www.GossamerSec.com

Assurance Activity Report for Architecture Technology Corporation Machete Router

Version 0.2
1/30/24

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

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

Revision History

Revision	Date	Authors	Summary
Version 0.1	12/14/23	Gossamer	Initial draft
Version 0.2	1/30/24	Gossamer	Updated AVA search

The TOE Evaluation was Sponsored by:
Architecture Technology Corporation
9971 Valley View Road
Eden Prairie, MN 55344

Evaluation Personnel:

- Yoel Fortaleza
- Douglas Kalmus
- Allison Keenan

Common Criteria Versions:

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

Common Evaluation Methodology Versions:

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

Table of Contents

1. Introduction 6
1.1 CAVP Certificates 6
1.2 Evaluated Platform Equivalence 7
1.3 References 8
2. Protection Profile SFR Assurance Activities 9
2.1 Security audit (FAU) 9
2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1) 9
2.1.2 Audit Data Generation (VPN Gateway) (VPNGW12:FAU_GEN.1/VPN) 11
2.1.3 User identity association (NDcPP22e:FAU_GEN.2) 12
2.1.4 Protected Audit Event Storage (NDcPP22e:FAU_STG_EXT.1) 13
2.2 Cryptographic support (FCS) 17
2.2.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1) 17
2.2.2 Cryptographic Key Generation (for IKE Peer Authentication) - per TD0723 (VPNGW12:FCS_CKM.1/IKE) 20
2.2.3 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2) 22
2.2.4 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4) 25
2.2.5 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS_COP.1/DataEncryption) 26
2.2.6 Cryptographic Operation (AES Data Encryption/Decryption) (VPNGW12:FCS_COP.1/DataEncryption) 31
2.2.7 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS_COP.1/Hash) 31
2.2.8 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS_COP.1/KeyedHash) 34
2.2.9 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS_COP.1/SigGen) 34
2.2.10 IPsec Protocol - Per TD0800 (NDcPP22e:FCS_IPSEC_EXT.1) 36
2.2.11 IPsec Protocol - per TD0657 (VPNGW12:FCS_IPSEC_EXT.1) 52
2.2.12 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1) 56
2.2.13 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1) 59
2.2.14 SSH Client Protocol - per TD0636 (NDcPP22e:FCS_SSHC_EXT.1) 61
2.2.15 SSH Server Protocol - per TD0631 (NDcPP22e:FCS_SSHS_EXT.1) 70
2.3 Identification and authentication (FIA) 79

2.3.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1) 79
2.3.2 Password Management - per TD0792 (NDcPP22e:FIA_PMG_EXT.1) 81
2.3.3 Pre-Shared Key Composition (VPNGW12:FIA_PSK_EXT.1) 83
2.3.4 Generated Pre-Shared Keys (VPNGW12:FIA_PSK_EXT.2) 84
2.3.5 Password-Based Pre-Shared Keys - per TD0771 (VPNGW12:FIA_PSK_EXT.3) 85
2.3.6 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7) 89
2.3.7 Password-based Authentication Mechanism (NDcPP22e:FIA_UAU_EXT.2) 90
2.3.8 User Identification and Authentication (NDcPP22e:FIA_UIA_EXT.1) 90
2.3.9 X.509 Certificate Validation (NDcPP22e/VPNGW12:FIA_X509_EXT.1/Rev) 93
2.3.10 X.509 Certificate Authentication (NDcPP22e/VPNGW12:FIA_X509_EXT.2) 98
2.3.11 X.509 Certificate Requests (NDcPP22e:FIA_X509_EXT.3) 100
2.4 Security management (FMT) 101
2.4.1 Management of Security Functions Behaviour (NDcPP22e:FMT_MOF.1/Functions) 101
2.4.2 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate) 104
2.4.3 Management of Security Functions Behaviour (NDcPP22e:FMT_MOF.1/Services) 105
2.4.4 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData) 107
2.4.5 Management of TSF Data (NDcPP22e:FMT_MTD.1/CryptoKeys) 108
2.4.6 Management of TSF Data (VPNGW12:FMT_MTD.1/CryptoKeys) 109
2.4.7 Specification of Management Functions - per TD0631 (NDcPP22e:FMT_SMF.1) 109
2.4.8 Specification of Management Functions (VPNGW12:FMT_SMF.1/VPN) 111
2.4.9 Restrictions on Security Roles (NDcPP22e:FMT_SMR.2) 112
2.5 Packet Filtering (FPF) 113
2.5.1 Packet Filtering Rules - per TD0683 (VPNGW12:FPF_RUL_EXT.1) 113
2.6 Protection of the TSF (FPT) 128
2.6.1 Protection of Administrator Passwords (NDcPP22e:FPT_APW_EXT.1) 128
2.6.2 Failure with Preservation of Secure State (Self-Test Failures) (VPNGW12:FPT_FLS.1/SelfTest) 129
2.6.3 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1) 130
2.6.4 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1) 131
2.6.5 TSF testing (NDcPP22e:FPT_TST_EXT.1) 133
2.6.6 TSF Testing (VPNGW12:FPT_TST_EXT.1) 135

2.6.7 Self-Test with Defined Methods (VPNGW12:FPT_TST_EXT.3) 136
2.6.8 Trusted update (NDcPP22e:FPT_TUD_EXT.1) 137
2.6.9 Trusted Update (VPNGW12:FPT_TUD_EXT.1) 142
2.7 TOE access (FTA) 143
2.7.1 TSF-initiated Termination (NDcPP22e:FTA_SSL.3) 143
2.7.2 TSF-Initiated Termination (VPN Headend) - per TD0656 (VPNGW12:FTA_SSL.3/VPN) 144
2.7.3 User-initiated Termination (NDcPP22e:FTA_SSL.4) 145
2.7.4 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1) 146
2.7.5 Default TOE Access Banners (NDcPP22e:FTA_TAB.1) 146
2.7.6 TOE Session Establishment - per TD0656 (VPNGW12:FTA_TSE.1) 147
2.7.7 VPN Client Management - per TD0656 (VPNGW12:FTA_VCM_EXT.1) 149
2.8 Trusted path/channels (FTP) 150
2.8.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1) 150
2.8.2 Inter-TSF Trusted Channel (VPN Communications) (VPNGW12:FTP_ITC.1/VPN) 152
2.8.3 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin) 153
3. Protection Profile SAR Assurance Activities 156
3.1 Development (ADV) 156
3.1.1 Basic Functional Specification (ADV_FSP.1) 156
3.2 Guidance documents (AGD) 157
3.2.1 Operational User Guidance (AGD_OPE.1) 157
3.2.2 Preparative Procedures (AGD_PRE.1) 159
3.3 Life-cycle support (ALC) 160
3.3.1 Labelling of the TOE (ALC_CMC.1) 160
3.3.2 TOE CM Coverage (ALC_CMS.1) 161
3.4 Tests (ATE) 161
3.4.1 Independent Testing - Conformance (ATE_IND.1) 161
3.5 Vulnerability assessment (AVA) 163
3.5.1 Vulnerability Survey (AVA_VAN.1) 163
3.5.2 Additional Flaw Hypothesis (AVA_VAN.1) 165

1. Introduction

This document presents evaluations results of the Architecture Technology Corporation Machete Router NDcPP22e/VPNGW12 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 CAVP Certificates

The TOE has the following CAVP certificates:

