, and destination port. By default, the three-way handshake is not enforced before data is allowed. This behavior can be changed using the configuration "firewall enforce-tcp-handshake". Once the session has been established in the datapath session table, future packets which match the source IP, destination IP, source port, and destination port are processed by the "fast path" which was previously programmed during session establishment.

The TOE enforces the default stateful traffic filtering rules specified in FFW_RUL_EXT.1.6 and any administrator defined rules. These rules, called access-lists (ACLs), are in the sequence specified in the packet processing algorithm described in the algorithm applied to incoming packets above. Only a single access-list may be applied to an Ethernet interface. Multiple access-lists may be applied to a user-role. If multiple access-lists are applied, they are processed in order from top to bottom. The first match found is selected and no further processing takes place. If no match is found, the default action is to deny. If a rule is applied permitting or denying traffic, applying the inverse of this rule will overwrite the pre-existing rule. The TOE will not allow inverse rules to be applied.

The TOE is also capable of dropping and logging network packets according to the rules specified in FFW_RUL_EXT.1.7. The administrator must configure a default Access Control List in order to ensure that this traffic is dropped and logged.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how the order of stateful traffic filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing.

Section "FFW_RUL_EXT.1.2/3/4" of the Admin Guide states that rules should be configured in order from highest priority to lowest; enforcement is based on a first-match principle where the first rule that matches a traffic flow is applied, and further rules are not processed. The <position> field may be used to insert new rules somewhere other than at the end of a policy.

Section "FFW_RUL_EXT.1.8" of the Admin Guide states the TOE access control lists function in a hierarchical structure. If a rule is applied at the beginning of the ACL, it will take priority over any rule following it. For instance, if an access rule denies a specific IP address and a later rule permits a subnet that contains the specific host, the initial rule denying that specific IP will be applied and traffic from that IP address will be dropped. This section also notes that if a rule is applied permitting or denying traffic, applying the inverse of this rule will overwrite the preexisting rule. The TOE will not allow inverse rules to be applied.

Testing Assurance Activities: Test1: If the TOE implements a mechanism that ensures that no conflicting rules can be configured, the evaluator shall try to configure two conflicting rules and verify that the TOE rejects the conflicting rule(s). It is important to verify that the mechanism is implemented in the TOE but not in the non-TOE environment. If the TOE does not implement a mechanism that ensures that no conflicting rules can be configured, the evaluator shall devise two equal stateful traffic filtering rules with alternate operations - permit and drop. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation. (TD0545 applied)

Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

These tests were performed using both IPv4 and IPv6.

Test 1: The evaluator attempted to configure the TOE (according to the admin guide) with two firewall rules using the same matching criteria, where one rule would permit while the other deny traffic. The evaluator configured a permit rule first with the second rule (deny). The TOE with a rule to first permit traffic to a specific IPv4 destination. The evaluator confirmed that the connection was successful. Subsequently, the evaluator configured a deny rule first with the second rule (permit). The evaluator observed that the connection failed.

Test 2: Continuing test 1, the evaluator repeated the procedure above, except the evaluator changed the rules to make one a subset of the other, and then tested both orders. The evaluator confirmed that the first is enforced regardless of the specificity of the rule.

2.4.1.9 STFFW14e:FFW_RUL_EXT.1.9

TSS Assurance Activities: The evaluator shall verify that the TSS describes the process for applying stateful traffic filtering rules and also that the behavior (either by default, or as configured by the administrator) is to deny packets when there is no rule match unless another required conditions allows the network traffic (i.e., FFW_RUL_EXT.1.5 or FFW_RUL_EXT.2.1).

Section 6.4 in the ST explains that the TOE enforces the default stateful traffic filtering rules specified in FFW_RUL_EXT.1.6 and any administrator defined rules. These rules, called access-lists (ACLs), are in the sequence specified in the packet processing algorithm as described in the assurance activities in FFW_RUL_EXT.1.8 above. Only a single access-list may be applied to an Ethernet interface. Multiple access-lists may be applied to a user role. If multiple access-lists are applied, they are processed in order from top to bottom. The first match found is selected and no further processing takes place. If no match is found, the default action is to deny.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the guidance documentation provides the appropriate instructions to configure the behavior to deny packets with no matching rules.

Section “FFW_RUL_EXT.1.9” of the Admin Guide states that an access control list must be applied to all in-use interfaces in the evaluated configuration. If traffic is sent to an interface and no rule within the ACL applied to that interface matches the packet in-transit, the packet will be dropped. To ensure logging of traffic, the following rule should be applied as the last rule of an ACL to drop and log all unwanted traffic: (config-sess-ACL)#any any any deny log

All unsolicited messages are dropped by default.

Testing Assurance Activities: For each attribute in FFW_RUL_EXT.1.2, the evaluator shall construct a test to demonstrate that the TOE can correctly compare the attribute from the packet header to the ruleset, and shall demonstrate both the permit and deny for each case. It shall also be verified that a packet is dropped if no matching rule can be identified for the packet. The evaluator shall check the log in each case to confirm that the relevant rule was applied. The evaluator shall record a packet capture for each test to demonstrate the correct TOE behaviour.

The evaluator defined several tests variations to exercise the attributes and rules from FFW_RUL_EXT.1.2. The evaluator generated traffic to match specific aspects of the configured firewall rule set and confirmed that all attributes demonstrated permit, deny and log for each test case. Some examples of test variations exercised rules for ICMP, IPv4, IPv6, TCP, and UDP traffic demonstrating both permit and deny rules for these protocols.

2.4.1.10 STFFW14E:FFW_RUL_EXT.1.10

TSS Assurance Activities: The evaluator shall verify that the TSS describes how the TOE tracks and maintains information relating to the number of half-open TCP connections. The TSS should identify how the TOE behaves when the administratively defined limit is reached and should describe under what circumstances stale half-open connections are removed (e.g. after a timer expires).

Section 6.4 of the ST states that by default, TCP sequence numbers are ignored. This behavior can be changed using the configuration “firewall enforce-tcp-handshake” command which prevents data from passing between two clients until the three-way TCP handshake has been performed, at an upper limit of 8 half-open TCP connections. The “firewall enforce-tcp-sequence” command enforces the TCP sequence numbers for all packets. When applied, the TOE will monitor traffic sent through the TOE and drop half-open TCP connections

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes the behaviour of imposing TCP half-open connection limits and its default state if unconfigured. The evaluator shall verify that the guidance clearly indicates the conditions under which new connections will be dropped e.g. perdestination or per-client.

Section “FFW_RUL_EXT.1.10” in the Admin Guide indicates that by default, TCP sequence numbers are ignored. Commands are provided to ensure that TCP sequencing is properly enforced. The firewall enforce-tcp-handshake” command prevents data from passing between two clients until the three-way TCP handshake has been performed, at an upper limit of 8 half-open TCP connections. The “firewall" enforce-tcp-sequence” command enforces the TCP sequence numbers for all packets. When applied, the TOE will monitor traffic sent through the TOE and drop half-open TCP connections. The “show datapath frame” command can be used to monitor if traffic is dropped.

Testing Assurance Activities: Test 1: The evaluator shall define a TCP half-open connection limit on the TOE. The evaluator shall generate TCP SYN requests to pass through the TOE to the target system using a randomised source IP address and common destination IP address. The number of SYN requests should exceed the TCP half-open threshold defined on the TOE. TCP SYN-ACK messages should not be acknowledged. The evaluator shall verify through packet capture that once the defined TCP half-open threshold has been reached, subsequent TCP SYN packets are not transmitted to the target system. The evaluator shall verify that when the configured threshold is reached that, depending upon the selection, either a log entry is generated or a counter is incremented.

The evaluator configured a TCP half-open connection limit on the TOE. The evaluator then generated TCP SYN requests which would pass through the TOE to a target system using a randomized source IP address and common destination IP address. The number of SYN requests sent exceeded the TCP half-open threshold defined on the TOE. TCP SYN-ACK messages were not acknowledged. Using a packet capture, the evaluator verified that once the defined TCP half-open threshold had been reached, subsequent TCP SYN packets are not transmitted to the target system. During the test, the evaluator used commands found in Operation Guidance and found that the number of packets was being correctly counted.

Component TSS Assurance Activities: The evaluator shall verify that the TSS provides a description of the TOE's initialization/startup process, which clearly indicates where processing of network packets begins to take place, and provides a discussion that supports the assertion that packets cannot flow during this process.

The evaluator shall verify that the TSS also include a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describe the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets. The description shall also include a description how the TOE behaves in the situation where the traffic exceeds the amount of traffic the TOE can handle and how it is ensured that also in this condition stateful traffic filtering rules are still applied so that traffic does not pass that shouldn't pass according to the specified rules.

Section 6.4 in the ST explains that the TOE performs stateful packet filtering. Filtering rules may be applied to appliance Ethernet interfaces and to user-roles (wireless clients connecting through APs are placed into user roles). The TOE is comprised of a data plane and a control plane. Network packets are processed in the data plane, which is the first component that is initialized. Network interfaces are not brought 'up' until initialization is complete and the data plane operating system is fully initialized. All packet level enforcement is performed within the data plane. In case of system error, packets are dropped by default. The control plane operating system (management interfaces) boots simultaneously.

The TOE supports stateful processing of the following protocols:

RFC 792 (ICMPv4)

RFC 4443 (ICMPv6)

RFC 791 (IPv4)

RFC 2460 (IPv6)

RFC 793 (TCP)

RFC 768 (UDP)

The TOE allows the definition of a stateful packet filtering policy based on the following attributes that are used to

define rules (based on the actions: permit, deny or log) for the associated protocols:

Note: ICMPv4 Code 134 is an unsolicited router advertisement which is discarded/dropped by the TOE. All other traffic is handled based upon access control lists configured on the TOE.

ICMPv4

o Source Address

o Destination Address

o Type

o Code1

ICMPv6

o Source Address

o Destination Address

o Type

o Code

IPv4

o Source address

o Destination Address

o Transport Layer Protocol

IPv6

o Source address

o Destination Address

o Transport Layer Protocol

o IPv6 Extension header type

TCP

o Source Address

o Destination Address

o Source Port

o Destination Port

o Sequence number

o Flags

UDP

o Source Address

o Destination Address

o Source Port

o Destination Port

Component Guidance Assurance Activities: The guidance documentation associated with this requirement is assessed in the subsequent test evaluation activities.

The guidance documentation associated with this requirement is assessed in the subsequent test evaluation activities.

Component Testing Assurance Activities: Test 1: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would otherwise be denied by the ruleset should be sourced and be directed at a host. The evaluator shall verify using a packet sniffer that none of the generated network traffic is permitted through the firewall during initialization.

Test 2: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would be permitted by the ruleset should be sourced and be directed at a host. The evaluator shall verify using a packet sniffer that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test evaluation activities.

Test 1: The evaluator configured the TOE to block network traffic directed at a specific destination. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator rebooted the TOE and monitored traffic on both sides of the TOE during the reboot process. The evaluator confirmed via packet capture and logs that that while packets were being delivered to the TOE ingress side, no matching packets were being passed from the egress side.

Test 2: Using guidance, the evaluator configured the TOE to permit network traffic directed at a specific destination. The evaluator generated suitable traffic to match the configured rule, while capturing both the packets entering into the TOE (Test Network/IN pcap) as well as those exiting the TOE (Probe Network/OUT pcap). The evaluator rebooted the TOE and monitored traffic on both sides of the TOE during the reboot process. The evaluator observed that there is a gap during which no traffic is passed shortly after the reboot is started until just prior to the login prompt being presented by the TOE. This demonstrates that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

2.5 Identification and authentication (FIA)

2.5.1 802.1X Port Access Entity (Authenticator) Authentication (WLANAS10:FIA_8021X_EXT.1)

2.5.1.1 WLANAS10:FIA_8021X_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 WLANAS10:FIA_8021X_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.3 WLANAS10:FIA_8021X_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: In order to show that the TSF implements the 802.1X-2010 standard correctly, the evaluator will ensure that the TSS contains the following information:

The sections (clauses) of the standard that the TOE implements

- For each identified section, any options selected in the implementation allowed by the standards are specified

- For each identified section, any non-conformance is identified and described, including a justification for the non-conformance

Because the connection to the RADIUS server will be contained in an IPsec or RadSec (TLS) tunnel, the security mechanisms detailed in the RFCs identified in the requirement are not relied on to provide protection for these communications. Consequently, no extensive analysis of the RFCs is required. However, the evaluator will ensure that the TSS describes the measures (documentation, testing) that are taken by the product developer to ensure that the TOE conforms to the RFCs listed in this requirement.

Section 6.3 of the ST states that for 802.1X authentication, the Extensible Authentication Protocol (EAP) over LAN (EAPOL) is used for communication between the wireless client and the TOE. The TOE also establishes an IPsec tunnel with the RADIUS authentication server. In an 802.1X authentication exchange, the TOE functions as the Authenticator and strictly follows the port-based network control model as defined in section 7.1 of 802.1X-2010. When a wireless client connects to the TOE, the TOE passes authentication protocol messages between the client and the RADIUS authentication server, until the user is authenticated, or authentication is denied.

The TOE implements the Authenticator's state machine and counters as defined in section 8.9 & 8.10 of 802.1X2010 and strictly enforces the EAPOL PDU format as defined in section 11 of 802.1X-2010. The TOE's Authenticator implementation uses all 8021.X-2010 mandatory definitions. No optional or non-conformance definitions of 802.1X are implemented.

The TOE is certified by the Wi-Fi Alliance for conformance with 802.1X and EAP/RADIUS. Aruba also uses the following automated test tools:

Ixia IxANVL to test conformance with RFCs and 802.1X. Information about IxANVL can be found via: http://www.ixiacom.com/products/network_test/applications/ixanvl/.

VeriWave test tools (http://www.ixiacom.com/solutions/wifi-performance-test/) to test Wi-Fi performance. One feature of the VeriWave test suite is to exercise 802.1X capabilities of the product.

Additionally, Aruba conducts interoperability testing through custom-built automated test beds which contain numerous client operating systems (Windows XP, Windows Vista, Windows 7, Windows 8, Mac OS X, Linux, Apple iOS, Android, etc.) connecting to Aruba Wi-Fi access points.

The Aruba Quality Assurance (QA) team performs protocol compliance testing using standards based tools and interoperability testing using a range of external vendor equipment. Additionally, Aruba conducts interoperability testing through custom-built automated test beds which contain numerous client operating systems (Windows XP, Windows Vista, Windows 7, Windows 8, Mac OS X, Linux, Apple iOS, Android, etc.) connecting to Aruba Wi-Fi access points.

Component Guidance Assurance Activities: None Defined

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

Test 1: The evaluator will demonstrate that a wireless client has no access to the test network. After successfully authenticating with a RADIUS server through the TOE, the evaluator will demonstrate that the wireless client does have access to the test network.

Test 2: The evaluator will demonstrate that a wireless client has no access to the test network. The evaluator will attempt to authenticate using an invalid client certificate, such that the EAP-TLS negotiation fails. This should result in the wireless client still being unable to access the test network.