Functions	Requirement	Certificate #
Encryption/Decryption		
AES CBC, GCM, CTR (128 and 256 bits)	NDcPP22e/VPNGW12 FCS_COP.1/DataEncryption	A4781
Cryptographic signature services		
• RSA Digital Signature Algorithm (rDSA) (2048, 3072 bits) • Elliptic Curve Digital Signature Algorithm (P-256, P-384, P-521)	NDcPP:FCS_COP.1/SigGen	A4781
Cryptographic hashing		
SHA-1, SHA-256, SHA-384, SHA-512	NDcPP22e:FCS_COP.1/Hash	A4781
Keyed-hash message authentication		
HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, HMAC-SHA-512	NDcPP22e:FCS_COP.1/KeyedHash	A4781
Random bit generation		
CTR_DRBG with sw based noise sources with a minimum of 256 bits of non-determinism	NDcPP22e:FCS_RBC_EXT.1	A4781
Key generation		
• RSA Key Generation (2048, 3072 bits) • ECC Key Generation (P-256, P-384, P-521)	NDcPP22e:FCS_CKM.1 VPNGW12:FCS_CKM.1/IKE	A4781

• FFC Safe primes key generation	NDcPP22e:FCS_CKM.1	Tested with a known good implementation
Key establishment		
• ECC KAS	NDcPP22e:FCS_CKM.2	A4781
• RSA Key Generation (2048, 3072 bits) • FFC Safe primes key generation	NDcPP22e:FCS_CKM.2	Tested with a known good implementation

1.2 Evaluated Platform Equivalence

The TOE is the Architecture Technology Corporation Machete Router running ARES v2.0. The TOE consists of the following hardware:

Model Identification	Platform	CPU Architecture	CPU Part Number
MACHETE-FIT2	Fitlet2	Intel Apollo Lake	Atom x7-E3950
MACHETE-OTN4	OnTime 4000 Series	Intel Apollo Lake	Atom x7-E3950
MACHETE-OTN6	OnTime 6000 Series	Intel Apollo Lake	Atom x7-E3950
MACHETE-OTN7	OnTime 7000 Series	Intel Apollo Lake	Atom x7-E3950
MACHETE-DCS2	DCS003289	Intel Apollo Lake	Atom x7-E3950
MACHETE-V1	VMware ESXi v7.0	AMD Ryzen 4000	Ryzen 4600G
MACHETE-AMD-R1	OL-ML100 Series	AMD Ryzen V1000	V1605B
MACHETE-WL1	BKNUC8V5PNB	Intel Whiskey Lake	Core i5-8365U
MACHETE-FIT3	Fitlet3	Intel Elkhart Lake	Atom x6425E

The evaluation team ran the entire test suite on each of the following devices:

- MACHETE-FIT2
- MACHETE-V1

The set of hardware noted in the table supports two distinct images. The MACHETE-V1 on ESXi has its own image while all the remaining physical hardware devices share a single image (represented by the MACHETE-FIT2 in testing).

The only differences between the physical devices are related to the different hardware characteristics and does not affect any of the security relevant functionality. Any differing hardware characteristics among the models in each group affect only non-security relevant functionality such as throughput, processing speed, number and type of network connections supported, number of concurrent connections supported, and amount of storage. All security functions provided by the TOE are implemented in software and the TOE security behavior is the same on all the devices for each of the SFRs defined by the Security Target. These SFRs are instantiated by the same version of the TOE software and in the same way on every platform. Where the CPU differs amongst hardware, algorithm testing was performed and showed no difference in cryptographic functionality. As such it is acceptable to test only 1 instance of each software image.

1.3 References

The following evidence was used to complete the Assurance Activities:

- Architecture Technology Corporation Machete Router Security Target, Version 0.6, November 29, 2023 (ST)
- Machete Router Common Criteria Operational Guidance, Version 1.6, December 14, 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 for cryptographic key management events, the TSF includes the identity of the cryptographic key being acted upon (key file name).

The TOE is not distributed.

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 11 of the admin guide contains the audit log messages including all required audit log formatting.

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

2.1.2.1 VPNGW12:FAU_GEN.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 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 The TOE generates audit logs for the events identified in **Error! Reference source not found.**. Section 6.1 of the ST further explains that for each packet filtering rule configured as described in Section 6.5, the administrator can instruct the TOE to log any traffic matching the rule. These logs include the source and destination addresses, TOE interfaces, Transport layer protocol (if applicable), and source/destination ports (if applicable).

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 NdCPP22:FAU_GEN.1.2

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.

2.1.3 User identity association (NDcPP22e:FAU_GEN.2)

2.1.3.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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.

See 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.

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

2.1.4.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

The evaluator shall examine the TSS to ensure it describes whether the TOE is a standalone TOE that stores audit data locally or a distributed TOE that stores audit data locally on each TOE component or a distributed TOE that contains TOE components that cannot store audit data locally on themselves but need to transfer audit data to other TOE components that can store audit data locally. The evaluator shall examine the TSS to ensure that for

distributed TOEs it contains a list of TOE components that store audit data locally. The evaluator shall examine the TSS to ensure that for distributed TOEs that contain components which do not store audit data locally but transmit their generated audit data to other components it contains a mapping between the transmitting and storing TOE components.

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

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 allocates 2GB of space to local audit storage. The TOE maintains 4 separate log types and up to 10 files for each log type, one active file and 9 archive files. The current file for each log type is allowed to reach 50MB in size before it is rotated, when the TOE attempts to write an audit log message to an audit file which would increase the size beyond 50MB then that audit log type is rotated. Audit log rotation will delete the oldest archive file, if archiving the current file will create more than 9 archive files. Each archive file is renamed with a suffix .1.gz through .9.gz, indicating the age of the archive file, 1 being the newest and 9 being the oldest. Then the current audit log is compressed and renamed with the suffix .1.gz. Once rotation is completed, a new empty current file is created and the queued audit message is written to the file. This approach ensures that no audit log messages are discarded during the log rotation process and that the allocated storage capacity of 2GB is never exceeded.

The TOE transmits audit data to an external audit server using the syslog protocol tunneled within IPsec or SSH. The TOE transmits audit records to the syslog server in real-time. The TOE does not have the ability to cache or retransmit audit records if the syslog server is unavailable.

The TOE is not distributed.

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.

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

Section 10.3.1 of the AGD details the configuration of protected syslog via a trusted channel. The TOE supports both audit over SSHC and audit over IPSec. Section 10.2 details local log storage. The TOE maintains 4 separate log types and up to 10 files for each log type, one active file and 9 archive files. The current file for each log type is allowed to reach 50MB in size before it is rotated, when the router attempts to write an audit log message to an audit file which would increase the size beyond 50MB then that audit log type is rotated. Audit log rotation will delete the oldest archive file, if archiving the current file will create more than 9 archive files. Each archive file is renamed with a suffix .1.gz through .9.gz, indicating the age of the archive file, 1 being the newest and 9 being the oldest. Then the current audit log is compressed and renamed with the suffix .1.gz. Once rotation is completed, a new empty current file is created and the queued audit message is written to the file. This approach ensures that no audit log messages are discarded during the log rotation process and that the allocated storage capacity of 2GB is never exceeded.

The CLI does not implement any commands that allow the modification or deletion of audit records. No configuration is needed or possible for local storage of audit data or audit log rotation.

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.

The evaluator also repeated this test for the SSH Client syslog server. The evaluator observed the TOE initiating an SSH 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.

In both instances, the evaluator observed this connection occurred automatically.

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 log file. The TOE's behavior once the local audit storage was filled was consistent with the explanation provided in the TSS.

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 asymmetric key generation using RSA key establishment (key size 2048/3072) , ECC key establishment (curves P-256, P-384, and P-521), FFC key establishment (key size 2048), and FFC Safe Primes key establishment as part of IPsec and SSH. The TOE is a SSH server and client, 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 7.4 of the admin guide states that private key generation offers a variety of options through the use of the command:

pki gen <key-name> <key-type> <key-length>

Where

<key-name> - The key file with extension “key”, “pem”, or “der”. Prefix with “exported-“ to auto export the key.

<key-type> <key-length> - can be either a key type of “rsa” with key length of 2048, 3072, 4096, or 8192, or a key type of “ecdsa” with a key length of 256, 384, or 521. If the key type and key length are not specified, “rsa” with the key length of 4096 are used by default.

Section 7.5 of the admin guide describes key generation. Section 11.4 details the commands used in key generation. Section 8.3.7 describes the configuration of SSH cryptographic algorithms and section 8.4.5 describes the configuration of the VPN cryptographic algorithms.

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. The TSF generates asymmetric RSA/ECDSA/ECDH keys for IKE key exchange and IKE authentication according to FIPS 186-4 Appendices B.3.3, B.4 and B.4.2. The TSF supports >= 2048 bit RSA keys and ECDSA curves P-256, P-384, and P-521 for IKE authentication. The TSF supports curves P-256, P-384 and P-521 for IKE key exchange. The TOE also generates DH keys for IKE key exchange according to FIPS 186-4 Appendices B.1 and B.1.2. The TOE implementation of Diffie-Hellman group 14 (2048 MODP), Diffie-Hellman group 16 (4096 MODP), and Diffie-Hellman group 18 (8192 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 7.4 of the admin guide states that private key generation offers a variety of options through the use of the command:

pki gen <key-name> <key-type> <key-length>

Where

<key-name> - The key file with extension “key”, “pem”, or “der”. Prefix with “exported-“ to auto export the key.

<key-type> <key-length> - can be either a key type of “rsa” with key length of 2048, 3072, 4096, or 8192, or a key type of “ecdsa” with a key length of 256, 384, or 521. If the key type and key length are not specified, “rsa” with the key length of 4096 are used by default.

Section 7.5 of the admin guide describes key generation. Section 11.4 details the commands used in key generation.

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 Establishment (NDcPP22e:FCS_CKM.2)

2.2.3.1 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 asymmetric key generation using RSA key establishment (key size 2048/3072) , ECC key establishment (curves P-256, P-384, and P-521), FFC key establishment (key size 2048), and FFC Safe Primes key establishment as part of IPsec and SSH. The TOE is a SSH server and client, 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	SSH (server)	RSA Schemes ECC Schemes DH-14, DH-16, DH-18
Trusted Channels for Syslog	SSH (client)	RSA Schemes ECC Schemes DH-14, DH-16, DH-18
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 8.3.7 of the admin guide describes the configuration of SSH cryptographic algorithms and section 8.4.5 describes the configuration of the VPN cryptographic algorithms. No additional configuration is needed.

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.

2.2.4 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.2.4.1 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 contains a table that list all of the relevant keys used to support the claimed SFRs. This table describes when and how the relevant keys are destroyed. The TSF destroys keys in RAM by writing a dynamic value to the memory address(es) containing the key being destroyed. The TSF destroys keys in Flash by logically addressing the storage address and overwriting the key with four different static patterns (i.e. 0x00, 0xFF, 0xAA, and 0x55). The TSF performs a read-verify of the 0x55 pattern after the final write. The TSF logs a key destruction error if the read-verify fails.

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 7.6 of the admin guide describes key destruction. The Machete router destroys keys in RAM by writing a dynamic value to the memory address(es) containing the key being destroyed. The TSF destroys keys in Flash by logically addressing the storage address and overwriting the key with four different static patterns (i.e. 0x00, 0xFF, 0xAA, and 0x55). The router performs a read-verify of the 0x55 pattern after the final write. The router logs a key destruction error if the read-verify fails.

Component Testing Assurance Activities: None Defined

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

2.2.5.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.2 of the ST contains Table 4 CAVP Certificates which identifies the following key size(s) and mode(s) for data encryption/decryption:

AES CBC (128 and 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 8.3.7 of the admin guide describes the configuration of SSH cryptographic algorithms and section 8.4.5 describes the configuration of the VPN cryptographic algorithms.

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.

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

2.2.6.1 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

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

2.2.7.1 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 cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512 with message digest sizes 160, 256, 384, and 512. The TOE supports SHA-256 and 384 hashing in support of HMAC and digital signatures, and SHA-512 for HMAC support, digital signatures, password obfuscation, and binary integrity checking.

The TSF supports the following algorithms for ESP:

- Message authentication:
  - HMAC-SHA-256
  - HMAC-SHA-384
  - HMAC-SHA-512

The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh_rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

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 8.3.7 of the admin guide describes the configuration of SSH cryptographic algorithms and section 8.4.5 describes the configuration of the VPN cryptographic algorithms. No additional configuration is needed.

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.

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

2.2.8.1 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 the TOE supports HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 (key and output MAC sizes 160, 256, 384, and 512, respectively) for keyed-hash message authentication. The TOE implements HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 in support of IPsec and HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 in support of SSH.

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 8.3.7 of the admin guide describes the configuration of SSH cryptographic algorithms and section 8.4.5 describes the configuration of the VPN cryptographic algorithms.

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.

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

2.2.9.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

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 the use of RSA with 2048 and 3072 bit key sizes, and ECDSA with a key size of 256 bits or greater for cryptographic signatures (specifically NIST curves P-256, P-384, or P-521).

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 8.1.3 of the admin guide details how to configure the TOE into NIAP mode. Section 8.3.7 describes the configuration of SSH cryptographic algorithms and section 8.4.5 describes the configuration of the VPN cryptographic algorithms. Section 7 details certificate and PKI management.

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.

2.2.10 IPsec Protocol - Per TD0800 (NDcPP22e:FCS_IPSEC_EXT.1)

2.2.10.1 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 when a packet is received by the TSF and addressed to the TOE, the TSF first checks the packet against the INPUT filter rules as described in Section 6.5. Packets that match a “permit” rule are allowed to be processed further by the TOE. Next, the TSF determines if the packet is an IKE packet (UDP port 500 or 4500). If the packet is an IKE packet, the TSF checks the packet against the VPN session establishment rules. If the packet does not pass the VPN session establishment rules, the TSF drops the packet. If the packet passes the checks, the TSF processes the IKE packet.

All other packets must match a “permit” INPUT rule for the TOE to process them.

If the packet is an ESP packet (IPv4 protocol 50/IPv6 Next Header 50), the TSF examines the SPI and compares it to the SPIs associated with established Phase 2/Child SAs. If the TSF finds a matching SPI, it attempts decryption and processes the decrypted packet according to Section 6.5. The TSF drops the packet if it does not find a matching SPI or the decryption fails.

All packets not addressed to the TOE are first examined by the FORWARD filter rules, if the packet matches a “permit” rule, then the TOE continues processing the packet. Next, the TSF determines if the packet matches any of the configured IPsec SPD rules. If the packet does not match any SPD rules, the packet is processed as described in Section 6.5. If the packet matches one or more SPD rules, the TSF selects the most specific rule and applies the configured action (i.e. PROTECT, BYPASS, or DISCARD). The specificity of the rule is determined first by the specificity of the IP address/mask, then by the TCP or UDP port, and lastly by the transport layer protocol. Source and destination addresses and ports are given equal weights, so the administrator is instructed to ensure rules cannot have equal specificity. Packets matching a BYPASS SPD rule are forwarded without modification on the appropriate network interface to reach their destination. If the VPN connection for a packet matching a PROTECT SPD rule is not currently established, the TSF sends an IKE init packet to the peer to attempt to establish a connection and the original packet is dropped; otherwise, the TSF forwards the packet through the established VPN connection. Packets that are forwarded through the VPN connection are subject to the OUTPUT filter rules described in Section 6.5.