Test 3: The evaluator will demonstrate that a wireless client has no access to the test network. The evaluator will attempt to authenticate using an invalid RADIUS certificate, such that the EAP-TLS negotiation fails. This should result in the wireless client still being unable to access the test network.

Note: Tests 2 and 3 above are not tests that 'EAP-TLS works,' although that is a by-product of the test. The test is actually that a failed authentication (under two failure modes) results in denial of access to the network, which demonstrates the enforcement of FIA_8021X_EXT.1.3.

Test 1: The evaluator configured the TOE for wireless radius authentication. The evaluator then attempted to access the network prior to completing authentication and verified the attempt failed. The evaluator then attempted to access the network after completing authentication and verified the attempt succeeded.

Test 2: The evaluator attempted a connection with an invalid client certificate and verified that the connection was rejected and the client was unable to access the test network.

Test 3: The evaluator attempted a radius authentication connection with an invalid RADIUS certificate and verified that the connection was rejected and the client was unable to access the test network.

2.5.2 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.5.2.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.2.2 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 6.3 of the ST explains that the TOE maintains a counter of failed authentication attempts for a given administrative username within the past three minutes. An unsuccessful authentication attempt is detected when an invalid password or public key is entered for a valid username. If the failed authentication threshold is reached for a given username, that user account is locked out until the configured lock-out period has expired. The failed authentication threshold is enforced by the TOE and when using an external authentication server. In order to ensure that authentication failures by remote administrators cannot lead to a situation where no administrator access is available, the serial port does not enforce account locking. Therefore, local administrators accessing the TOE via the local console will always have access.

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 “FIA_AFL.1” of the Admin Guide states all configuration related to administrative login is configured using "aaa password-policy mgmt". It then provides an example to configure failed authentication lockout that will lock an administrative account for five minutes, when five failed login attempts occur in a three minute period. Note that if the remote authentication server locks out a user, the local account with the same name will not be marked as locked.

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).

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: The evaluator configured the TOE for a 3 failed authentication attempt and 1 minute lockout duration. The evaluator then attempted to connect to a session and verified after 3 failures the session was locked. After waiting the 1 minute lockout period the evaluator was able to reconnect. This was confirmed for both SSH and Web GUI.

Test 2: This was performed as part of test 1, where the evaluator waited until just before the configured timeout setting and attempted a login with valid credentials, and confirmed that this attempt was unsuccessful. The evaluator then waited until just after the configured time period and attempted a login with valid credentials and confirmed that the attempt was successful. This was confirmed for both SSH and Web GUI.

2.5.3 Password Management (NDcPP22e:FIA_PMG_EXT.1)

2.5.3.1 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 examine the TSS to determine that it contains the lists of the supported special character(s) and minimum and maximum number of characters supported for administrator passwords.

Section 6.5 of the ST describes that the TOE supports the following password characteristics.

Password is case-sensitive

A-Z, a-z, 0-9, !@#$%^&*()_+, and extended characters

Password minimum length is set to 8 and the maximum length is 128.

Passwords have maximum lifetime and new passwords must contain a minimum of 4 character changes from the previous password

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 "FIA_PMG_EXT.1" of the Admin Guide identifies the special characters that can be used in addition to numbers and letter in passwords. It also explains how to set the minimum password length and password strength characteristics so that strong passwords are required.

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 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.

The evaluator configured passwords to meet all of the rules in the test procedures and verified that passwords that do not meet the requirements were rejected.

2.5.4 Pre-Shared Key Composition (VPNGW12:FIA_PSK_EXT.1)

2.5.4.1 VPNGW12:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.4.2 VPNGW12: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 that it identifies all protocols that allow pre-shared keys. For each protocol identified by the requirement, the evaluator shall confirm that the TSS states which pre-shared key selections are supported.

Section 6.5 of the ST states the TOE accepts between 6-160 character text based pre-shared keys (composed of any combination of upper and lower case letters, numbers, and special characters (that include: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')')) for IPsec. It accepts bit-based pre-shared keys and accepts text-based PSKs that are transformed into bit-based PSKs. Text-based keys are conditioned by being directly converted to binary.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on how to configure all selected pre-shared key options if any configuration is required.

Section "FIA_PSK_EXT.1 (VPNGW)" of the Admin Guide explains that ArubaOS supports IKE pre-shared keys, when certificates are not used for authentication. In order to use pre-shared keys, an IKE policy that uses pre-shared keys must be created. The commands with examples for creating text-based and bit-based pre-shared keys are provided in this section. It also explains that when configuring the pre-shared key, the administrator must ensure that the PSK is at least 22 characters, contains at least one uppercase character, one lowercase character, one special character, and one digit. It specifies the allowable characters and the list is equal to those claimed in the requirement.

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).

Test 1: For each mechanism selected in FIA_PSK_EXT.1.2 the evaluator shall attempt to establish a connection and confirm that the connection requires the selected factors in the PSK to establish the connection.

This test was performed with both bit-based and password-based Pre-shared keys.

The evaluator configured the TOE and a test server to use pre-shared keys for authentication during IPsec IKEv2 negotiations. The IPsec IKEv2 connection was successful for both types of Pre-shared key.

2.5.5 Pre-Shared Key Composition (WLANAS10:FIA_PSK_EXT.1)

2.5.5.1 WLANAS10:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.2 WLANAS10:FIA_PSK_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.5.3 WLANAS10:FIA_PSK_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator will verify that the TSS describes

1. the protocols that can use pre-shared keys and that these are consistent with the selections made in FIA_PSK_EXT.1.1.

2. the allowable values for pre-shared keys and that they are consistent with the selections made in FIA_PSK_EXT.1.2.

3. the way bit-based pre-shared keys are procured and that it is consistent with the selections made in FIA_PSK_EXT.1.3.

Section 6.3 of the ST states that the TOE accepts between 6-160 character text based pre-shared keys (composed of any combination of upper and lower case letters, numbers, and special characters (that include: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')')) for IPsec (IKEv1, IKEv2) WPA-3 (WPA3-SAE) and WPA2 (WPA-PSK). It accepts bit-based pre-shared keys and accepts text-based PSKs that are transformed into bit-based PSKs. For WPA3/WPA2, text-based keys are conditioned using PBKDF2, as specified in 802.11i. For IPsec, text-based keys are conditioned by being directly converted to binary. This is consistent with the FIA_PSK_EXT.1.3 requirement.

Component Guidance Assurance Activities: The evaluator will examine the operational guidance to determine that it provides guidance to administrators on the composition of strong text-based pre-shared keys, and (if the selection indicates keys of various lengths can be entered) that it provides information on the range of lengths supported. The guidance must specify the allowable characters for pre-shared keys, and that list must be a superset of the list contained in FIA_PSK_EXT.1.2.

The evaluator will confirm the operational guidance contains instructions for either entering bit-based preshared keys for each protocol identified in the requirement or for generating a bit-based pre-shared key (or both).

Section “FIA_PSK_EXT.1 (WLAN)” of the Admin Guide provides the commands for configuring the pre-shared key for 802.1X. When configuring the pre-shared key, the administrator must ensure that the PSK is at least 22 characters, contains at least one uppercase character, one lowercase character, one special character, and one digit. Allowable special characters are: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')'

Additionally, this section references section "FIA_PMG_EXT.1" which contains guidelines for composing strong text-based pre-shared keys.

Component Testing Assurance Activities: The evaluator will 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 1: The evaluator will compose a pre-shared key of 22 characters that contains a combination of the allowed characters in accordance with the operational guidance and demonstrates that a successful protocol negotiation can be performed with the key.

Test 2: [conditional]: If the TOE supports pre-shared keys of multiple lengths, the evaluator will repeat Test 1 using 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 3: [conditional]: If the TOE does not generate bit-based pre-shared keys, the evaluator will obtain a bit-based pre-shared key of the appropriate length and enter it according to the instructions in the operational guidance. The evaluator will then demonstrate that a successful protocol negotiation can be performed with the key.

Test 4: [conditional]: If the TOE does generate bit-based pre-shared keys, the evaluator will generate a bit-based pre-shared key of the appropriate length and use it according to the instructions in the operational guidance. The evaluator will then demonstrate that a successful protocol negotiation can be performed with the key.

Test 1 - the evaluator configured a 22 character pre-shared key on the TOE and performed a successful protocol connection using this key. This test was repeated for WPA3-SAE, WPA2-PSK, WPA3 Enterprise, and WPA2 Enterprise.

Test 2 - The evaluator attempted to configure several different pre-shared keys and ensured that the various conditions are met with the expected outcome.

Test 3 - The evaluator configured the TOE to accept a 256 bit hex key and then performed a successful protocol connection using the key.

2.5.6 Generated Pre-Shared Keys (VPNGW12:FIA_PSK_EXT.2)

2.5.6.1 VPNGW12:FIA_PSK_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If 'generate' is selected, the evaluator shall confirm that this process uses the RBG specified in FCS_RBG_EXT.1 and the output matches the size selected in FIA_PSK_EXT.2.1.

The TOE does not generate pre-shared keys.

Component Guidance Assurance Activities: The evaluator shall confirm the operational guidance contains instructions for entering generated pre-shared keys for each protocol identified in the FIA_PSK_EXT.1.1.

Section "FIA_PSK_EXT.1 (VPNGW)" of the Admin Guide contains instructions for entering text-based and bit-based pre-shared keys for the IPsec protocol.

Component Testing Assurance Activities: Test 1: [conditional] If generate was selected the evaluator shall generate a pre-shared key and confirm the output matches the size selected in FIA_PSK_EXT.2.1.

This test is not applicable as the TOE does not generate pre-shared keys.

2.5.7 Password-Based Pre-Shared Keys (VPNGW12:FIA_PSK_EXT.3)

2.5.7.1 VPNGW12:FIA_PSK_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.2 VPNGW12:FIA_PSK_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.3 VPNGW12:FIA_PSK_EXT.3.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.4 VPNGW12:FIA_PSK_EXT.3.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.5 VPNGW12:FIA_PSK_EXT.3.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.6 VPNGW12:FIA_PSK_EXT.3.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.7.7 VPNGW12:FIA_PSK_EXT.3.7

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 used.

Support for length: The evaluator shall check to ensure that the TSS describes the allowable ranges for PSK lengths, and that at least 64 characters or a length defined by the platform may be specified by the user. Support for character set: The evaluator shall check to ensure that the TSS describes the allowable character set and that it contains the characters listed in the SFR.

Support for PBKDF: The evaluator shall examine the TSS to ensure that the use of PBKDF2 is described and that the key sizes match that described by the ST author.

The evaluator shall check that the TSS describes the method by which the PSK is first encoded and then fed to the hash algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself.

For the NIST SP 800-132-based conditioning of the PSK, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1/KeyedHash). The evaluator shall confirm that the minimum length is described.

The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1. [conditional] If password strength meter or password denylist is selected, the evaluator shall examine the TSS to ensure any password checking functionality provided by the TSF is described and contains details on how the function operates.

Section 6.5 of the ST states that the TOE accepts between 6 (minimum) and 160 (maximum) character text based pre-shared keys.

Section 6.5 of the ST states that the TOE accepts pre-shared keys for IPsec (IKEv1, IKEv2) WPA-3 (WPA3-SAE) and WPA2 (WPA-PSK). It accepts bit-based pre-shared keys and accepts text-based PSKs that are transformed into bit-based PSKs. For WPA3/WPA2, text-based keys are conditioned using PBKDF2, as specified in 802.11i. For IPsec, text-based keys are conditioned by being directly converted to binary

Component Guidance Assurance Activities: The evaluator shall confirm the operational guidance contains instructions for entering bit-based pre-shared keys for each protocol identified in the requirement, or generating a bit-based pre-shared key (or both). The evaluator shall confirm that any management functions related to pre-shared keys that are performed by the TOE are specified in the operational guidance.

The guidance must specify the allowable characters for pre-shared keys, and that list must include, at minimum, the same items contained in FIA_PSK_EXT.3.2. The evaluator shall confirm the operational guidance contains any necessary instructions for enabling and configuring password checking functionality.

Section "FIA_PSK_EXT.1/2/3 (VPNGW)" of the Admin Guide contains instructions for entering bit-based pre-shared keys for the IPsec protocol.

The section also specifies the set of allowable characters characters for pre-shared keys. It also states that password checking is enabled by default and no configruation is required.

Component Testing Assurance Activities: Support for Password/Passphrase characteristics: In addition to the analysis above, the evaluator shall also perform the following tests on a TOE configured according to the Operational Guidance:

Test 1: The evaluator shall compose a pre-shared key of at least 64 characters that contains a combination of the allowed characters in accordance with the FIA_PSK_EXT.1.3 and verify that a successful protocol negotiation can be performed with the key.

Test 2: [conditional]: If the TOE supports pre-shared keys of multiple lengths, the evaluator shall repeat Test 1 using the minimum length and invalid lengths that are below the minimum length, above the maximum length, null length, empty length, or zero length. The minimum test should be successful, and the invalid lengths must be rejected by the TOE.

Test 3: [conditional]: If the TOE initiates connections, initiate and establish a remote connection, disconnect from the connection, verify that the PSK is required when initiating the connection a second time.

Test 4: [conditional]: If the TOE supports a password meter, the evaluator shall enter a password and verify the password checker responds per the description in the TSS.

Test 5: [conditional]: If the TOE supports a password denylist, the evaluator shall enter a denylisted password and verify that the password is rejected or flagged as such.

Test 1: The evaluator composed a 64 character pre-shared key that contains a combination of allowed characters. The evaluator then attempted an IPsec IKEv2 connection using this pre-shared key and confirmed that the protocol negotiation was successful.

Test 2: The evaluator attempted to set the pre-shared key on the TOE to lengths that were exactly at the minimum, below the minimum, above the maximum, and zero length. The TOE only accepted the pre-shared key that was exactly at the minimum length.

Test 3: The evaluator caused the TOE to initiate an IPsec IKEv2 connection to a server using a pre-shared key. After the connection was successful, the evaluator caused the TOE to disconnect from the connection. The evaluator then changed the PSK on the server (but left the original PSK on the TOE) and caused the TOE to attempt the connection again. This subsequent attempt was unsuccessful.

Test 4: is not applicable as the TOE does not support a password meter.

Test 5: is not applicable as the TOE does not support a password denylist.

2.5.8 yeRe-Authenticating (WLANAS10:FIA_UAU.6)

2.5.8.1 WLANAS10:FIA_UAU.6.1

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: The evaluator will attempt to change their password as directed by the operational guidance. While making this attempt, the evaluator will verify that re-authentication is required. If other re-authentication conditions are specified, the evaluator will cause those conditions to occur and verify that the TSF re-authenticates the authenticated user.

Test 1: The evaluator followed operational guidance to change the administrator password, then verified that the session immediately required re-authentication.

2.5.9 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.5.9.1 NDcPP22e:FIA_UAU.7.1

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: 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.

Section "FIA_UAU.7" of the Admin Guide states that no configuration is necessary to ensure that authentication data is not revealed.

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: The evaluator confirmed the password was not echoed back while being typed in.

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

2.5.10.1 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 NDcPP22e_FIA_UIA_EXT.1

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.