All network traffic must match a “permit” OUTPUT rule. Then, locally originating traffic is matched against any of the configured IPsec SPD rules in the same manner as described above for forwarded traffic.

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 8.4.4.3 of the AGD details SPD configuration, which describes setting the “mode” parameter for each of the child configurations. By setting the “mode” to “pass”, packets will bypass the tunnel. Setting it to “drop” will drop the packets. To encrypt traffic for a connection, set the “mode” parameter to “tunnel” or “transport”. The guidance also details a sample configuration with all three options configured. The same format that is used for the default field can be used for each connection and will override the default when there is a difference. This enables an administrator to ensure security at the tunnel policy level while keeping the verbosity of the configuration file low. Additionally, this requires pass/drop policies to have a higher priority (lower numeric value) than tunnel policies, otherwise they will not be applied. To verify these priorities, use the command show vpn policy-db. To manually set the priority, add the “priority” parameter to the child policy and set it to the desired integer. By default, the “priority” is set to 0, which dynamically calculates priorities based on the size of the traffic selectors. Policies must be configured on all peers.

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.

The TOE uses VPN configurations in order to control the flow of traffic. If traffic is destined for an IP address associated with the VPN tunnel, then the traffic will be protected. Otherwise, any traffic matching other defined VPN traffic selectors will be dropped or bypassed from the tunnel as configured. A TOE config was created with rules for bypassing ICMP packets, discarding SSH traffic, and the regular tunnel configuration. The default drop rule was accomplished with a firewall implicit deny.

The evaluator first established the tunnel. The packet capture showed that traffic was encrypted with IPSEC.

The evaluator then sent ICMP packets which were configured with a bypass rule. The result was that the traffic was sent in plaintext and the TOE responded in cleartext.

The evaluator attempted to SSH to the TOE. The packet capture did not show any SSH traffic since the SSH traffic was encrypted in ESP before it is dropped. Thus, the SSH attempt was discarded.

Finally, to test the default drop rule, the evaluator sent HTTP traffic to the TOE and observed that the TOE did not respond.

2.2.10.2 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.

2.2.10.3 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 8.4.4.3 of the admin guide states how to configure tunnel and transport mode.

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.

This test was repeated for both IKE versions.

2.2.10.4 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 TSF supports the following algorithms for ESP:

- Encryption:
  - AES-CBC-128
  - AES-CBC-256
- Message authentication:
  - HMAC-SHA-256
  - HMAC-SHA-384
  - HMAC-SHA-512

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 8.4.5.1 and section 8.4.5.2 of the admin guide states the encryption and integrity algorithms supported.

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 and AES-CBC-256 and verified via logs and packet captures that the connection was successfully established for each algorithm. This was repeated for IKEv1 and IKEv2.

2.2.10.5 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 the TSF implements IKEv1, IKEv2, and ESP to protect IPsec communications. The TSF will not propose aggressive mode when using IKEv1 and rejects any init packet proposing aggressive mode.

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 8.4.4.3 of the admin guide details how to configure the TOE to use IKEv1 and IKEv2.

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.

2.2.10.6 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 of the ST states the TSF supports the following algorithms for IKEv1 and IKEv2:

- Authentication:
  - X.509 Certificate using RSA >= 2048 bits, ECDSA P-256, P-384, or P-521
  - Pre-Shared Key
- Key exchange (Private key "x" is generated as specified in Section 7.2.1):
  - Group 14
  - Group 19
  - Group 20
  - Group 24
- Encryption:
  - AES-CBC-128
  - AES-CBC-256
  - AES-GCM-128 (IKEv2 only)
  - AES-GCM-256 (IKEv2 only)
- Message authentication/PRF:
  - HMAC-SHA-256
  - HMAC-SHA-384
  - HMAC-SHA-512

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

Section 8.4.4.3 of the admin guide details how to configure the TOE to use IKEv1 and IKEv2. Section 8.4.5.3, 8.4.5.4, and 8.4.5.5, 8.4.5.5.2, and 8.4.5.5.3 state the AEAD algorithms, psuedo-random functions, DH groups, mod-p and elliptic curve groups supported.

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 for (AES-CBC-128, AES-CBC-256) and IKEv2 for (AES-CBC-128, AES-CBC-256, AES-GCM-128, and AES-GCM-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. This is consistent with the TSS which states that only IKEv2 supports the GCM algorithms.

2.2.10.7 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 the TSF allows the IKEv1 Phase 1 SA and IKEv2 IKE SA lifetimes to be configured to between 15 minutes and 24 hours.

The TSF allows the IKEv1 Phase 2 SA and IKEv2 Child SA lifetimes to be configured to between 15 minutes and 8 hours and/or 10 million bytes and 4 billion bytes, whichever occurs first.

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. (TD0800 applied)

Section 8.4.4.3 of the admin guide states that the TOE configuration includes options with some default fields included to illustrate the full configuration option. The “tunnel-conn” connection drops SSH traffic from specific addresses, bypasses ICMP traffic, and uses tunnel mode for its child SA of the same name. IKEv1 and IKEv2 can be explicitly configured using the version field in the connection_base object. A value of 1 uses IKEv1 (aka ISAKMP), 2 uses IKEv2. A connection using the default of 0 accepts both IKEv1 and IKEv2 as a responder and initiates the connection actively with IKEv2. The SA lifetime of a connection can be configured by using the “rekey_time” parameter. In addition to time the child SA lifetime can be configured to rekey after transmitting a data limit. This can be configured with the “rekey_bytes” parameter of the child connection with a minimum of 10 million and maximum of 4 billion bytes.

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 lifetime Phase 1 SA on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that a new Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation. (TD0800 applied)

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. The evaluator repeated this for both IKEv1 and IKEv2.

2.2.10.8 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 the TSF allows the IKEv1 Phase 1 SA and IKEv2 IKE SA lifetimes to be configured to between 15 minutes and 24 hours.

The TSF allows the IKEv1 Phase 2 SA and IKEv2 Child SA lifetimes to be configured to between 15 minutes and 8 hours and/or 10 million bytes and 4 billion bytes, whichever occurs first.

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. (TD0800 applied)

Section 8.4.4.3 of the admin guide states that the TOE configuration includes options with some default fields included to illustrate the full configuration option. The “tunnel-conn” connection drops SSH traffic from specific addresses, bypasses ICMP traffic, and uses tunnel mode for its child SA of the same name. IKEv1 and IKEv2 can be explicitly configured using the version field in the connection_base object. A value of 1 uses IKEv1 (aka ISAKMP), 2 uses IKEv2. A connection using the default of 0 accepts both IKEv1 and IKEv2 as a responder and initiates the connection actively with IKEv2. The SA lifetime of a connection can be configured by using the “rekey_time” parameter. In addition to time the child SA lifetime can be configured to rekey after transmitting a data limit. This can be configured with the “rekey_bytes” parameter of the child connection with a minimum of 10 million and maximum of 4 billion bytes.

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. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once a new Phase 2 SA is negotiated when or before 8 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 2 negotiation. (TD0800 applied)

Test 1 - For this test, the evaluator configured the minimum 10MB data rekey limit on the TOE. The evaluator then established an IPsec connection, and sent pings over the tunnel until the 10MB limit was reached. The evaluator confirmed that the rekey occurred before the 10MB data limit was hit. The evaluator repeated this for both IKEv1 and IKEv2.