Section “FIA_UAU_EXT.2” in the Admin Guide provides instructions for configuring administrative users and setting the password using the ‘mgmt-user’ command.

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 NDcPP22e_FIA_UIA_EXT.1

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

2.5.11.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.11.2 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.

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 6.5 in the ST explains that the TOE supports role-based authentication. Users can authenticate to an external authentication server or to the Controller's internal database. Prior to requiring the non TOE entity to initiate the identification and authentication process, the TOE displays an administrator configured advisory notice and consent warning message regarding unauthorized use of the TOE. The TOE requires an administrator to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that user.

The administrator can create a user account in the internal database and assign a predefined role to that account. Users logging in to the Controller are restricted based on their assigned role. In this case, the authentication mechanism is provided by the TOE and the credentials are maintained in the internal database. The administrator can also configure the TOE so that users are authenticated using an external authentication server. The TOE supports RADIUS and TACACS+ servers. A trusted channel using IPsec is established between the TOE and an external authentication server.

Remote administrators are configured as users who have privileges to access the CLI and Web GUI administration interfaces and who are authenticated as users using the local database or an external authentication server. The remote administrators authenticate as users using a username and password via Web GUI and username/password or public key for SSH. The Web GUI interface provides a trusted path to connect to the TOE via HTTPS. The HTTPS interface uses a server RSA or ECDSA certificate which is stored on the TOE. SSH provides a trusted path to connect to the CLI. It uses SSH_RSA for public keys. Direct console to the CLI only supports username/password. A successful logon takes place when a recognized username/password combination is provided and/or a recognized X.509 client certificate is presented by the administrator's web browser or SSH client.

For users connecting with a VPN client, the IPsec/IKE VPN is established between the TOE and the client prior to the user authentication using pre-shared keys or certificates and can optionally authenticate to the external authentication server using a username and password. The external authentication server communicates success or failure of the authentication to the TOE.

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 “FIA_UIA_EXT.1” of the Admin Guide states that the TOE permits authentication by an administrator through SSH or Web UI over TLS or via a serial console (direct) connection to the CLI. Authentication is permitted through username/password and public key authentication via local authentication or by a remote authentication server (RADIUS/TACACS+). No user can perform any actions prior to successful authentication to the TOE outside of viewing the warning banner. Instructions for configuring the warning banner are provided.

The Admin Guide primarily provides guidance instructions using command-line interface (CLI) commands. The “Introduction “section in the Admin Guide provides a web link to the ArubaOS User Guide for Web UI instructions if needed. The User Guide states that the Web UI is accessible through a standard web browser from a remote management console or workstation.

Section “FIA_UAU_EXT.2” in the Admin Guide provides instructions for configuring administrative users and setting the password using the ‘mgmt-user’ command.

Section “FIA_PMG_EXT.1” in the Admin Guide provides guidance for configuring a password policy.

Section “FCS_SSHS_EXT.1” in the Admin Guide provides instructions for configuring authentication for SSH using public key (SSH-RSA).

Section “FCS_TLSS_EXT.1” in the Admin Guide provides instructions for configuring the TLS web-server profile.

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.

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

- SSH using passwords

- SSH using public/private key pairs

- SSH Connection using Radius Account

- SSH Connection using TACACS+ Account

- HTTPS (Web UI) using passwords

Test 1 - Using each interface the evaluator performed an unsuccessful and successful logon of each type using bad and good credentials respectively.

Test 2 - Using each interface the evaluator was able to observe the TOE displayed a banner to the user before login.

Test 3 - Using each interface the evaluator found that, prior to login, no functions were available to the administrator with the exception of acknowledging the banner.

Test 4 - The TOE is not distributed, thus tests 1 through 3 above test the only TOE component.

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

2.5.12.1 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 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.

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 and a peer with valid certificates. The evaluator then attempted to make a connection between the peer devices. A successful connection was made. The evaluator then configured a server certificate with an invalid certification path by deleting an intermediate root CA so that the certificate chain was invalid because of a missing (or deleted) certificate. The connection between the peers was refused.

Test 2 -- The evaluator attempted to make a connection between the TOE and a test server. The test server then presented an expired certificate during the negotiation resulting in a failed connection.

Test 3 -- The evaluator attempted to make a connection between the TOE and a test server. The test server then presented a certificate during the TLS negotiation where the certificate was valid. A packet capture was obtained of this TLS negotiation which shows that the connection was successful. The evaluator revoked certificates in the chain (individually) and attempted the same connection. The attempt after revoking the certificate was not successful.

Test 4 -- The evaluator configured an OCSP responder to present a certificate that does not have the OCSP signing purpose. The evaluator established a connection between the TOE and a test server such that the TOE receives OCSP response signed by the invalid certificate and ensured that the connection was not negotiated successfully.

Test 5 -- The evaluator configured a test server to present a certificate that had a byte in the first eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect, the connection failed.

Test 6 -- The evaluator configured a test server to present a certificate that had a byte in the last eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect, the connection failed.

Test 7 -- The evaluator configured a test server to present a certificate that had a byte in the public key of the certificate modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect, the connection was refused.

Test 8a - The evaluator configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where the ECDSA certificates have a supported named curve. The TOE accepted the certificate as valid.

Test 8b - The evaluator then configured the test peer to send a valid chain of ECDSA certificates where only the root CA is located in the trust store, and where an intermediate CA's ECDSA certificate has an explicit curve. The TOE rejected the certificate as invalid.

Test 8c - The evaluator attempted to load an intermediate CA ECDSA certificate with a supported named curve into the TOE trust store and observed that the certificate can be loaded. The evaluator attempted to load an intermediate CA ECDSA certificate with an explicit curve into the TOE trust store and observed that the certificate could not be loaded.

2.5.12.2 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 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: The evaluator configured a test server to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then attempted a connection between the TOE and the test server and observed that the TOE rejected the connection.

Test 2: The evaluator configured a test server to present a certificate chain containing a CA certificate having the basicConstraints section but with the cA flag not set (i.e., FALSE). The evaluator then attempted a connection between the TOE and the test server and observed that the TOE rejected the connection.

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 6.5 in the ST provides the certificate evaluation algorithm. Certificates are validated as part of the authentication process when they are presented to the TOE and when they are loaded into the TOE. The TOE validates certificates in accordance with RFC 5280 certificate validation rules. The TOE validates the revocation status of certificates using OCSP as specified in RFC 6960, Section 5. OCSP responder servers must be configured in the revocation profile. The profile consists of the server URL, the responder cert (used to verify OCSP responses), and the action to take if the OCSP responder is non-responsive (permit or deny). If revocation checking is enabled, all certs in the chain except for the root are verified in order, starting with the peer cert and ending at the penultimate CA certificate. Certificate validation includes verification of the basicConstraints extension and the CA flag to determine whether they are present and set to TRUE. The check also ensure that the key usage extension is present. OCSP certificates must have the OCSP signing purpose in the extendedKeyUsagefield.

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 "FIA_X509_EXT.1/2" of the Admin Guide states that the TOE uses OCSP for revocation checking. All certificates in the configured chain are checked for revocation status. This section also states that ArubaOS validates the extendedKeyUsage field in accordance with client certificates presented for TLS and OCSP certificated presented for OCSP responses. ArubaOS does not claim FCS_TLSC_EXT.x or X.509 certificates for trusted updates, making no claim to the rules specified under FIA_X509_EXT.1.1/Rev. These claims are trivially satisfied.

Component Testing Assurance Activities: None Defined

2.5.13 X.509 Certificate Validation (VPNGW12:FIA_X509_EXT.1/Rev)

2.5.13.1 VPNGW12:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

See NDcPP22e: FIA_X509_EXT.1/Rev

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.5.14.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.14.2 NDcPP22e:FIA_X509_EXT.2.2

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 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 6.5 of the ST explains that the TOE protects, stores and allows authorized administrators to load X.509v3 certificates for use to support authentication for IPsec connections. Certificates are loaded into the controller using the “Certificate Manager” section of the Web-based user interface. The controller supports loading of certificates in PEM, DER, or PFX format. Private keys may be loaded onto the controller through a password-protected PFX file or may originate on the controller at the time a Certificate Signing Request (CSR) is generated.

During runtime, certificates and private keys are stored in ramdisk, in volatile memory, in decrypted form. This allows private keys to be accessed rapidly for high network load conditions. When powered off, private keys are stored encrypted in non-volatile (flash) memory using AES256. If the TOE cannot determine the validity of a certificate, the administrator has the option of choosing whether or not to accept the certificate.

Section “FIA_X509_EXT.1” of the Admin Guide provides a detailed description of certificate checking. It also explains how the administrator configures the behavior if the revocation server cannot be reached. To configure the behavior in the event an OCSP responder cannot be reached, the "server-unreachable" keyword under the RCP configuration should be used.

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 "FIA_X509_EXT.1/2" of the Admin Guide describes the configuration required to configure the operating environment and TOE to use certificates for authentication.

This section also describes the configuration steps to set the behavior of the TOE when the connection fails for a validity check of a certificate.

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.

The evaluator first performed a control test to verify that a successful connection was made using an authentication certificate with valid/accessible revocation servers sent from a test peer to the TOE. The evaluator then configured the TOE to allow a connection even when a revocation server is unreachable and verified that the connection succeeded. Finally, the evaluator configured the TOE to reject a connection when a revocation server is unreachable and verified that the connection failed.

2.5.15 X.509 Certificate Authentication (VPNGW12:FIA_X509_EXT.2)

2.5.15.1 VPNGW12:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.15.2 VPNGW12:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to support its use for IPsec at a minimum. The evaluator shall ensure that all evaluation of this SFR is performed against its use in IPsec communications as well as any other supported usage.

See NDcPP22e: FIA_X509_EXT.2

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5.16 X.509 Certificate Requests (NDcPP22e:FIA_X509_EXT.3)

2.5.16.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.16.2 NDcPP22e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If the ST author selects 'device-specific information', the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

Device specific information is not selected in the requirement.

Component Guidance Assurance Activities: The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certification Request. If the ST author selects 'Common Name', 'Organization', 'Organizational Unit', or 'Country', the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the Certification Request.

Section "FIA_X509_EXT.3" of the Admin Guide provides the commands and an example for requesting and exporting a CSR. It states that before creating a CSR, the administrator must ensure that the CN, country, O, and OU have been set. The example includes how to set each of these fields.

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

a) Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a Certification Request. The evaluator shall capture the generated request and ensure that it conforms to the format specified. The evaluator shall confirm that the Certification Request provides the public key and other required information, including any necessary user-input information.

b) Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the response message, and demonstrate that the function succeeds.

Test 1- The evaluator followed operational guidance to log into the TOE and then generate the CSR. The evaluator then used the openssl command to ensure that the CSR contains the information claimed in the ST.

Test 2 - The evaluator tested that a certificate without an intermediate CA cannot be validated. The evaluator then demonstrated that a valid certificate signed by a CA that the TOE recognized can be successfully added.

2.6 Security management (FMT)

2.6.1 Management of Security Functions Behaviour (NDcPP22e:FMT_MOF.1/Functions)

2.6.1.1 NDcPP22e:FMT_MOF.1.1/Functions

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 for each administrative function identified the TSS details how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE).

See NDcPP22e: FMT_SMF.1 for a description of administrative functions provided by the TSS.

See NDcPP22e: FAU_GEN.1 AND FAU_STG_EXT.1 for a description of logging behavior.

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 describes how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE) are performed to include required configuration settings.

Section FAU_STG_EXT.1 of the Admin Guide states that the local protected log storage operates using the first in, first out (FIFO) method, therefore audit logs are overwritten when the available space is exhausted. This section also contains configuration instructions for protecting the syslog server communication with IPsec.

Section FAU_GEN.1 of the Admin Guide contains instructions to configure the TOE to send audits to an external syslog server.

Component Testing Assurance Activities: Test 1 (if 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters 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 modify the security related parameters 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.

Test 2 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as Security Administrator. The effects of the modifications should be confirmed.

The evaluator does not have to test all possible values of the security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity but at least one allowed value per parameter.

Test 1 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters 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 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 term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

Test 2 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data with prior authentication as Security Administrator. The effects of the modifications should be confirmed. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

The evaluator does not necessarily have to test all possible values of the security related parameters for configuration of the handling of audit data but at least one allowed value per parameter.

Test 1 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This attempt 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 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.

Test 2 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full with prior authentication as Security Administrator. This attempt should be successful. The effect of the change shall be verified.

Test 3 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions without administrator authentication shall 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 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.

Test 4 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection): The evaluator shall try to determine the behaviour of all options chosen from the second selection with prior authentication as Security Administrator. This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions with Security Administrator authentication shall be successful.

Test 1: The evaluator attempted commands prior to login and confirmed that the configuration could not be modified prior to authentication.

Test 2: This was tested as part of NDcPP22e:FAU_STG_EXT.1. The evaluator modified parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity successfully as an authenticated administrator.

Test 3: This test is not applicable because the selection 'determine the behavior of' was not made in this Security Target.

Test 4: This test is not applicable because the selection 'determine the behavior of' was not made in this Security Target.

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

2.6.2.1 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.

Not applicable, as 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).

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 “FPT_TUD_EXT.1” of the Admin Guide describes the update process. The “copy” command is used to download new firmware images from an FTP or TFTP server. From the WebUI, the administrator should navigate to Maintenance>Software Management>Upgrade page to upload an ArubaOS image from a local filesystem. An option is then provided in either case to reboot the device with the new image file.

ArubaOS images are integrity-protected through three methods:

1. When downloading a firmware image from http://support.arubanetworks.com, a file may be found in the download directory that contains SHA256 hashes of each file. This hash may be checked manually after downloading an image.

2. ArubaOS images are digitally signed using RSA 2048 bit signature validation. The mobility controller will check the digital signature immediately after downloading a new firmware image, and will refuse to install an image whose digital signature does not match.

3. Mobility controllers also check the digital signature of an ArubaOS image when booting. The controller will refuse to boot a corrupted ArubaOS image file.

The Admin Guide does not identify any functions that may cease to operate during the update. If digital signature verification fails, the TOE will enter into an error state. The TOE’s error state will allow direct console access only, where an administrator can change to a new file partition or TFTP a new image and re-boot.

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 evaluator attempted commands to perform an update prior to login and confirmed that the configuration could not be viewed or modified prior to authentication

2.6.3 Management of Security Functions Behaviour (NDcPP22e:FMT_MOF.1/Services)

2.6.3.1 NDcPP22e:FMT_MOF.1.1/Services

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 services the Security Administrator is able to start and stop and how that how that operation is performed.

Section 6.6 of the ST lists the services that the Security Administrator is able to start and stop. It states that the services can be enabled or disabled through policy configuration.

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 describes how the TSS lists the services the Security Administrator is able to start and stop and how that how that operation is performed.

Section FMT_MOF.1/Services of the Admin Guide states that:

An administrator can start and stop remote syslog services, certificate validation checks, IPsec, TLS, SSH, NTP polling, authentication services, 802.11x, and certificate validation. These services can be enabled or disabled by following the guidance found throughout this document as well as in the ArubaOS 8.10 User Guide. Starting and stopping these services can be performed through policy configuration.