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

2.2.10.9 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 & 24, 19, and 20, respectively).

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.10.10 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 of the ST states nonces are generated using the FIPS validated RBG specified in FCS_RBG_EXT.1. The TSF generates nonce that are 32 bytes long, which is half the length of the longest PRF hash (SHA-512) and greater than the strength of the strongest Diffie-Hellman group (Group 20).

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.

2.2.10.11 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, 20, and 24 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 8.4.5.5 of the admin guide states the supported DH groups.

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.

2.2.10.12 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 of the ST states the following encryption algorithms are supported for IKEv1/IKEv2:

- AES-CBC-128
- AES-CBC-256

- AES-GCM-128 (IKEv2 only)
- AES-GCM-256 (IKEv2 only)

The following encryption algorithms are supported for ESP:

- AES-CBC-128
- AES-CBC-256

The TSF will not load/enable any IPsec configuration where the ESP encryption algorithm is or can be stronger than the IKE authentication algorithm (e.g. IKE is configured to use AES 128 or 256 while ESP is configured to use AES 256). The TSF will reject any init proposal that specifies an ESP algorithm stronger than the IKE algorithm.

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.

Test 3 - The evaluator attempted to establish a connection with an unsupported algorithm/hash combination. The connection attempt failed.

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

These four test cases were performed with both IKEv1 and IKEv2.

2.2.10.13 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 >= 2048 bits and ECDSA P-256, P-384, or P-521 certificates and pre-shared keys. This is consistent with the algorithms specified in FCS_COP.1/SigGen.

Section 6.3 of the ST states The TOE supports text-based pre-shared keys for IKE PSK authentication. The TOE supports keys composed of any combination of uppercase, lowercase, numbers, and the symbols in the SFR. Text-based pre-shared keys can be anywhere from 8 to 130 characters long. The TSF uses SHA-256 condition text-based pre-shared keys. The TSF allows the administrator to use any of the following methods to create a pre-shared key: generated bit-based and password-based.

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 8.4.4.2 of the admin guide details how to configure IPsec with certificates. Section 8.4.4.1 describes pre-shared key configuration along with the minimum requirements for generating pre-shared keys. Section 7 details certificate and PKI management including how to load a trusted CA, CRL validity checks, and generating CSR's.

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 this SFR demonstrates that a successful IPsec connection using an RSA certificate can be established.

Testing in NDcPP22e_FCS_IPSEC_EXT.1.14-t2 demonstrates that a successful IPsec connection using an ECDSA certificate can be established.

Testing in NDcPP22e_FCS_IPSEC_EXT.1.4-t1 and VPNGW12_FIA_PSK_EXT.1 demonstrates that a successful IPsec connection using pre-shared keys can be established.

2.2.10.14 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 if the certificate is valid, the TSF attempts to authenticate the connection using the ID Payload sent by the peer or ID configured by the Administrator for the specific peer. If the ID Type is ID_IPV4_ADDR, ID_RFC822_ADDR, or ID_IPV6_ADDR; the TSF attempts to match the ID against an appropriate SAN field in the certificate. If the ID Type is ID_DER_ASN1_DN, the TSF matches the ID against the DN in the certificate.

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 7 of the admin guide describes the supported identifiers and any required configuration and all supported certificate validation checks. Section 7.2 details the required certificate authority extensions, private key size, and signature algorithm strength.

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: Not applicable. The TOE does not support CN identifiers.

Test 2: For the first part of this test, the evaluator alternately configured a test peer to use an authentication certificate with the correct SAN: IP address, SAN: DNS, and SAN:User DNS. The evaluator then attempted to

establish an IPsec connection between the TOE and the test peer and confirmed that the connection was successful. CN identifiers are not supported by the TOE so the second part of the test is not applicable.

Test 3: Not applicable. The TOE does not support CN identifier types.

Test 4: For this test, the evaluator alternately configured the TOE to look for each of the supported SAN reference identifiers (SAN:IP Address, SAN: DNS, and SAN:User DNS). The evaluator then configured the test peer to use a certificate that would present an incorrect SAN reference identifier and a correct CN reference identifier as CN checking is not prioritized over SAN (CN is not supported). In each case, the evaluator attempted to establish an IPsec connection to the TOE and then expected the TOE to reject the connection.

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

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

2.2.11.1 VPNGW12:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.2 VPNGW12:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.3 VPNGW12:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.4 VPNGW12:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.5 VPNGW12:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.6 VPNGW12:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.7 VPNGW12:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.8 VPNGW12:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.9 VPNGW12:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.10 VPNGW12:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.11 VPNGW12:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.12 VPNGW12:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.13 VPNGW12:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.11.14 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 6.2 of the ST states IKEv1 authentication using X.509 certificates with RSA or ECDSA keys is performed as specified in RFC 2409 Section 5.1, while authentication using pre-shared keys is performed as specified in RFC 2409 Section 5.4.

IKEv2 authentication using X.509 certificates with RSA or ECDSA keys and authentication using pre-shared keys is performed as specified in RFC 5996 Section 2.15.

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 8.4 details configuration of connections using both PSK and certificates. Section 8.4.4 describes example VPN configurations using both PSK and certificates and any configuration necessary to enable the authentication method.

Component Testing Assurance Activities: None Defined

2.2.12 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1)

2.2.12.1 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 provides the ability to synchronize its time with a NTP server using NTP v4. The time data is protected by an IPsec connection.

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 4.2 of the admin guide states Configuration of the external time source is restricted to NTPv4, hence no configuration is available to set the NTP version. Section 4.2.2 of the admin guide provides an example of configuring multiple NTP servers.

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.

2.2.12.2 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 4.2.2.1 of the admin guide details how to configure the TOE to protect NTP traffic via 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.

2.2.12.3 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 4.2 of the admin guide states that no further configuration is needed to prevent accepting broadcast and multicast NTP packets that would result in the timestamp being updated, as this is the default behavior.

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.

2.2.12.4 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 the same NTP connections as the previous test and observed the current time on the TOE and NTP server. The evaluator set the time on the NTP server ahead by 1 hour. 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 from the untrusted NTP server and did not update its time.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.2.13 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.2.13.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.13.2 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 the TSF implements a NIST SP 800-90A CTR_DRBG with AES-256 for generating random bits. The TSF instantiates the DRBG using 262,144 of data gathered from the RDRAND instruction which is estimated to provide at least 256-bits of entropy.

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.

Section 7.4.2 of the admin guide states that for the random bit generation (RNG Functionality), the CPU RDrand instruction is utilized which uses a hardware thermal entropy source. No configuration is necessary to enable 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.

2.2.14 SSH Client Protocol - per TD0636 (NDcPP22e:FCS_SSHC_EXT.1)

2.2.14.1 NDcPP22e:FCS_SSHC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.14.2 NDcPP22e:FCS_SSHC_EXT.1.2

TSS Assurance Activities: The evaluator shall check to ensure that the TSS contains a list of the public key algorithms that are acceptable for use for user authentication and that this list is consistent with asymmetric key generation algorithms selected in FCS_CKM.1, hashing algorithms selected in FCS_COP.1/Hash, and signature generation algorithms selected in FCS_COP.1/SigGen. The evaluator shall confirm the TSS is unambiguous in declaring the TOE's ability to authenticate itself to a remote endpoint with a user-based public key.

If password-based authentication method has been selected in the FCS_SSHC_EXT.1.2, then the evaluator shall confirm it is also described in the TSS.