Component Testing Assurance Activities: The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) without prior authentication as Security Administrator (either by authenticating as a user with no administrator privileges, if possible, or without prior authentication at all). The attempt to enable/disable this service/these services 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 enable/disable this service/these services 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 enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) with prior authentication as security administrator. The attempt to enable/disable this service/these services should be successful.

The evaluator attempted to set and disable logging with an unauthorized account and verified that the account did not have permission to execute the command. Next, the evaluator attempted to set and disable logging as authorized administrator and verified that the attempt to enable/disable this service was successful.

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

2.6.4.1 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.

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 6.6 in the ST states that only Security Administrators can manage TSF data. There are no security functions available through any interfaces prior to administrator login. Non-administrative users do not have access to the TOE via the Web UI or the CLI, therefore, they do not have any access to the security functions of the TOE.

Section 6.5 in the ST states that the TOE protects, stores and allows authorized administrators to load X.509v3 certificates for use to support authentication for IPsec connections. Certificates are loaded into the controller using the “Certificate Manager” section of the Web-based user interface. The controller supports loading of certificates in PEM, DER, or PFX format. Private keys may be loaded onto the controller through a password-protected PFX file or may originate on the controller at the time a Certificate Signing Request (CSR) is generated. During runtime, certificates and private keys are stored in ramdisk, in volatile memory, in decrypted form. This allows private keys to be accessed rapidly for high network load conditions. When powered off, private keys are stored encrypted in non-volatile (flash) memory. The encryption method used is AES256 as described in the CSP table.

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 only users permitted to log onto the TOE are administrators so all such access is limited to administrators. The Admin Guide has sections for each area in the cPP including audit, firewall, cryptography, packet filtering, identification & authentication, TOE access and Trusted path. In each section are the commands to configure the associated function.

Section “FMT_MTD.1/CoreData” in the Admin Guide states that an administrator with the management role of “root” has full privileges to modify, add, and delete configuration and user accounts. The “root” role maps to the Security Administrator role.

Section “FIA_X509_EXT.3” in the Admin Guide provides instructions for generating a certificate sign request and exporting it. Section “FIA_X509_EXT.1” in the Admin Guide states that best practice is to generate the CSR on the controller, then load the resulting certificate after issuance by the CA. This protects the private key from

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 of the management functions have been exercised under the other SFRs as demonstrated throughout the AAR.

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

2.6.5.1 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 6.2 of the ST contains Table 12: CSPs. This lists all CSPs, as well as their generation/use, storage, and associated zeroization process.

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 FMT_MTD.1/CryptoKeys of the Admin Guide states that an administrator can manage all X.509 certificates, server certificates, and SSH public keys as defined within the ArubaOS 8.10 User Guidance and the specified sections of this document. Additionally, an administrator seeking to regenerate all persistent device keys can perform a ‘wipe out flash’ to zeroize keys. The TOE will regenerate these keys upon restart.

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 results of NDcPP22e:FIA_UIA_EXT.1-t3 demonstrate there are no functions available to users prior to login aside from viewing the login banner.

Testing in NDcPP22e:FIA_X509_EXT.1.1/Rev-t1 indicates that an authenticated Security Administrator can load a trusted root into the TOE as well as show that the trusted root can be removed from the TOE.

2.6.6 Management of TSF Data (VPNGW12:FMT_MTD.1/CryptoKeys)

2.6.6.1 VPNGW12:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to make it mandatory because of the TOE's required support for IPsec.

The assurance activities related specifically to keys and certificates were performed as part of NDcPP22e: FMT_SMF.1.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.6.7.1 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).

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.

Section 6.6 in the ST identifies all management functions consistent with those specified in FMT_SMF.1. The administrator can configure TOE security settings and policies using the Web UI via HTTPS, or the command line interface via serial console locally or remotely using SSHv2. The Web GUI is a just a front-end to the CLI (i.e., calls the CLI internally). It provides a user-friendly interface for the administrator to manage the TOE. Every function that can be performed on the Web GUI, can also be performed using the CLI but not vice versa. However, every security management function claimed can be done either using the Web GUI or CLI.

Section “FIA_UIA_EXT.1” in the Admin Guide describes the local and remote administration interfaces and this is documented in FIA_UIA_EXT.1 of this report. Section “FIA_UIA_EXT.1” in the Admin Guide clarifies that the local administrative interface is a serial console (direct) connection to the CLI which is enough information for the administrator to ensure that an interface is local.

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. The TOE is compliant with all requirements in the ST as identified in this report. The evaluator used the Guidance as part of testing and was able to configure all claimed functionality.

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.

The TOE is compliant with all requirements in the ST as identified in this report. The evaluator used the Guidance as part of testing and was able to configure all claimed functionality in order to perform the associated tests as identified in the test assurance activities.

2.6.8 Specification of Management Functions (WLAN Access Systems) (WLANAS10:FMT_SMF.1/AccessSystem)

2.6.8.1 WLANAS10:FMT_SMF.1.1/AccessSystem

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator will confirm that the TSS includes which security types (e.g., WPA3), authentication protocol (e.g., SAE), and frequency bands the WLAN AS supports. The evaluator will confirm that the TSS includes how connection attempts from clients that are not operating on an approved security type are handled.

Section 6.10 of the ST lists the security types and authentication protocols supported by the TOE. These are WPA3-Enterprise, WPA2-Enterprise, WPA3-SAE, and WPA2-PSK.

Section 6.6 of the ST lists the frequency bands that the TOE supports. These are 2.4ghz, 5ghz, and 6ghz.

Section 6.6 of the ST states that connection attempts from a client using a non-configured/approved security type will be rejected by the TOE.

Component Guidance Assurance Activities: The evaluator will confirm that the operational guidance includes instructions for configuring the WLAN AS for each feature listed.

Section FMT_SMF.1/AccessSystem of the Admin Guide provides a reference to the User Guide for instructions to configure each of the features listed in the Assurance Activity.

Component Testing Assurance Activities: Test 1: For each security type specified in the TSS, configure the network to the approved security type and verify that the client can establish a connection. Maintaining the same SSID, change the security type of the client to a non-approved security type and attempt to establish a connection. Verify that the connection was unsuccessful.

Test 2: For each authentication protocol specified in the TSS, configure the network accordingly per the AGD. Verify that the client connection attempt is successful when using the correct client credentials and that the connection is unsuccessful when incorrect authentication credentials are used.

Test 3: Configure the SSID to be broadcasted. Using a network sniffing tool, capture a beacon frame and confirm that the SSID is included. Configure the SSID to be hidden. Using a network sniffing tool, capture a beacon frame and confirm that the SSID is not listed.

Test 4: The evaluator will configure the AS to operate in each of the selected frequency bands and verify using a network sniffing tool.

Test 5: The evaluator will demonstrate that the client can establish a connection to the AS on the default power level. After disconnecting, the power level should be adjusted and then the client should be able to successfully connect to the AS again.

Test 1: For each security type claimed by the TOE, the evaluator first configured the WLAN client to use the approved security type and attempted a connection The evaluator observed these connection attempts were successful. The evaluator then maintained the SSID configured on the TOE, and configured the client for a security type that was not configured/approved on the TOE. The evaluator then caused the client to attempt a connection to the TOE. The evaluator observed that these connection attempts using a non-approved security type were unsuccessful.

Test 2: The evaluator attempted two connections with each authentication protocol (WPA2-PSK, WPA3-SAE, WPA2 Enterprise, WPA3 Enterprise). The first attempt had the client provide correct authentication credentials, and the connection was successful. The following attempt had the client provide incorrect credentials, and the connection was unsuccessful.

Test 3: The evaluator configured the TOE to broadcast its SSID following Operational Guidance. The evaluator started a sniffing session of wireless traffic, and confirmed that the resulting packet capture included a beacon frame that listed the TOEs SSID. The evaluator then configured the TOE to hide its SSID, and repeated the process. The evaluator confirmed that the resulting packet capture included a beacon frame that did not list the SSID.

Test 4: The evaluator configured the TOE following operational guidance to broadcast only on each of the claimed bands (2.4ghz, 5ghz, 6ghz.) For each band, the evaluator disabled the other two and attempted to connect a client, which was successful. The evaluator then verified that the client connected with the configured band using a command on the TOE.

Test 5: For each of the claimed radio band frequencies (2.4ghz, 5ghz, 6ghz), the evaluator first demonstrated a successful connection from a client to the AS using the default power level. The evaluator then disconnected from the AS, then adjusted the power level. With this new power level setting, the evaluator attempted to connect the client again and observed that the connection attempt was successful.

2.6.9 Specification of Management Functions (STFFW14e:FMT_SMF.1/FFW)

2.6.9.1 STFFW14e:FMT_SMF.1.1/FFW

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, 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).

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 NDcPP22e:FMT_SMF.1

Component Guidance Assurance Activities: See TSS Activity.

See TSS Activity.

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.

The management functions identified by STFFW14e:FMT_SMF.1 were used (at least) during the identified tests.

• Ability to configure firewall rules (STFFW14e:FFW_RUL_EXT.1 and VPNGW12:FPF_RUL_EXT.1)

2.6.10 Specification of Management Functions (VPNGW12:FMT_SMF.1/VPN)

2.6.10.1 VPNGW12:FMT_SMF.1.1/VPN

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 confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the TSS identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

See NDcPP22e:FMT_SMF.1

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the operational guidance identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface.

Section "Aruba Firewall high-level concepts" identifies and describes the logical interfaces used for trusted network and VPN communications.

Section "FIA_UIA_EXT.1" of the Admin Guide states that The TOE permits authentication by an administrator through SSH or Web UI over TLS or via a serial console (direct) connection to the CLI.

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

All TOE security functions are identified in the Operational Guidance and have been tested as documented throughout this AAR.

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

2.6.11.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.11.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.11.3 NDcPP22e:FMT_SMR.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 determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Section 6.5 in the ST explains that the TOE supports role-based authentication. Users can authenticate to an external authentication server or to the Controller's internal database. Prior to requiring the non TOE entity to initiate the identification and authentication process, the TOE displays an administrator configured advisory notice and consent warning message regarding unauthorized use of the TOE. The TOE requires an administrator to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that user.

Remote administrators are configured as users who have privileges to access the CLI and Web GUI administration interfaces and who are authenticated as users using the local database or an external authentication server. The remote administrators authenticate as users using a username and password via Web GUI and username/password or public key for SSH. The Web GUI interface provides a trusted path to connect to the TOE via HTTPS. The HTTPS interface uses a server RSA or ECDSA certificate which is stored on the TOE. SSH provides a trusted path to connect to the CLI. It uses SSH_RSA for public keys. Direct console to the CLI only supports username/password. A successful logon takes place when a recognized username/password combination is provided and/or a recognized X.509 client certificate is presented by the administrator's web browser or SSH client.

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 NDcPP22e:FIA_UIA_EXT.1. The Admin Guide does not identify any configuration that needs to be performed on the client for remote administration.

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.

All administrative interfaces were used throughout the course of testing including console and SSH for all devices and HTTPS.

2.6.12 No Administration from Client (WLANAS10:FMT_SMR_EXT.1)

2.6.12.1 WLANAS10:FMT_SMR_EXT.1.1

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: The evaluator will review the operational guidance 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. The evaluator will confirm that the TOE does not permit remote administration from a wireless client by default.

See NDcPP22e:FIA_UIA_EXT.1. The Admin Guide does not identify any configuration that needs to be performed on the client for remote administration. The evaluator confirmed through testing of this SFR that by default, the TOE does not permit remote administration by wireless clients.

Component Testing Assurance Activities: The evaluator will demonstrate that after configuring the TOE for first use from the operational guidance, it is possible to establish an administrative session with the TOE on the 'wired' portion of the device. They will then demonstrate that an identically configured wireless client that can successfully connect to the TOE cannot be used to perform administration.

Test 1: The evaluator first configured the wired connection for admin authentication and verified the TOE was able to perform administrative actions. Next the evaluator attempted the same connection on the wireless device and was successfully able to connect to the AP and ping the TOE’s interface, however, it was unable to connect to the web administration page.

2.7 Packet Filtering (FPF)

2.7.1 Packet Filtering Rules - per TD0683 (VPNGW12:FPF_RUL_EXT.1)

2.7.1.1 VPNGW12:FPF_RUL_EXT.1.1

TSS Assurance Activities: The evaluator shall verify that the TSS provide a description of the TOE's initialization/startup process, which clearly indicates where processing of network packets begins to take place, and provides a discussion that supports the assertion that packets cannot flow during this process.

The evaluator shall verify that the TSS also includes a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describes the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets.

Section 6.4 of the ST provides a description of the TOE's startup and packet processing. The TOE is comprised of a data plane and a control plane. Network packets are processed in the data plane, which is the first component that is initialized. Network interfaces are not brought 'up' until initialization is complete and the data plane operating system is fully initialized. All packet level enforcement is performed within the data plane. In case of system error, packets are dropped by default. The control plane operating system (management interfaces) boots simultaneously.

Guidance Assurance Activities: The operational guidance associated with this requirement is assessed in the subsequent test EAs.

Refer to the subsequent test assurance activities below.

Testing Assurance Activities: Test 1: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would otherwise be denied by the ruleset should be sourced and directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization.

Test 2: The evaluator shall attempt to get network traffic to flow through the TOE while the TOE is being initialized. A steady flow of network packets that would be permitted by the ruleset should be sourced and directed to a host. The evaluator shall use a packet sniffer to verify none of the generated network traffic is permitted through the TOE during initialization and is only permitted once initialization is complete.

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test Evaluation Activities.

These tests were performed and described as part of the component testing assurance activities for FFW_RUL_EXT.1.1 where the evaluator determined that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

2.7.1.2 VPNGW12:FPF_RUL_EXT.1.2

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet piltering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

See VPNGW12: FPF_RUL_EXT.1.4

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.3 VPNGW12:FPF_RUL_EXT.1.3

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

See VPNGW12: FPF_RUL_EXT.1.4

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.1.4 VPNGW12:FPF_RUL_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes a packet filtering policy that can use the following fields for each identified protocol, and that the RFCs identified for each protocol are supported:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

The evaluator shall verify that the TSS describes how conformance with the identified RFCs has been determined by the TOE developer (e.g., third party interoperability testing, protocol compliance testing).

The evaluator shall verify that each rule can identify the following actions: permit, discard, and log.

The evaluator shall verify that the TSS identifies all interface types subject to the Packet Filtering policy and explains how rules are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used), they can be treated collectively as a distinct network interface.

Section 6.4 in the ST explains that the TOE performs stateful packet filtering. Filtering rules may be applied to appliance Ethernet interfaces and to user-roles (wireless clients connecting through APs are placed into userroles). The TOE allows the stateful packet filtering rules to be assigned to each distinct network interface. The interfaces can be viewed through the CLI command “show interface”.

The TOE supports stateful processing of the following protocols:

RFC 792 (ICMPv4)

RFC 4443 (ICMPv6)

RFC 791 (IPv4)

RFC 2460 (IPv6)

RFC 793 (TCP)

RFC 768 (UDP)

The TOE allows the definition of a stateful packet filtering policy based on the following attributes that are used to define rules (based on the actions: permit, deny or log) for the associated protocols: Note: ICMPv4 Code 134 is an unsolicited router advertisement which is discarded/dropped by the TOE. All other traffic is handled based upon access control lists configured on the TOE.

ICMPv4

o Type

o Code

ICMPv6

o Type

o Code

IPv4

o Source address

o Destination Address

o Transport Layer Protocol

IPv6

o Source address

o Destination Address

o Transport Layer Protocol

o IPv6 Extension header type

TCP

o Source Port

o Destination Port

UDP

o Source Port

o Destination Port

The Aruba Quality Assurance (QA) team performs protocol compliance testing using standards based tools and interoperability testing using a range of external vendor equipment.

Guidance Assurance Activities: The evaluators shall verify that the operational guidance identifies the following protocols as being supported and the following attributes as being configurable within packet filtering rules for the associated protocols:

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

The evaluator shall verify that the operational guidance indicates that each rule can identify the following actions: permit, discard, and log.

The evaluator shall verify that the operational guidance explains how rules are associated with distinct network interfaces.

The guidance may describe the other protocols contained within the ST (e.g., IPsec, IKE, potentially HTTPS, SSH, and TLS) that are processed by the TOE. The evaluator shall ensure that it is made clear what protocols were not considered as part of the TOE evaluation.

Section "FFW_RUL_EXT.1.2/3/4" of the Admin Guide lists each protocol and their attributes from the requirement and provides an example for how to set a firewall rule for each protocol including setting a permit, deny and log rule.

Section “FFW_RUL_EXT.1.5” in the Admin Guide provides the commands and examples for assigning rules to an interface.

As identified in this report, the Admin Guide also describes the IPsec, IKE, HTTPS, SSH and TLS protocols.

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

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet filter rules can be created that permit, discard, and log packets for each of the following attributes:

- IPv4

o Source address

o Destination Address

o Protocol

- IPv6

o Source Address

o Destination Address

o Next Header (Protocol)

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

Test 2: The evaluator shall repeat Test 1 above for each distinct network interface type supported by the TOE to ensure that Packet filtering rules can be defined for each all supported types.

Note that these test activities should be performed in conjunction with those of FPF_RUL_EXT.1.6 where the effectiveness of the rules is tested; here the evaluator is just ensuring the guidance is sufficient and the TOE supports the administrator creating a ruleset based on the above attributes. The test activities for FPF_RUL_EXT.1.6 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfill the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

Test 1 & 2: The evaluator configured firewall rules for testing of the other VPNGW12:FPF_RUL_EXT.1 (including VPNGW12:FPF_RUL_EXT.1.6) tests using instructions provided within the administrative guidance and found all necessary instructions were provided accurately. The tests performed for FPF_RUL_EXT.1.6 incorporate numerous variations of packet filtering rules that demonstrate proper enforcement of the packet filtering ruleset (e.g., permit, deny, and log rules, ICMPv*, IPv4 and IPv6, TCP and UDP, numerous ports, source & destination differences, and transport protocols).

2.7.1.5 VPNGW12:FPF_RUL_EXT.1.5

TSS Assurance Activities: The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset.

Section 6.4 of the ST states that the TOE allows the stateful packet filtering rules to be assigned to each distinct network interface. The interfaces can be viewed through the CLI command “show interface”. Stateful session tracking is established and maintained by the data plane operating system thorough inspection of network packets, including handshake transactions.

The algorithm applied to incoming packets is as follows:

Check for IP fragments and assemble.

Parse and identify protocol in the IP packet

Perform length checks and apply default rules.

Enforce interface access-lists (ACLs) if configured.

Lookup session. If exists, then apply corresponding session / stateful ACLs.

Add session if it doesn’t exist. (stateful if the protocol warrants, as listed above.). If stateful also open reverse ports for returning/stateful session. The protocol attributes identified above are used in session determination and will include IP protocol attributes for higher level protocols. TCP processing is further described below. Generate log message, if the ‘log’ keyword is configured in the rule.

Derive role for the user and apply role based ACLs. If no role ACLs, then apply default ACL (deny).

Perform bandwidth contract enforcement.

Perform NATing if required.

During TCP session establishment, the session is identified by source IP, destination IP, source port, and destination port. By default, the three-way handshake is not enforced before data is allowed. This behavior can be changed using the configuration “firewall enforce-tcp-handshake”. Once the session has been established in the datapath session table, future packets which match the source IP, destination IP, source port, and destination port are processed by the “fast path” which was previously programmed during session establishment.

By default, TCP sequence numbers are ignored. This behavior can be changed using the configuration “firewall enforce-tcp-sequence”. Where the controller is used as a wired firewall, sequence number checks do provide some value and in this case the feature should be enabled.

TCP flags are largely ignored by the firewall, with the obvious exception of SYN/ACK during session establishment, and FIN/RST during session teardown.

The TOE enforces the default stateful traffic filtering rules specified in FFW_RUL_EXT.1.6 and any administrator defined rules. These rules, called access-lists (ACLs), are in the sequence specified in the packet processing algorithm above. Only a single access-list may be applied to an Ethernet interface. Multiple access-lists may be applied to a user-role. If multiple access-lists are applied, they are processed in order from top to bottom. The first match found is selected and no further processing takes place. If no match is found, the default action is deny.

The TOE is also capable of dropping and logging network packets according to the rules specified in FFW_RUL_EXT.1.7. The administrator must configure a default Access Control List in order to ensure that this traffic is dropped and logged.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes how the order of Packet filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing.

Section “FFW_RUL_EXT.1.2/3/4” of the Admin Guide states that rules should be configured in order from highest priority to lowest; enforcement is based on a first-match principle where the first rule that matches a traffic flow is applied, and further rules are not processed. The <position> field may be used to insert new rules somewhere other than at the end of a policy.

Section “FFW_RUL_EXT.1.8” of the Admin Guide states the TOE access control lists function in a hierarchical structure. If a rule is applied at the beginning of the ACL, it will take priority over any rule following it. For instance, if an access rule denies a specific IP address and a later rule permits a subnet that contains the specific host, the initial rule denying that specific IP will be applied and traffic from that IP address will be dropped.

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

Test 1: The evaluator shall devise two equal Packet Filtering rules with alternate operations â€“ permit and discard. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.

Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule.

Test 1: The evaluator configured the TOE (according to the admin guide) with two packet filtering rules using the same matching criteria, where one rule would permit while the other would deny traffic. Packets matching the ACL entry rule were sent through the TOE and the evaluator observed that the action taken by the TOE matched the action specified by the first ACL entry.

Test 2: Continuing test 1, the evaluator repeated the procedure above, except the evaluator changed the rules to make one a subset of the other, and then tested both orders. The evaluator confirmed that the first rule is enforced regardless of the specificity of the rule.

2.7.1.6 VPNGW12:FPF_RUL_EXT.1.6

TSS Assurance Activities: The evaluator shall verify that the TSS describes the process for applying Packet Filtering rules and also that the behavior (either by default, or as configured by the administrator) is to discard packets when there is no rule match. The evaluator shall verify the TSS describes when the IPv4/IPv6 protocols supported by the TOE differ from the full list provided in the RFC Values for IPv4 and IPv6 table.

Section 6.4 of the ST states that the TOE enforces the default stateful traffic filtering rules specified in FFW_RUL_EXT.1.6 and any administrator defined rules. These rules, called access-lists (ACLs), are in the sequence specified in the packet processing algorithm above. Only a single access-list may be applied to an Ethernet interface. Multiple access-lists may be applied to a user-role. If multiple access-lists are applied, they are processed in order from top to bottom. The first match found is selected and no further processing takes place. If no match is found, the default action is to deny.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the operational guidance provides the appropriate instructions to configure the behavior to discard packets with no matching rules. The evaluator shall verify that the operational guidance describes the range of IPv4 and IPv6 protocols supported by the TOE.

Section “FFW_RUL_EXT.1.9” of the Admin Guide states that an access control list must be applied to all in-use interfaces in the evaluated configuration. If traffic is sent to an interface and no rule within the ACL applied to that interface matches the packet in-transit, the packet will be dropped. To ensure logging of traffic, the following rule should be applied as the last rule of an ACL to drop and log all unwanted traffic: (config-sess-ACL)#any any any deny log. All unsolicited messages are dropped by default.

Section “FFW_RUL_EXT.1.6” identifies the protocols that are dropped by default.

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

Test 1: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 2: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 3: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 4: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 5: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 6: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 7: The evaluator shall configure the TOE to permit and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

Test 8: The evaluator shall configure the TOE to discard and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 9: The evaluator shall configure the TOE to permit and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Here the evaluator ensures that the UDP port 500 (IKE) is included in the set of tests.

Test 10: The evaluator shall configure the TOE to discard and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Again, the evaluator ensures that UDP port 500 is included in the set of tests.

The following table identifies the RFC defined values for the protocol fields for IPv4 and IPv6 to be used in configuring and otherwise testing Packet Filtering rule definition and enforcement:

Test 1: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured. The packets that were sent were constructed such that the packet would match only one configured rule.

• IPv4 Protocol permitted and logged based on specific source and destination addresses.

• IPv4 Protocol permitted and logged based on specific destination addresses.

• IPv4 Protocol permitted and logged based on specific source addresses.

• IPv4 Protocol permitted and logged based on wildcard addresses.

Test 2: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

• IPv4 Protocol all permitted with some denied and logged based on specific source and destination addresses.

• IPv4 Protocol all permitted with some denied and logged based on specific destination addresses.

• IPv4 Protocol all permitted with some denied and logged based on specific source addresses.

• IPv4 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 3: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule.

• IPv4 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.

• IPv4 Protocol some permitted and logged and some denied and logged based on specific destination addresses.

IPv4 Protocol some permitted and logged and some denied and logged based on specific source addresses.

• IPv4 Protocol some permitted and logged and some denied and logged based on wildcard addresses.

• IPv4 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 4: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied.

Additionally, supported protocols are filtered properly and the rules are enforced.

• IPv6 Protocol permitted and logged based on specific source and destination addresses.

• IPv6 Protocol permitted and logged based on specific destination addresses.

• IPv6 Protocol permitted and logged based on specific source addresses.

• IPv6 Protocol permitted and logged based on wildcard addresses.

Test 5: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied.

Additionally, supported protocols are filtered properly and the rules are enforced.

• IPv6 Protocol all permitted with some denied and logged based on specific source and destination

addresses.

• IPv6 Protocol all permitted with some denied and logged based on specific destination addresses.

• IPv6 Protocol all permitted with some denied and logged based on specific source addresses.

• IPv6 Protocol all permitted with some denied and logged based on wildcard addresses.

Test 6: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied.

Additionally, supported protocols are filtered properly and the rules are enforced.

• IPv6 Protocol some permitted and logged and some denied and logged based on specific source and destination addresses.

• IPv6 Protocol some permitted and logged and some denied and logged based on specific destination addresses.

• IPv6 Protocol some permitted and logged and some denied and logged based on specific source addresses.

• IPv6 Protocol some permitted and logged and some denied and logged based on wildcard addresses.

• IPv6 Protocol some permitted and logged and some denied and logged based on default addresses.

Test 7: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied.

Additionally, supported protocols are filtered properly and the rules are enforced.

• TCP (IPv4) permitted and logged based on source port.

• TCP (IPv4) permitted and logged based on destination port.

• TCP (IPv4) permitted and logged based on source and destination port.

• TCP (IPv6) permitted and logged based on source port.

• TCP (IPv6) permitted and logged based on destination port.

• TCP (IPv6) permitted and logged based on source and destination port.

Test 8: The evaluator attempted to send packets through the TOE when the TOE had the following rules

configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied.

Additionally, supported protocols are filtered properly and the rules are enforced.

• TCP (IPv4) denied and logged based on source port.

• TCP (IPv4) denied and logged based on destination port.

• TCP (IPv4) denied and logged based on source and destination port.

• TCP (IPv6) denied and logged based on source port.

• TCP (IPv6) denied and logged based on destination port.

• TCP (IPv6) denied and logged based on source and destination port.

Test 9: The evaluator attempted to send packets through the TOE when the TOE had the following rules

configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied.

Additionally, supported protocols are filtered properly and the rules are enforced.

• UDP (IPv4) permitted and logged based on source port.

• UDP (IPv4) permitted and logged based on destination port.

• UDP (IPv4) permitted and logged based on source and destination port.

• UDP (IPv6) permitted and logged based on source port.

• UDP (IPv6) permitted and logged based on destination port.

• UDP (IPv6) permitted and logged based on source and destination port.

Test 10: The evaluator attempted to send packets through the TOE when the TOE had the following rules configured: The packets that were sent were constructed such that the packet would match only one configured rule. The evaluator confirmed that packets sent using protocols that are not supported by the TOE are denied.

Additionally, supported protocols are filtered properly and the rules are enforced.

• UDP (IPv4) denied and logged based on source port.

• UDP (IPv4) denied and logged based on destination port.

• UDP (IPv4) denied and logged based on source and destination port.

• UDP (IPv6) denied and logged based on source port.

• UDP (IPv6) denied and logged based on destination port.

• UDP (IPv6) denied and logged based on source and destination port.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8 Protection of the TSF (FPT)

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

2.8.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 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 6.8 of the ST states that passwords are not stored in plaintext. They are stored in flash using a SHA1 hash.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.2 Failure with Preservation of Secure State (WLANAS10:FPT_FLS.1)

2.8.2.1 WLANAS10: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 will examine the TSS to determine that the TOE's implementation of the fail secure functionality is documented. The evaluator will examine the TSS to ensure that it describes all failure conditions and how a secure state is preserved if any of these failures occur. The evaluator will ensure that the definition of a secure state is suitable to ensure the continued protection of any key material and user data.

Section 6.5 of the ST states that in order to prevent an insecure state, the TOE will shutdown when the following failures occur: failure of power on self-tests, failure of integrity check of the TSF executable image and failure of noise source health tests. If a self-test fails, the TOE will immediately halt operation and enter an error state thereby preventing potentially insecure operations (i.e., maintaining a secure state). The controller will reboot after a self-test failure. During reboot, memory is re-initialized, which wipes all keys and user data. If a self-test failure continues to occur, the controller will continue to reboot repeatedly and will require return to manufacturer

Component Guidance Assurance Activities: The evaluator will examine the operational guidance to verify that it describes applicable recovery instructions for each TSF failure state.

Section FPT_TST_EXT.1 of the Admin Guide states that the controller will reboot after a self-test failure. If a self-test failure continues to occur, the controller will continue to reboot repeatedly and will require return to manufacturer. If a firmware image fails its integrity check, the TOE will load the previous image (if one is present). An error will be output during boot in this instance stating that the firmware validation failed.

Component Testing Assurance Activities: For each failure mode specified in the ST, the evaluator will ensure that the TOE attains a secure state (e.g., shutdown) after initiating each failure mode type.