Section 6.2 of the ST states the TOE supports SSHv2 as both a client and a server. The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521. Administrators must associate a public key with each user account that is authenticated with public-keys. The TOE's SSHv2 supports both public-key and password-based authentication. Refer to NDcPP22e:FIA_PMG_EXT.1 for more information on the password authentication.

The ssh-rsa public key algorithm is consistent with the supported claims for FCS_COP.1/Hash and FCS_COP.1/SigGen.

Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed mechanisms are used in SSH connections initiated by the TOE.

Section 8.3.5 of the admin guide describes SSH client configuration. Section 8.3.5.2 details automatic connections such that the TOE automatically connects after the service restarts.

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

Test 1: For each claimed public-key authentication method, the evaluator shall configure the TOE 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 server to demonstrate the use of all claimed 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: [Conditional] If password-based authentication method has been selected in the FCS_SSHC_EXT.1.2, then following the guidance documentation the evaluator shall configure the TOE to perform password-based authentication with a remote SSH server to demonstrate that the TOE can successfully authenticate using a password as an authentication method.

Test 1: The evaluator attempted to connect from the TOE to an SSH server using a pubkey. The TOE claims support for RSA and ECDSA public keys. This supported public key algorithm resulted in a successful connection.

Test 2: The evaluator attempted to connect the TOE to an SSH server alternately using the correct and incorrect password. The evaluator found that only the correct password would yield a successful SSH session.

2.2.14.3 NDcPP22e:FCS_SSHC_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 6.3 in the ST states that the maximum packet size accepted by the TOE SSH Server/Client is limited to 35,000 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 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 35,000 bytes.

The evaluator observed that the TOE's SSH client rejected the packet and the connection was closed.

2.2.14.4 NDcPP22e:FCS_SSHC_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 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

This matches the SFR claims.

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 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms.

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 with a remote server (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, aes128-gcm@openssh.com, and aes256-gcm@openssh.com. The evaluator captured packets associated with each of the connection attempts and

observed that using these algorithms the evaluator was able to successfully connect the TOE's SSH client to the SSH test server.

2.2.14.5 NDcPP22e:FCS_SSHC_EXT.1.5

TSS Assurance Activities: The evaluator shall confirm the TSS describes how a host-key public key (i.e., SSH server's public key) is associated with the server identity.

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

If x509v3-based public key authentication algorithms are claimed, the evaluator shall confirm that the TSS includes the description of how the TOE establishes the server's identity and how this identity is confirmed with the one that is presented in the provided certificate. For example, the TOE could verify that a server's configured IP address matches the one presented in the server's x.509v3 certificate.

Section 6.2 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521. For the SSH Client, the SSH server can be associated with its public key locally.

This matches the SFR claims.

x509v3-based public key authentication algorithms are not supported.

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 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms.

Testing Assurance Activities: Test 1: The evaluator shall establish an SSH connection using each of the public key algorithms specified by the requirement to authenticate an SSH server to the TOE. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Test objective: The purpose of this positive test is to check the authentication of the server by the client (when establishing the transport layer connection), and not for checking generation of the authentication message from the client (in the User Authentication Protocol). The evaluator is therefore intended to establish sufficient separate SSH connections (with an appropriately configured server) to cause the TOE to demonstrate use of all public key algorithms claimed in FCS_SSHC_EXT.1.5 in the ST.

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

Test 1: The evaluator attempted to connect the TOE's SSH Client to the SSH server 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 algorithms "ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521" as claimed.

Test 2: The evaluator followed up with a disallowed authentication algorithm and confirmed that it was not accepted.

2.2.14.6 NDcPP22e:FCS_SSHC_EXT.1.6

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

Section 6.2 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256

- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

This matches the SFR claims.

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 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms. It also states that the MAC algorithm “none” is not allowed in the NIAP security mode.

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 server to only allow a MAC algorithm that is not included in the ST selection. The evaluator shall attempt to connect from the TOE to the SSH server 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 the TOE to an SSH server alternately using each of the MAC algorithms that can be claimed to determine which ciphers are supported with successful connections. The evaluator observed that only the claimed algorithms "hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512" (except implicit) were able to successfully connect to the TOE.

Test 2: The evaluator followed up with a disallowed MAC algorithm to ensure it was not accepted (hmac-md5). The TOE rejected a connection attempt using this disallowed MAC algorithm.

2.2.14.7 NDcPP22e:FCS_SSHC_EXT.1.7

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

Section 6.2 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

This matches the SFR claims.

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 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms.

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

Test 1 - The evaluator attempted to connect the TOE to an a SSH server using each of the key exchange algorithms that can be claimed to determine which ciphers are supported with successful connections. The algorithms used were "diffie-hellman-group14-sha1, ecdh-sha2-nistp256, diffie-hellman-group14-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, ecdh-sha2-nistp384, ecdh-sha2-nistp521" and the evaluator confirmed that the connections were successful in each case.

2.2.14.8 NDcPP22e:FCS_SSHC_EXT.1.8

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

1. Both thresholds are checked by the TOE.

2. Rekeying is performed upon reaching the threshold that is hit first.

Section 6.2 in the ST states The TOE initiates a session rekey after the configured rekey limit is reached. This limit is based on time and data, with a rekey triggered whenever either limit is reached. The time-based rekey can be configured by an administrator to values between 1 second and 1 hour. The data-based rekey limit can be configured by an administrator to values between 1 kilobyte and 1 gigabyte of traffic.

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 8.3.6 details the SSH session rekey configuration and enforcement. The router may be configured to enforce SSH session rekeying events either after a time limit, of 1 second up to 1 hour, or a data limit, of 1K up to 1G of data, has been reached.

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 the TOE to connect to an SSH server 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 server the TOE is connected to.

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH server, 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_SSHC_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 server the TOE is connected to.

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 first configured the TOE for lower rekey thresholds. 50KB for data and 5 minutes for time. The evaluator then attempted to connect to the TOE to an SSH server generating 50KB of data and confirmed via logs and packet captures that the rekey happened before the threshold was reached. The evaluator then attempted to connect the TOE to an SSH server waiting the time limit (5 minutes) and confirmed via logs and packet captures that the rekey happened before the threshold was reached.

2.2.14.9 NDcPP22e:FCS_SSHC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: Test 1: The evaluator shall delete all entries in the TOE's list of recognized SSH server host keys and, if selected, all entries in the TOE's list of trusted certification authorities. The evaluator shall initiate a connection from the TOE to an SSH server. The evaluator shall ensure that the TOE either rejects the connection or displays the SSH server's public key (either the key bytes themselves or a hash of the key using any allowed hash algorithm) and prompts the Security Administrator to accept or deny the key before continuing the connection.

Test 2: The evaluator shall add an entry associating a host name with a public key into the TOE's local database. The evaluator shall replace, on the corresponding SSH server, the server's host key with a different host key. If 'password-based' is selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using password-based authentication, shall ensure that the TOE rejects the connection, and shall ensure that the password was not transmitted to the SSH server (for example, by instrumenting the SSH server with a debugging capability to output received passwords). If 'password-based' is not selected for the TOE in FCS_SSHC_EXT.1.2, the evaluator shall initiate a connection from the TOE to the SSH server using public key-based authentication, and shall ensure that the TOE rejects the connection.

Test1: The evaluator first deleted all entries from the TOE's known hosts file (certification authorities are not applicable here). The evaluator then initiated a connection from the TOE to the previously known SSH server and observed that the administrator was prompted to accept or deny the key before continuing the connection. The evaluator chose the former option and demonstrated a successful connection.