The evaluator verified that for each failure mode specified in the ST, that the TOE attains a secure state (shutdown) after initiating each failure mode type

2.8.3 Failure with Preservation of Secure State (Self-Test Failures) (VPNGW12:FPT_FLS.1/SelfTest)

2.8.3.1 VPNGW12:FPT_FLS.1.1/SelfTest

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure the TSS describes how the TOE ensures a shutdown upon a self-test failure, a failed integrity check of the TSF executable image, or a failed health test of the noise source. If there are instances when a shut-down does not occur, (e.g., a failure is deemed non- security relevant), the evaluator shall ensure that those cases are identified and a rationale is provided that supports the classification and justifies why the TOE's ability to enforce its security policies is not affected in any such instance.

Section 6.8 of the ST states that In order to prevent entering an insecure state, the TOE will shutdown when the following failures occur: failure of power on self-tests, failure of integrity check of the TSF executable image and failure of noise source health tests.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides information on the self-test failures that can cause the TOE to shut down and how to diagnose the specific failure that has occurred, including possible remediation steps if available.

Section “FPT_TST_EXT.1” in the Admin Guide states that if a self-test fails, the TOE will immediately halt operation and enter an error state thereby preventing potentially insecure operations (i.e., maintaining a secure state). The controller will reboot after a self-test failure. During reboot, memory is re-initialized, which wipes all keys and user data. If a self-test failure continues to occur, the controller will continue to reboot repeatedly and will require return to manufacturer. The error output of a failed self-test will appear as follows: “FIPS Aruba Cryptographic asymmetric key KAT failure, main: FIPS_powerupSelfTest failed.” If a firmware image fails its integrity check, the TOE will load the previous image (if one is present). An error will be output during boot in this instance stating that the firmware validation failed. If the issue continues, the administrator should contact support at http://support.arubanetworks.com.

Component Testing Assurance Activities: None Defined

2.8.4 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1)

2.8.4.1 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 6.5 of the ST states that the TOE provides no interfaces that allow pre-shared, symmetric or private keys to be read. Table 12 in Section 6.2 of the ST provides a detailed listing of where each key is stored and how it is stored. For those keys stored encrypted, the table identifies how the key is encrypted.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.5 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1)

2.8.5.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.5.2 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.

Section 6.5 of the ST states the TOE has an internal hardware clock that provides reliable time stamps used for auditing. VMCs use Timestamp Counters (TSC) as the clock source through the hypervisor. The internal clock may be synchronized with a time signal obtained from an external NTP server. The clock is used primarily to provide a timestamp for audit records, but is also used to support timing elements of cryptographic functions, certificate validity checks, session timeouts, and unlocking of administrator accounts locked as a result of authentication failure.

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.

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.

Section “FPT_STM_EXT.1” in the Admin Guide provides instructions for configuring the time manually as well as configuring the use of NTPv4 servers via the ‘ntp server <ip address>’ command which specifies the NTP servers that the NTP client can connect to. The TOE supports configuration of up to 3 NTP servers. If a remote NTP server is used, the administrator must ensure that the connection is protected via IPsec. Section “FCS_IPSEC_EXT.1” in the Admin Guide provides instructions for configuring IPsec connections.

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.

Test 1 - The evaluator set the local clock from the console and observed the time change and corresponding audit record.

Test 2 - The evaluator configured NTP and observed the time change and corresponding audit record.

Test 3 - Is not applicable, as the TOE does not obtain its time from an underlying VS.

2.8.6 TSF TESTING (NDcPP22e:FPT_TST_EXT.1)

2.8.6.1 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 6.5 of the ST states that the TOE runs a suite of self-tests during power-up and periodically during operation, which includes demonstration of the correct operation of the hardware and the use of cryptographic functions to verify the integrity of TSF executable code and static data. An administrator can choose to reboot the TOE to perform power-up self-tests. The Mobility Controller runs the suite of FIPS 140-2 validated cryptographic module self-tests during start-up or on request from the administrator, including immediately after generation of a key (FIPS self-tests, including the continuous RNG test). The self tests operate in the same manner on both the MCs and VMCs.

The following test are performed:

ArubaOS OpenSSL Module:

o AES Known Answer Tests (KAT)

o Triple-DES KAT

o RNG KAT

o RSA KAT

o ECDSA (sign/verify)

o SHA (SHA1, SHA256 and SHA384) KAT

o HMAC (HMAC-SHA1, HMAC-SHA256 and HMAC-SHA384) KAT

ArubaOS Cryptographic Module

o AES KAT

o Triple-DES KAT

o SHA (SHA1, SHA256, SHA384 and SHA512) KAT

o HMAC (HMAC-SHA1, HMAC-SHA256, HMAC-SHA384 and HMAC- SHA512) KAT

o RSA (sign/verify)

o ECDSA (sign/verify)

o FIPS 186-2 RNG KAT

ArubaOS Uboot BootLoader Module

o Firmware Integrity Test: RSA 2048-bit Signature Validation

Aruba Hardware Known Answer Tests:

AES KAT

o AES-CCM KAT

o AES-GCM KAT

o Triple DES KAT

o HMAC (HMAC-SHA1, HMAC-SHA256, HMAC-SHA384 and HMAC-SHA512) KAT

The following Conditional Self-tests are performed by the TOE:

Continuous Random Number Generator Test. This test is run upon generation of random data by the switch's random number generators to detect failure to a constant value. The module stores the first random number for subsequent comparison, and the module compares the value of the new random number with the random number generated in the previous round and enters an error state if the comparison is successful.

Bypass test. Ensures that the system has not been placed into a mode of operation where cryptographic operations have been bypassed, without the explicit configuration of the cryptographic officer. To conduct the test, a SHA1 hash of the configuration file is calculated and compared to the last known good hash of the configuration file. If the hashes match, the test is passed. Otherwise, the test fails (indicating possible tampering with the configuration file) and the system is halted.

RSA Pairwise Consistency test. When the TOE generates a public and private key pair, it carries out pairwise consistency tests for both encryption and digital signing. The test involves encrypting a randomly generated message with the public key. If the output is equal to the input message, the test fails. The encrypted message is then decrypted using the private key and if the output is not equal to the original message, the test fails. The same random message is then signed using the private key and then verified with the public key. If the verification fails, the test fails.

ECDSA Pairwise Consistency test. See above RSA pairwise consistency test description.

Firmware Load Test. This test is identical to the Uboot BootLoader Module Firmware Integrity Test, except that it is performed at the time a new software image is loaded onto the system. Instead of being performed by the BootLoader, the test is performed by the ArubaOS operating system. If the test fails, the newly loaded software image will not be copied into the image partition, and instead will be deleted.

Known-answer tests (KAT) involve operating the cryptographic algorithm on data for which the correct output is already known and comparing the calculated output with the previously generated output (the known answer). If the calculated output does not equal the known answer, the known-answer test shall fail.

If a self-test fails, the TOE will immediately halt operation and enter an error state thereby preventing potentially insecure operations (i.e., maintaining a secure state). The controller will reboot after a self-test failure. During reboot, memory is re-initialized, which wipes all keys and user data. If a self-test failure continues to occur, the controller will continue to reboot repeatedly and will require return to manufacturer.

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 “FPT_TST_EXT.1” in the Admin Guide states that if a self-test fails, the TOE will immediately halt operation and enter an error state thereby preventing potentially insecure operations (i.e., maintaining a secure state). The controller will reboot after a self-test failure. During reboot, memory is re-initialized, which wipes all keys and user data. If a self-test failure continues to occur, the controller will continue to reboot repeatedly and will require return to manufacturer. The error output of a failed self-test will appear as follows: “FIPS Aruba Cryptographic asymmetric key KAT failure, main: FIPS_powerupSelfTest failed.” If a firmware image fails its integrity check, the TOE will load the previous image (if one is present). An error will be output during boot in this instance stating that the firmware validation failed. If the issue continues, the administrator should contact support at http://support.arubanetworks.com.

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.

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 TOE performs an integrity check on its image and executes FIPS Power on self-tests during boot. The evaluator booted the TOE and reviewed the results of the self-tests in the console messages.

2.8.7 TSF Testing (VPNGW12:FPT_TST_EXT.1)

2.8.7.1 VPNGW12:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module requires a particular self-test to be performed, but this self-test is still evaluated using the same methods specified in the Supporting Document.

See NDcPP22e: FPT_TST_EXT.1

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.8 TSF Testing (WLANAS10:FPT_TST_EXT.1)

2.8.8.1 WLANAS10: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 will perform the following activities in addition to those required by the NDcPP:

The evaluator will examine the TSS to ensure that it describes how to verify the integrity of stored TSF executable code when it is loaded for execution, which includes the generation and protection of the 'check value' used to ensure integrity as well as the verification step. This description will also cover the digital signature service used in performing these functions. The evaluator also checks the operational guidance to ensure that any actions required by the administrator to initialize or operate this functionality are present.

The evaluator will also ensure that the TSS or operational guidance describes the actions that take place for successful and unsuccessful execution of the integrity test.

Section 6.5 of the ST details the self-tests. See the description in NDcPP22e:FPT_TST_EXT.1.1

Section FPT_TST_EXT.1 of the Admin Guide describes actions that take place for successful and unsuccessful execution of the integrity test. See NDcPP22e_FPT_TST_EXT.1 for this description.

Component Guidance Assurance Activities: The evaluator will perform the following activities in addition to those required by the NDcPP:

The evaluator will ensure that the TSS or operational guidance describes the actions that take place for successful and unsuccessful execution of the integrity test.

Section FPT_TST_EXT.1 of the Admin Guide states that The module performs at power-on the Cryptographic Algorithm Self-Tests (CASTs), Pre-Operational Self-Tests (POSTs), and Conditional self-tests. After the cryptographic algorithm, pre-operational (including integrity tests), and conditional self-tests are successfully concluded, the module automatically transitions to the operational state and is operating in the approved mode of operation by default. While the module is executing the cryptographic algorithm and pre-operational self-tests, services are not available, and input and output are inhibited. In addition, the module also performs Conditional self-tests. All cryptographic algorithm self-tests are run at power-up, prior to the first operational use of the cryptographic algorithm.

If a self-test fails, the TOE will immediately halt operation and enter an error state thereby preventing potentially insecure operations (i.e., maintaining a secure state). The controller will reboot after a self-test failure. During reboot, memory is re-initialized, which wipes all keys and user data. If a self-test failure continues to occur, the controller will continue to reboot repeatedly and will require return to manufacturer. The error output of a failed self-test will appear as follows: "FIPS Aruba Cryptographic asymmetric key KAT failure, main: FIPS_powerupSelfTest failed." If a firmware image fails its integrity check, the TOE will load the previous image (if one is present). An error will be output during boot in this instance stating that the firmware validation failed.

Component Testing Assurance Activities: The evaluator will perform the following activities in addition to those required by the NDcPP:

The evaluator will perform the following tests:

Test 1: Following the operational guidance, the evaluator will initialize the integrity protection system. The evaluator will perform actions to cause TSF software to load and observe that the integrity mechanism does not flag any executables as containing integrity errors.

Test 2: The evaluator will modify the TSF executable and cause that executable to be loaded by the TSF. The evaluator will observe that an integrity violation is triggered (care must be taken so that the integrity violation is determined to be the cause of the failure to load the module and not the fact that the module was modified so that it was rendered unable to run because its format was corrupt).

Test 1: The evaluator rebooted the TOE and observed that the integrity mechanism did not flag any executables as containing integrity errors.

Test 2: The evaluator attempted to boot an image with an integrity violation and verified that an integrity violation was triggered and the failed integrity image was rejected.

2.8.9 Self-Test with Defined Methods (VPNGW12:FPT_TST_EXT.3)

2.8.9.1 VPNGW12:FPT_TST_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.9.2 VPNGW12:FPT_TST_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator verifies that the TSS describes the method used to perform self-testing on the TSF executable code, and that this method is consistent with what is described in the SFR.

Section 6.8 of the ST describes that the TOE performs self-tests during power-up and periodically during operation.

The following test are performed:

- ArubaOS OpenSSL Module:
  - AES Known Answer Tests (KAT)
  - Triple-DES KAT
  - RNG KAT
  - RSA KAT
  - ECDSA (sign/verify)
  - SHA (SHA1, SHA256 and SHA384) KAT
  - HMAC (HMAC-SHA1, HMAC-SHA256 and HMAC-SHA384) KAT
- ArubaOS Cryptographic Module
  - AES KAT
  - Triple-DES KAT
  - SHA (SHA1, SHA256, SHA384 and SHA512) KAT
  - HMAC (HMAC-SHA1, HMAC-SHA256, HMAC-SHA384 and HMAC- SHA512) KAT
  - RSA (sign/verify)
  - ECDSA (sign/verify)
  - FIPS 186-2 RNG KAT
- ArubaOS Uboot BootLoader Module
  - Firmware Integrity Test: RSA 2048-bit Signature Validation

- Aruba Hardware Known Answer Tests:
  - AES KAT
  - AES-CCM KAT
  - AES-GCM KAT
  - Triple DES KAT
  - HMAC (HMAC-SHA1, HMAC-SHA256, HMAC-SHA384 and HMAC-SHA512) KAT

The following Conditional Self-tests are performed by the TOE:

- **Continuous Random Number Generator Test.** This test is run upon generation of random data by the switch's random number generators to detect failure to a constant value. The module stores the first random number for subsequent comparison, and the module compares the value of the new random number with the random number generated in the previous round and enters an error state if the comparison is successful.

- **Bypass test.** Ensures that the system has not been placed into a mode of operation where cryptographic operations have been bypassed, without the explicit configuration of the cryptographic officer. To conduct the test, a SHA1 hash of the configuration file is calculated and compared to the last known good hash of the configuration file. If the hashes match, the test is passed. Otherwise, the test fails (indicating possible tampering with the configuration file) and the system is halted.

- **RSA Pairwise Consistency test.** When the TOE generates a public and private key pair, it carries out pair-wise consistency tests for both encryption and digital signing. The test involves encrypting a randomly-generated message with the public key. If the output is equal to the input message, the test fails. The encrypted message is then decrypted using the private key and if the output is not equal to the original message, the test fails. The same random message is then signed using the private key and then verified with the public key. If the verification fails, the test fails.

- **ECDSA Pairwise Consistency test.** See above RSA pairwise consistency test description.

- **Firmware Load Test.** This test is identical to the Uboot BootLoader Module Firmware Integrity Test, except that it is performed at the time a new software image is loaded onto the system. Instead of being performed by the BootLoader, the test is performed by the ArubaOS operating system. If the test fails, the newly loaded software image will not be copied into the image partition, and instead will be deleted.

- **Known-answer tests (KAT)** involve operating the cryptographic algorithm on data for which the correct output is already known and comparing the calculated output with the previously generated output (the known answer). If the calculated output does not equal the known answer, the known-answer test shall fail.

This is consistent with what is described in the SFR.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.8.10 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.8.10.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.10.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.10.3 NDcPP22e:FPT_TUD_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 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 6.8 of the ST states that the TOE allows administrators to query the current version of its firmware/software and allows those administrators to manually initiate firmware/software updates. Prior to installing any update, the administrator can verify the digital signature of the update. Administrators can update the TOE executable code using image files manually downloaded from the Aruba support portal. The administrator may perform an update from either the WebUI or CLI. Upgrade instructions are documented in the release notes for each software release, which will be posted in the same directory as the image file on the support portal.

The update image is digitally signed using RSA 2048-bit signature validation. When an update is initiated, the TOE verifies the digital signature with the stored public root CA certificate which is programmed into the boot ROM or virtual boot ROM (ie. for the VMCs) of all Aruba products at the time of manufacturing. Upon successful verification, the TOE boots using the new image. Should verification fail, the TOE will enter into an error state. The TOE’s error state will allow direct console access only, where an administrator can change to a new file partition or TFTP a new image and re-boot.

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 "FPT_TUD_EXT.1" of the Admin Guide describes the update process. The "copy" command is used to download new firmware images from an FTP or TFTP server. From the WebUI, the administrator should navigate to Maintenance>Software Management>Upgrade page to upload an ArubaOS image from a local filesystem. An option is then provided in either case to reboot the device with the new image file.

ArubaOS images are integrity-protected through three methods:

1. When downloading a firmware image from http://support.arubanetworks.com, a file may be found in the download directory that contains SHA256 hashes of each file. This hash may be checked manually after downloading an image.

2. ArubaOS images are digitally signed using RSA 2048 bit signature validation. The mobility controller will check the digital signature immediately after downloading a new firmware image, and will refuse to install an image whose digital signature does not match.

3. Mobility controllers also check the digital signature of an ArubaOS image when booting. The controllerwill refuse to boot a corrupted ArubaOS image file.

The Admin Guide does not identify any functions that may cease to operate during the update. If digital signature verification fails, the TOE will enter into an error state. The TOE's error state will allow direct console access only, where an administrator can change to a new file partition or TFTP a new image and re-boot.

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)

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 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 displayed the version of the product and then installed an update. The signature verified and the update was successfully installed.

Test 2 - The evaluator attempted to upload three images: a corrupted image, an image with an invalid signature and an image missing a signature. In each case, the TOE checks the integrity of the image after the download. The TOE correctly detects an invalid image file and rejects the download.

Test 3 - Not applicable as published hash is not used to verify the integrity of the TOE updates.

2.8.11 Trusted Update (VPNGW12:FPT_TUD_EXT.1)

2.8.11.1 VPNGW12:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.11.2 VPNGW12:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.11.3 VPNGW12:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: There is no change to the Evaluation Activities specified for this SFR in the NDcPP Supporting Document. The PP-Module modifies this SFR to mandate that a particular selection be chosen, but this selection is part of the original definition of the SFR so no new behavior is defined by the PP-Module.

See NDcPP22e:FPT_TUD_EXT.1

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.9 TOE ACCESS (FTA)

2.9.1 TSF-INITIATED TERMINATION (NDCPP22E:FTA_SSL.3)

2.9.1.1 NDCPP22E:FTA_SSL.3.1

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 the administrative remote session termination and the related inactivity time period.

Section 6.9 of the ST states that the TOE terminates remote or local administrator sessions after session inactivity time exceeds a configurable session idle timeout. The session idle timeout is the maximum amount of time an administrator may remain idle.

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 "FTA_SSL.3" of the Admin Guide provides instructions for configuring the inactivity time period for remote and local administrative session termination. For both local and remote administrative sessions, an idle timeout may be set to disconnect idle sessions. The default value is 300 seconds (5 minutes).

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.

Test 1: The evaluator configured timeout values on the TOE. The evaluator then performed an action on the TOE followed by no further activity until after the session timeout. The evaluator observed that in each case, the session is terminated after the configured time period. The evaluator performed this test over both SSH and the TOE's Web UI.

2.9.2 User-initiated Termination (NDcPP22e:FTA_SSL.4)

2.9.2.1 NDcPP22e:FTA_SSL.4.1

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 the local and remote administrative sessions are terminated.

Section "FTA_SSL.4" in the Admin Guide states that an administrator can terminate their own session by exiting the SSH session or logging out from the Web UI session.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Section "FTA_SSL.4" in the Admin Guide states that an administrator can terminate their own session by exiting the SSH session or logging out from the Web UI session.

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 - The evaluator established a local session and manually 'exit'ed per the command identified in the Guidance. The session was terminated and a logout audit record was logged.

Test 2 - The evaluator established a remote session and manually 'exit'ed per the command identified in the Guidance ('log out' via the Web UI). In each case, the session was terminated and a logout audit record was logged. This was tested as part of NDcPP22e:FIA_UIA_EXT.1 where a logout was performed as described.

2.9.3 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.9.3.1 NDcPP22e:FTA_SSL_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 whether local administrative session locking or termination is supported and the related inactivity time period settings.

Section 6.9 of the ST details that local inactive administrator sessions on the TOE are terminated, just like remote inactive administrator sessions, after the configured timeout period.

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 "FTA_SSL_EXT.1" of the Admin Guide provides instructions for configuring the inactivity time period for remote and local administrative session termination. For both local and remote administrative sessions, an idle timeout may be set to disconnect idle sessions. The default value is 300 seconds (5 minutes).

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 tested the TSF-initiated timeout of a console session, by configuring the TOE timeout values for 1 and 2 minutes. The evaluator then performed a login from the console and no further activity until the session was automatically closed by the TOE.

2.9.4 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.9.4.1 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 6.9 of the ST states that whether connecting to the CLI (locally or remotely) or web GUI, the TOE displays an advisory message when an administrator logs on. The message is configurable by TOE administrators.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Section “FIA_UIA_EXT.1” in the Admin Guide provides the command for setting the warning banner.

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.

Test 1: The evaluator configured the TOE for login banners and verified with each method of access that the banner was displayed.

2.9.5 TOE Session Establishment - per TD0656 (VPNGW12:FTA_TSE.1)

2.9.5.1 VPNGW12:FTA_TSE.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 verify that it describes the methods by which the TSF can deny the establishment of an otherwise valid remote VPN client session (e.g., client credential is valid, not expired, not revoked, etc.), including day, time, and IP address at a minimum.

Section 6.9 of the ST states that the TOE can deny establishment of a remote VPN client session based on location, time, day, and blacklist state. Location is defined as the client's IP address. Blacklist refers to a list of denied clients identified by MAC address.

Component Guidance Assurance Activities: The evaluator shall review the operational guidance to determine that it provides instructions for how to enable an access restriction that will deny VPN client session establishment for each attribute described in the TSS.

Section "FTA_TSE.1" in the Admin Guide describes the restriction of user sessions based on location, time, and day. It also details configuration of the max authentication failure limit and associated blacklist commands.

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

Test 1: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it, noting the IP address from which the client connected. The evaluator shall follow the steps described in the operational guidance to prohibit that IP address from connecting, attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 2: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it. The evaluator shall follow the steps described in the operational guidance to prohibit the VPN client from connecting on a certain day (whether this is a day of the week or specific calendar date), attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 3: The evaluator shall successfully connect a remote VPN client to the TOE and then disconnect it. The evaluator shall follow the steps described in the operational guidance to prohibit the VPN client during a range of times that includes the time period during which the test occurs, attempt to reconnect using the same VPN client, and observe that it is not successful.

Test 4: [conditional] If any other attributes are identified in FTA_TSE.1, the evaluator shall conduct a test similar to tests 1 through 3 to demonstrate the enforcement of each of these attributes. The evaluator shall demonstrate a successful remote client VPN connection, configure the TSF to deny that connection based on the attribute, and demonstrate that a subsequent connection attempt is unsuccessful.

Test 1: The evaluator connected a remote VPN client to the TOE and recorded the IP address from which the client connected, and then disconnected the client. The evaluator configured the TOE to prevent that IP address from connecting, then attempted to reconnect using the same VPN client, and observed that it was not successful.

Test 2: The evaluator connected a remote VPN client to the TOE and recorded the IP address from which the client connected, and then disconnected the client. The evaluator configured the TOE to prevent that IP address from connecting based on time of day and month. The evaluator then attempted to reconnect using the same VPN client, and observed that it was not successful.

Test 3: Since the TOE configuration requires a time range to be configured in conjunction with the date range, this test was performed as part of test 2 above.

Test 4: The TOE is able to blacklist wireless clients by the client's MAC address. After blacklisting, the wireless clients no longer have any network access and thus are unable to connect to the VPN gateway server. The evaluator configured the TOE to deny clients based on location (MAC address). The evaluator then added the wireless client to the blacklist and verified that the client was not able to connect to the AP.

2.9.6 TOE Session Establishment - per TD0679 (WLANAS10:FTA_TSE.1)

2.9.6.1 WLANAS10:FTA_TSE.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 all of the attributes on which a client session can be denied are specifically defined.

Section 6.6 of the ST states that in order to limit access to the administrative functions, the TOE can be configured to deny wireless clients and remote VPN clients based on the TOE interface, time, day and blacklist state. Firewall rules are used to restrict access and can be configured to blacklist clients when a rule is violated. Unlike the other properties, the blacklist is dynamically managed by the TOE identifying potentially undesirable network devices based on observed activities. If a device is actively identified in the blacklist, it cannot be used to connect to an administrative interface.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine that it contains guidance for configuring each of the attributes identified in the TSS.

Section "FTA_TSE.1 (VPNGW/WLAN)" in the Admin Guide provides instructions for restricting user sessions based on TOE interface, time, day and blacklist state. It also details configuration of the max authentication failure limit and associated blacklist commands. Denial of a connection through an interface is done through access control lists which are further described in section "FFW_RUL_EXT.1" in the Admin Guide.

Component Testing Assurance Activities: The evaluator shall also perform the following test for each attribute:

Test 1: The evaluator successfully establishes a client session with a wireless client. The evaluator then follows the operational guidance to configure the system so that that client's access is denied based on a specific value of the attribute. The evaluator shall then attempt to establish a session in contravention to the attribute setting (for instance, the client is denied WLAN access based upon the TOE interface (e.g. WLAN access point) it is connecting to or the client is denied access based upon the time-of-day or day-of-week it is attempting connection on). The evaluator shall observe that the access attempt fails.

Test 1 - The evaluator configured the following: a time range to deny wifi access, a command to deny clients based on location, adding a wireless client to the blacklist and denying access based on TOE interface. The evaluator then attempted to establish a session in contravention to these attribute settings and in all cases observed that the access attempt failed.

2.9.7 VPN Client Management - per TD0656 (VPNGW12:FTA_VCM_EXT.1)

2.9.7.1 VPNGW12:FTA_VCM_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 the TSS to verify that it asserts the ability of the TSF to assign a private IP address to a connected VPN client.

Section 6.9 of the ST states the TOE assigns a private IP address (internal to the trusted network for which the TOE is the headend) to a VPN client upon a successful establishment of a security session.

Component Guidance Assurance Activities: There are no operational guidance EAs for this component.

There are no operational guidance EAs for this component.

Component Testing Assurance Activities: The evaluator shall connect a remote VPN client to the TOE and record its IP address as well as the internal IP address of the TOE. The evaluator shall verify that the two IP addresses belong to the same network. The evaluator shall disconnect the remote VPN client and verify that the IP address of its underlying platform is no longer part of the private network identified in the previous step.

The evaluator connected a remote VPN client to the TOE and recorded its IP address with the internal IP address of the TOE. The evaluator verified that the two IP addresses belong to the same network. The evaluator disconnected the remote VPN client and verified that the IP address of its underlying platform after the disconnection was not part of the private network identified in the previous step.

2.10 Trusted path/channels (FTP)

2.10.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1)

2.10.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.1.3 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 6.10 of the ST states the TOE uses the IPsec/IKE protocol with pre-shared keys or certificates to establish a trusted channel between itself and the external authentication server, syslog server, NTP server, and other VPN Gateways (ie. site-to-site VPN). To configure the channels, the administrator uses the Configuration -> Services -> VPN panel of the Web GUI to create the host-to-host IPsec/IKE connections. All configuration settings must specify CAVP tested encryption algorithms as specified by the FCP_COP.1 requirements.

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 “FTP_ITC.1” of the Admin Guide states that the ArubaOS supports IPsec as the inter-TSF trusted channel.

This channel is to be used between a Mobility Controller and a) a syslog server, b) a RADIUS server c) an NTP server, d) remote VPN Gateways/Peers. Instructions for IPsec are in section “FCS_IPSEC_EXT.1” of the Admin Guide.

If for any reason a connection is unintentionally broken, the TOE will re-establish the connection once connectivity is restored. If the timeout period has expired, re-authentication/re-negotiation is required.

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.

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 - The evaluator configured the TOE for external authentication using TACACS, and RADIUS over the tunneled IPsec Network. The evaluator also configured the TOE for NTP and syslog over the tunneled IPsec network.

Test 2 - This was tested through the FCS_IPSEC_EXT.1 tests as well as test 4 below. The TOE can be configured to connect as a client to a peer and will attempt to establish a connection.

Test 3 - This test was performed as part of test 4 below where the packet captures showed that channel data was not sent in plaintext.

Test 4 - With the connection established, the evaluator physically disconnected the network between the TOE and the remote audit/NTP/RADIUS/TACACS+ server. After re-establishing the physical connection, the evaluator verified that the TOE reestablished the encrypted tunnel and no data was sent in plaintext. For the MAC layer timeout, the evaluator physically disrupted the connection for about 2 minutes. For the APP layer timeout, the evaluator physically disrupted the connection for about 5 minutes.

2.10.2 Inter-TSF Trusted Channel (WLANAS10:FTP_ITC.1)

2.10.2.1 WLANAS10:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.2.2 WLANAS10:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.2.3 WLANAS10: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 will perform the following activities in addition to those required by the NDcPP:

The evaluator will examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each communications mechanism is identified in terms of the allowed protocols for that IT entity. The evaluator will also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

Section 6.7 of the ST explains that for wireless users operating in a Robust Security Network (RSN), IEEE 802.11-2016 (WPA3) and 802.11-2012 (WPA2), IEEE 802.1X are used to provide a trusted channel between the TOE and WLAN clients. The TOE uses the IPsec/IKE protocol with pre-shared keys or certificates to establish a trusted channel between itself and WLAN clients and 802.1X authentication servers.

Component Guidance Assurance Activities: The evaluator will perform the following activities in addition to those required by the NDcPP:

The evaluator will 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 “FTP_ITC.1” of the Admin Guide states that the ArubaOS supports IPsec as the inter-TSF trusted channel. This channel is to be used between a Mobility Controller and a) a syslog server, b) a RADIUS server c) an NTP server, d) remote VPN Gateways/Peers. Instructions for IPsec are in section “FCS_IPSEC_EXT.1” of the Admin Guide.

If for any reason a connection is unintentionally broken, the TOE will re-establish the connection once connectivity is restored. If the timeout period has expired, re-authentication/re-negotiation is required.

Component Testing Assurance Activities: The evaluator will perform the following activities in addition to those required by the NDcPP:

The evaluator will perform the following tests:

Test 1: The evaluator will 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.

Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator will follow the guidance documentation to ensure that the communication channel can be initiated from the TOE.