Test 2: The evaluator created an entry in the TOE's known hosts file corresponding to the test SSH server's host name. The evaluator then changed the host key on the server. Since password-based is selected, the evaluator initiated a connection from the TOE to the test SSH server with password-based authentication. The evaluator observed that the password was not transmitted to the SSH server.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.2.15.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.15.2 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 of the ST states the TOE supports SSHv2 as both a client and a server. The TOE supports user public key authentication using ssh_rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521. Administrators must associate a public key with each user account that is authenticated with public-keys. The TOE's SSHv2 supports both public-key and password-based authentication. Refer to NDcPP22e:FIA_PMG_EXT.1 for more information on the password authentication.

The ssh-rsa public key algorithm is consistent with the supported claims for FCS_COP.1/Hash and FCS_COP.1/SigGen.

X509 authentication is not supported.

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 partially been performed in FIA_UIA_EXT.1-t1 where the evaluator demonstrated a login using RSA public keys. The evaluator also demonstrated a successful connection using ECDSA pubkeys under this test case.

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.

2.2.15.3 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 6.3 in the ST states that the maximum packet size accepted by the TOE SSH Server/Client is limited to 35,000 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 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 35,000 bytes.

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

2.2.15.4 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 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

This matches the SFR claims.

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 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms.

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, aes128-gcm@openssh.com, and aes256-gcm@openssh.com. 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.

2.2.15.5 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 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256

- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521. Administrators must associate a public key with each user account that is authenticated with public-keys

This matches the SFR claims.

x509v3-based public key authentication algorithms are not supported.

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 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms.

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 algorithms "ssh-rsa, rsa-sha2-256, rsa-sha2-512,

ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521" 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.

2.2.15.6 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 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

This matches the SFR claims.

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).

matches the SFR claims. Section 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms.

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 algorithms "hmac-sha1, hmac-sha1-96, hmac-sha2-256, hmac-sha2-512, implicit" were able to successfully connect to the TOE.

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

2.2.15.7 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 of the ST states The TOE supports SSHv2 with AES (CBC/CTR/GCM modes) 128- or 256-bit ciphers, in conjunction with HMAC-SHA-1, HMAC-SHA-1-96, HMAC-SHA2-256, and HMAC-SHA2-512 integrity algorithms as well as RSA and ECDH using the following key exchange methods:

- diffie-hellman-group14-sha1
- ecdh-sha2-nistp256
- diffie-hellman-group14-sha256
- diffie-hellman-group16-sha512
- diffie-hellmangroup18-sha512
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

The TOE supports user public key authentication using ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, or ecdsa-sha2-nistp521.

This matches the SFR claims.

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 8.3.7 of the admin guide describes the SSH cryptographic algorithm configuration along with the supported algorithms.

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 methods "diffie-hellman-group14-sha1, ecdh-sha2-nistp256, diffie-hellman-group14-sha256, diffie-hellman-group16-sha512, diffie-hellman-group18-sha512, ecdh-sha2-nistp384, ecdh-sha2-nistp521" and confirmed that the connection was successful.

2.2.15.8 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 in the ST states The TOE initiates a session rekey after the configured rekey limit is reached. This limit is based on time and data, with a rekey triggered whenever either limit is reached. The time-based rekey limit can be configured by an administrator to values between 1 second and 1 hour. The data-based rekey limit can be configured by an administrator to values between 1 kilobyte and 1 gigabyte of traffic.

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 8.3.6 details the SSH session rekey configuration and enforcement. The router may be configured to enforce SSH session rekeying events either after a time limit, of 1 second up to 1 hour, or a data limit, of 1K up to 1G of data, has been reached.

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 first configured the TOE for lower rekey thresholds. 1MB for data and 5 minutes for time. The evaluator then attempted to connect to the TOE using an SSH client generating 1MB 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 the time limit (5 minutes) and confirmed via logs and packet captures that the rekey happened before the threshold was reached.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3 IDENTIFICATION AND AUTHENTICATION (FIA)

2.3.1 AUTHENTICATION FAILURE MANAGEMENT (NDCPP22E:FIA_AFL.1)

2.3.1.1 NDCPP22E:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.1.2 NDCPP22E:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.3 of the ST explains that the TOE maintains a counter of successive failed remote authentication attempts associated with each administrator username. The TSF increments this counter when it receives an invalid password. If the counter exceeds the administrator configured threshold (between 1 and 20), the TOE locks the account from logging in remotely, records the lock time, and will not allow any remote authentication attempts to succeed until the administrator configured time period has elapsed. The account remote lock time period can be configured to be 1 to 3600 seconds. While the account is locked, the TOE will not process any remote authentication attempts for the locked account. Any attempt to log in to the remote account before the configured lock period has expired will reset the lock time to the configured time period. The TOE resets the failed authentication counter when the administrator configured time period has elapsed and when a valid password is entered for an unlocked account. Authentication and administration using the local console remains available for accounts that have been locked out from using the remote authentication mechanisms.

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 6.4 of the admin guide details authentication failure management. Accounts which fail to authenticate remotely are locked after a configurable integer between 1 and 20, inclusively, of unsuccessful attempts. Once an account is locked, the account remains locked for a configurable integer between 1 and 3600, inclusively, duration of seconds. By default, accounts have 3 attempts to successfully authenticate before being locked for the duration of 300 seconds (5 minutes).

Any attempt to remotely authenticate a locked account before the account lock time has expired results in a failure and the timer is reset to the full account lock time. The account is not provided any feedback from the system that the account is locked or unlocked.

Each initiation of a remote session will prompt the user for a password 3 times, regardless of the lock attempts configured. For example, if the number of lock attempts is set to 2 and a user initiates a connection, the user will be prompted to enter the password three times, even if authentication failed the first two times resulting in the account already being locked when prompted the third time.
Additionally, any prompt for the password that occurs while the account is locked will result in a failed login attempt, even if the lock time has elapsed and the correct password is provided. For example, if the number of lock attempts is set to 2 and the lock time set to 30 seconds, when a remote session is initiated and authentication fails the first 2 password prompts and the user wait 30 seconds to enter the 3rd attempt, the 3rd attempt will still fail even though the account lock time has elapsed and the correct password is given because the prompt for the password occurred while the account was still locked. Remotely locked accounts can always authenticate through the local console.

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 2 failed authentication attempt limit and 5-minute lockout duration. The evaluator then attempted to connect to a session and verified after 2 failures the session was locked. After waiting the 5-minute lockout period the evaluator was able to reconnect. This test was repeated with the 5 failed authentication attempt limit.

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.

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

2.3.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

(TD0792 applied)

Section 6.3 of the ST states the TSF supports passwords composed of upper case, lowercase, numbers, and the symbols in the SFR. The TOE be configured to enforce a minimum password length of 6 to 100 characters.

The symbols in the SFR are the following: ! " # $ % & ' ( ) * + , - . / : ; < = > @ [\ ] ^ _ ` { | } ~ ? <space>.

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 6.2 of the AGD details password management. Passwords are composed of printable characters which are defined as the following four class types: upper case characters, lower case characters, digits, and the following symbols:
! " # $ % & ' ( ) * + , - . / : ; < = > @ [\ ] ^ _ ` { | } ~ ? <space>
A password must contain at least 6 - 100 characters. The use of configurable credit rules does not reduce the required number of characters below 6.

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.

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

2.3.3.1 VPNGW12:FIA_PSK_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.3.2 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 6.3 of the ST states the TOE supports text-based pre-shared keys for IKE PSK authentication. The TOE supports keys composed of any combination of uppercase, lowercase, numbers, and the symbols in the SFR. Text-based pre-shared keys can be anywhere from 8 to 130 characters long. The TSF uses SHA-256 condition text-based pre-shared keys. The TSF allows the administrator to use any of the following methods to create a pre-shared key: generated bit-based and password-based.