Test 3: The evaluator will ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

Test 4: The evaluator will, for each protocol associated with each authorized IT entity tested during test 1, physically interrupt an established connection. The evaluator will ensure that when physical connectivity is restored, communications are appropriately protected.

Test 1 - The TOE supports IEEE 802.11-2020 wireless traffic, and IPsec wired traffic. Each of these tests were performed as part of test 4 (see below). The AP broadcast protocol is 802.11. The evaluator confirmed through packet capture of the wireless traffic from the AP in test 4.

Test 2 - This test is conducted as part of NDcPP22e:FTP_ITC.1 by determining that the TOE initiates the IPsec channel to request services.

Test 3 - This test is conducted as part of NDcPP22e:FTP_ITC.1 by inspecting the packet captures between the TOE and each IT entity and observing that there is no plaintext data present.

Test 4 - IEEE 802.11-2020: The evaluator configured the TOE to broadcast a wireless network protected by a WPA2 PSK (password) and started a sniffing session of wireless traffic. The evaluator connected the wireless client to the TOE network. Once the client was connected, the evaluator disconnected the wireless client by disabling the Wi-Fi radio on the client for 1 minute. After at least 1 minute had passed, the evaluator reconnected the client to the TOE network. The evaluator observed that the client needed to reauthenticate with the saved WPA2 PSK and transmit new EAPOL messages in order to get access with the TOE network. This test was repeated with WPA3-SAE, WPA2 Enterprise, and WPA3 Enterprise.

2.10.3 Inter-TSF Trusted Channel (WLAN Client Communications) (WLANAS10:FTP_ITC.1/Client)

2.10.3.1 WLANAS10:FTP_ITC.1.1/Client

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.3.2 WLANAS10:FTP_ITC.1.2/Client

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.3.3 WLANAS10:FTP_ITC.1.3/Client

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: This component is adequately evaluated when performing the evaluation activities for FTP_ITC.1 in the Network Device, version 2.2e base-PP.

See NDcPP22e: FTP_ITC.1

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.10.4 Inter-TSF Trusted Channel (VPN Communications) (VPNGW12:FTP_ITC.1/VPN)

2.10.4.1 VPNGW12:FTP_ITC.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

See NDcPP22e:FTP_ITC.1 and NDcPP22e:FCS_IPSEC_EXT.1

Component Guidance Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

See NDcPP22e:FTP_ITC.1 and NDcPP22e:FCS_IPSEC_EXT.1

Component Testing Assurance Activities: TheEAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications. Additional evaluation testing for IPsec is covered in FCS_IPSEC_EXT.1.

See NDcPP22e:FTP_ITC.1 and NDcPP22e:FCS_IPSEC_EXT.1

2.10.5 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin)

2.10.5.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.5.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.10.5.3 NDcPP22e:FTP_TRP.1.3/Admin

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 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 6.10 of the ST states that the for remote administrators, the TOE uses HTTPS/TLS to offer secure remote web GUI-based administration and SSH to offer a secure remote administration CLI. The TOE uses SSH, TLS/HTTPS to provide a trusted communication path between itself and remote administrators that is logically distinct from other communication paths and provides assured identification of its end points and protection of the communicated data from disclosure and detection of modification of the communicated data. This is consistent with the protocols specified in the requirement.

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 “FTP_TRP.1/Admin” of the Admin Guide states that communication between a Mobility Controller and a remote administrator may be protected by TLS/HTTPS (when using the Web-based interface) or SSH (when using the command-line interface). All remote administration must take place over one of these interfaces. The Admin Guide provides instructions for using TLS/HTTPS and SSH in sections FCS_TLSS_EXT.1, FCS_SSHS_EXT.1 and FCS_HTTPS_EXT.1

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.

Test 1: The evaluator configured the TOE for each protocol according to the guidance. The evaluator connected an SSH client to the TOE and verified that the connection was successful, and that the traffic was encrypted. The evaluator connected to the TOE's HTTPS server and ensured that the communication was successful and secured with application data.

Test 2: This test was performed as part of test 1 where the packet captures showed that the network traffic is protected appropriately.

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 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 cPP, 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 Supporting Document (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.

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.

Section “Evaluated Platforms” in the **Admin Guide** identifies the evaluated hardware and software. The manual is organized by sets of requirements so the reader knows what was included in the evaluated configuration.

Section "FCS_CKM.1 in the **Admin Guide** states FIPS mode must be enabled. This ensures the correct cryptography is used.

Section "FPT_TUD_EXT.1" of the **Admin Guide** describes the update process. The "copy" command is used to download new firmware images from an FTP or TFTP server. From the WebUI, the administrator should navigate to Maintenance>Software Management>Upgrade page to upload an ArubaOS image from a local filesystem. An option is then provided in either case to reboot the device with the new image file.

ArubaOS images are integrity-protected through three methods:

1. When downloading a firmware image from http://support.arubanetworks.com, a file may be found in the download directory that contains SHA256 hashes of each file. This hash may be checked manually after downloading an image.
2. ArubaOS images are digitally signed using RSA 2048 bit signature validation. The mobility controller will check the digital signature immediately after downloading a new firmware image, and will refuse to install an image whose digital signature does not match.
3. Mobility controllers also check the digital signature of an ArubaOS image when booting. The controller will refuse to boot a corrupted ArubaOS image file

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).

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:

- Aruba OS 8.10 Supplemental Guidance (Common Criteria Configuration Guidance), 2.0, April 2023 **(Admin Guide)**

In some instances, the document referenced general Aruba manuals which the evaluation team could find on the Aruba 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).

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.

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.

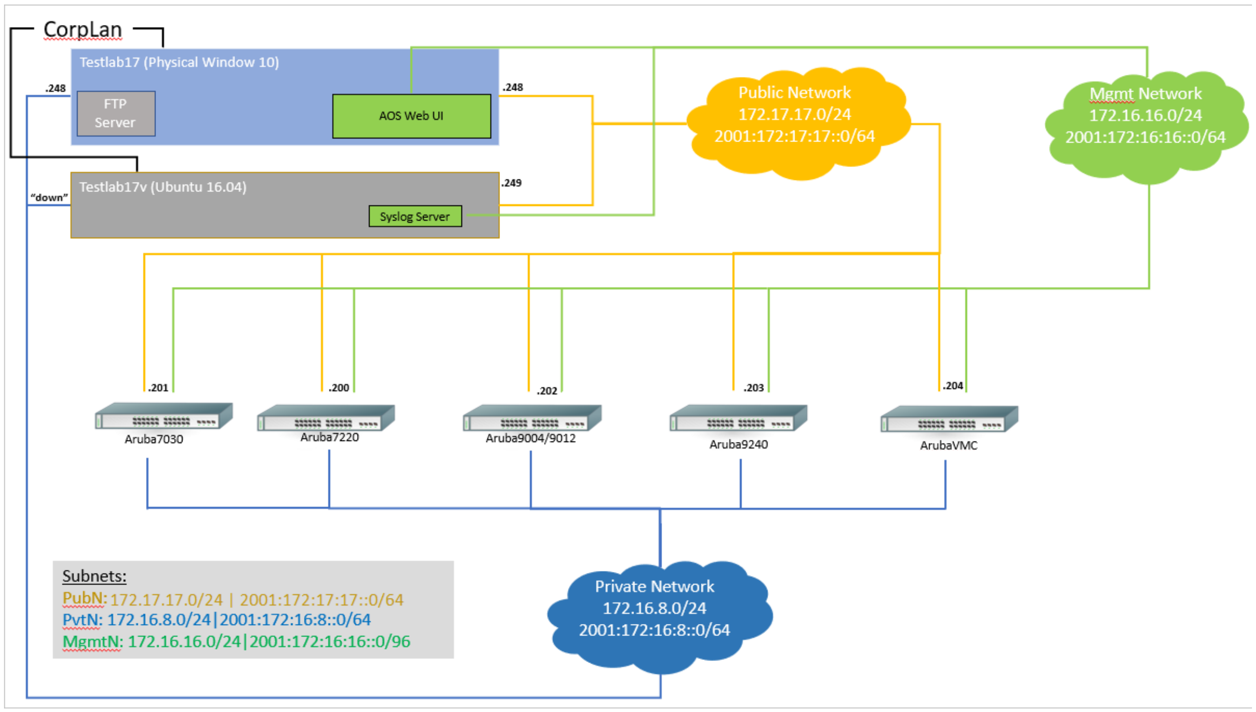

Figure 1 Test Network Setup

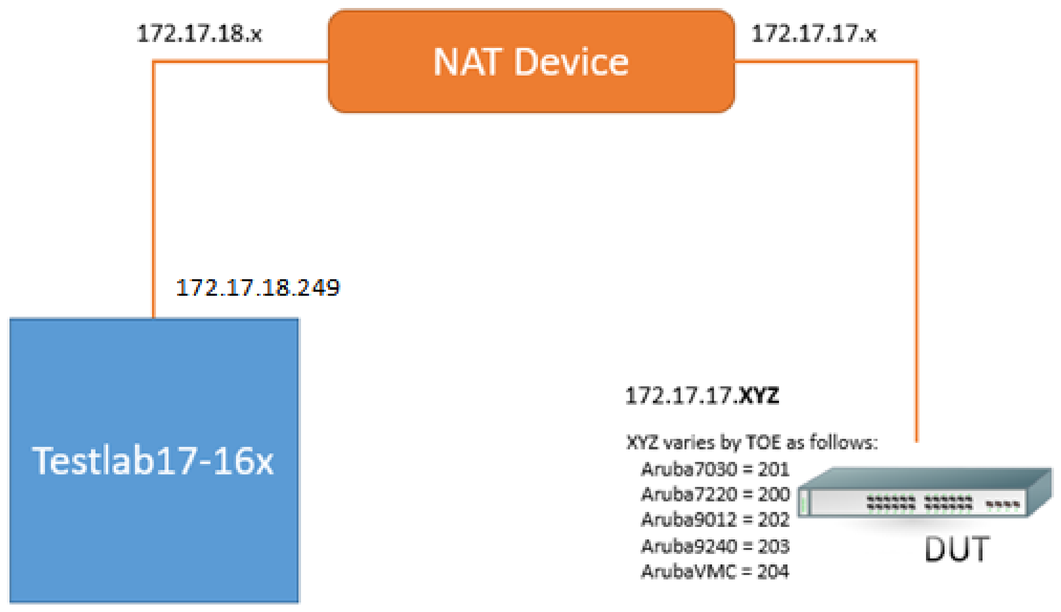

Figure 2 IPsec NAT Test Setup

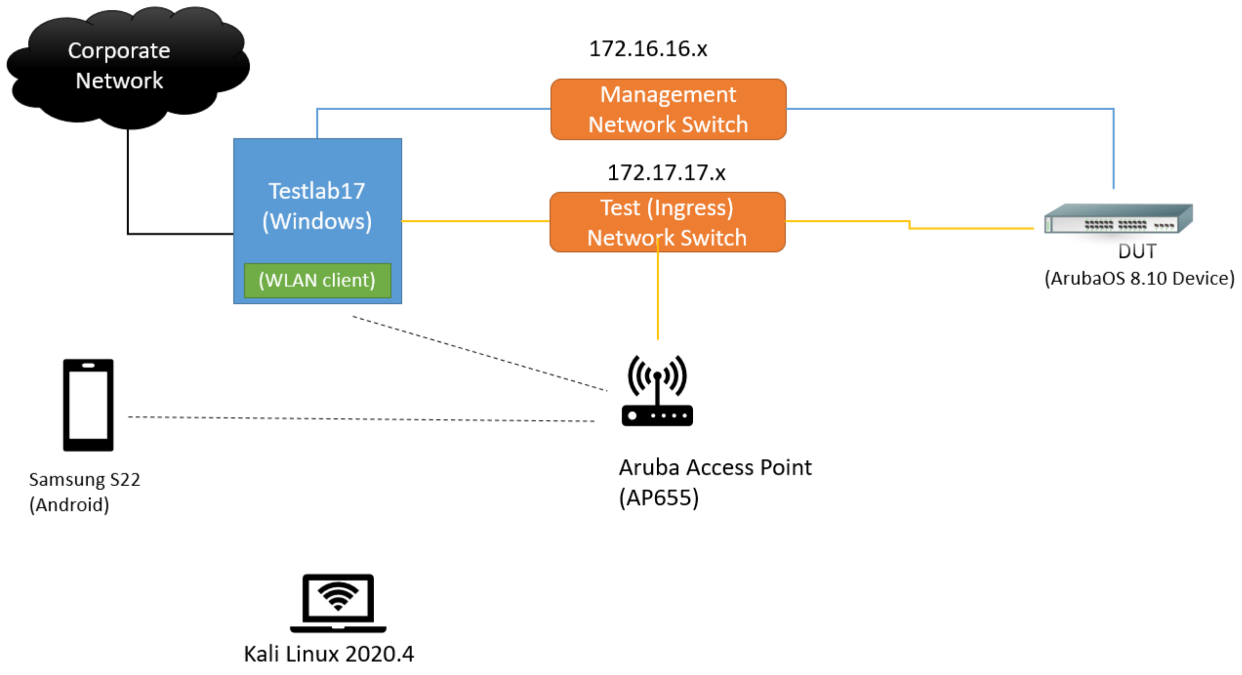

Figure 3 WLAN Test Network Setup

TOE Platforms:

- Aruba7030, Aruba7220, Aruba9004 (also used Aruba9012), and ArubaVMC with ArubaOS version 8.10

Supporting Software:

- Windows 10.0
- Wireshark version 2.6.6
- Windows SSH Client – Putty version 0.76 (used to connect to device console and SSH)

The Gossamer Test servers with an Ubuntu environment acted as platforms to initiate testing. The test servers also acted as a syslog server.

- Openssh-client version 7.2p2
- Big Packet Putty version 6.2
- Nmap version 7.01
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Stunnel version 5.30
- Openssl version 1.0.2g
- Strongswan version 5.2.5

- Rsyslog version 8.16.0
- Virtual Intranet Access (VIA) 3.4.2

3.5 Vulnerability assessment (AVA)

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 components7 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 the TOE) such as a web server and protocol or cryptographic libraries. 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 hypotheses during their analysis.

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.

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 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)
- Exploit / Vulnerability Search Engine (http://www.exploitsearch.net)
- SecurITeam Exploit Search (http://www.securiteam.com)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search)
- Offensive Security Exploit Database (https://www.exploit-db.com/)

The search was performed on 5/19/2023 with the following search terms: "aruba 802.1X", "aruba tcp", "aruba mobility controller", "arubaos", "aruba ipsec", "aruba ssh", "aruba tls", "arubaos openssl", "arubaos uboot", "aruba vmc", "aruba vpn", "esxi", "sos", "sibyte", "Linux OS v2.6.32 kernel", "Linux OS v4.14.181 kernel", "Intel Atom C3508", "Broadcom XLP208", "Broadcom XLP316", "Broadcom XLP416", "Broadcom XLP432", "Broadcom XLP780".

3.5.2 Additional Flaw Hypotheses (AVA_VLA.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 Considerations.

This test is not applicable. The TOE is a TLS server, but does not support ciphersuites that use RSA transport.