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 8.4 of the admin guide states that for vpn the TOE enforces an 8 to 130 character password length with any uppercase, lowercase, number, or special symbol: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')', '+', '/', '-', '_', '=','?', '<space>. It also provides sample configuration for establishing vpn connection using PSK.

Section 7.5 of the admin guide details PSK generation.

The TOE supports the generation of bit-based and password-based pre-shared keys. Both of these key types may be generated with the use of the pki psk command.

pki psk <pskfile> gen [bit-type] <length>

Where

<pskfile> - File name to store the key; must have an extension of .psk (for VPN) or .wpa2 (for WiFi) or .pass

[bit-type]- Option of “base64” or “hex”

<length> - length of the new PSK in bytes; if bit-type is “base64”, allowed values are:.psk/pass: 4-96, .wpa2: 4-45. If bit-type is “hex”, allowed values are: .psk/pass: 3-64, .wpa2: 4-32

For password based PSK generation, enter only upto the filename of the .psk or .pass file using the command specified above.This will launch a text-editing session where the password can be entered and saved.

For bit-based PSK generation, the commend described above will be used in its entirety to specify the bit type and PSK length to be generated.

The files generated via this command will be stored in the /psks directory in the file system

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.

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

2.3.4.1 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.

Section 6.3 states the TOE can generate bit-based pre-shared keys of size 128 or 256 bits using the evaluated DRBG.

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.

See FIA_PSK_EXT.1.1

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.

The evaluator followed operational guidance to generate two pre-shared keys on each TOE device. The first key was 128 bits and the second was 256 bits. The evaluator confirmed the keys that were generated matched these lengths in each case.

2.3.5 Password-Based Pre-Shared Keys - per TD0771 (VPNGW12:FIA_PSK_EXT.3)

2.3.5.1 VPNGW12:FIA_PSK_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.2 VPNGW12:FIA_PSK_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.3 VPNGW12:FIA_PSK_EXT.3.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.4 VPNGW12:FIA_PSK_EXT.3.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.5 VPNGW12:FIA_PSK_EXT.3.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.6 VPNGW12:FIA_PSK_EXT.3.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.5.7 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 password-based pre-shared keys are used. (TD0771 applied)

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 6.3 states the TOE accepts pre-shared keys for IPsec. It accepts bit-based pre-shared keys and accepts text-based PSKs that are transformed into bit-based PSKs. For IPsec, text-based keys are conditioned by HMAC-SHA-256, with 10,000 iterations, and output cryptographic key sizes 256 that meet the following: NIST SP 800-132.

The TOE supports text-based pre-shared keys for IKE PSK authentication. The TOE supports keys composed of any combination of uppercase, lowercase, numbers, and the symbols in the SFR. Text-based pre-shared keys can be anywhere from 8 to 130 characters long.

The SFR notes the following symbols are supported: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')', '+', '/', '-', '_', '=','?', '<space>.

Component Guidance Assurance Activities: The evaluator shall confirm the operational guidance contains instructions for entering password-based pre-shared keys for each protocol identified in the requirement. 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. (TD0771 applied)

Section 8.4 of the admin guide states that for vpn the TOE enforces an 8 to 130 character password length with any uppercase, lowercase, number, or special symbol: '!', '@', '#', '$', '%', '^', '&', '*', '(', and ')', '+', '/', '-', '_', '=','?', '<space>. It also provides sample configuration for establishing vpn connection using PSK.

Section 7.5 of the admin guide details PSK generation.

The TOE supports the generation of bit-based and password-based pre-shared keys. Both of these key types may be generated with the use of the pki psk command.

pki psk <pskfile> gen [bit-type] <length>

Where

<pskfile> - File name to store the key; must have an extension of .psk (for VPN) or .wpa2 (for WiFi) or .pass

[bit-type]- Option of “base64” or “hex”

<length> - length of the new PSK in bytes; if bit-type is “base64”, allowed values are:.psk/pass: 4-96, .wpa2: 4-45. If bit-type is “hex”, allowed values are: .psk/pass: 3-64, .wpa2: 4-32

For password based PSK generation, enter only upto the filename of the .psk or .pass file using the command specified above.This will launch a text-editing session where the password can be entered and saved.

For bit-based PSK generation, the commend described above will be used in its entirety to specify the bit type and PSK length to be generated.

The files generated via this command will be stored in the /psks directory in the file system

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.

2.3.6 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.3.6.1 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 6.3 of the AGD states that no additional configuration is required to obscure password data.

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.

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

2.3.7.1 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.

See NDcPP22e_FIA_UIA_EXT.1

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

See NDcPP22e_FIA_UIA_EXT.1

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

2.3.8.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 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 6.3 of the ST states the logon process for the local and remote administration is the same. The TOE presents the user with the administrator configured banner and requires the user enter a username and password combination. If the username does not exist, the TOE indicates the authentication attempt failed (without specifically indicating the username does not exist). If the username exists; the TOE reads the password salt associated with the username, concatenates the salt with the provided password, SHA-512 hashes the combined value, and compares the resulting hash to the stored hash. If the hashes match, the TOE authenticates the user. If the hashes do not match, the TOE indicates the authentication attempt failed. The TOE allows administrators to authenticate to the local console by entering a username and password.

Section 6.5 of the ST identifies the remote administration stating the TSF utilizes SSH to provide the trusted path with protection from disclosure and modification for all remote administration sessions. Optionally, IPsec can be used to protect the SSH session.

Section 6.3 goes on to state The TSF does not provide any services at the local console prior to authentication.

The TOE is not distributed.

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 6 describes identification and authentication. This includes creation and management of user accounts, passwords, and privilege levels in order to limit allowed services.

Section 8.3 details SSH configuration for remote authentication.

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

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.

2.3.9 X.509 Certificate Validation (NDcPP22e/VPNGW12:FIA_X509_EXT.1/Rev)

2.3.9.1 NDcPP22e/VPNGW12: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 IPsec negotiation where the certificate was valid. A packet capture was obtained of this IPsec negotiation which shows that the connection was successful. The evaluator revoked certificates in the chain (individually, using CRLs) and attempted the same connection. The attempt after revoking the certificate was not successful.

Test 4 -- The evaluator configured the test server to present a certificate that does not have the CRL signing purpose. The evaluator established a connection between the TOE and a test server such that the TOE receives this certificate and ensured that the connection was not negotiated successfully due to failure of the CRL validation.

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.

2.3.9.2 NDcPP22e/VPNGW12: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 6.3 in the ST states the the TOE performs the following checks when validating certificates:

- All Certificates:
  - Proper encoding
  - Valid signature
  - Current time is after the Not Valid Before value
  - Current time is before the Not Valid After value
  - (if present in the certificate) CRL Check
- CA Certificates
  - Basic Constraints: CA=true
  - Extended Key Usage:
  - Certificate Sign
  - CRL Sign (for CA certificates used to verify a CRL)
- TOE and Peer Certificates
  - Basic Constraints: CA=false

The administrator has the option of choosing whether or not to accept the certificate. The administrator accomplishes this by setting the certificate revocation policy in the router configuration.

The TOE does not support the Code Signing, Server Authentication, Client Authentication or OCSP signing extended key usage fields.

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 8.4.4.2 of the admin guide states that the TOE automatically performs certificate validation for VPN connections with certificate authentication by default. This behavior can be configured in the VPN::Modules::charon-systemd::plugins::revocation object. The revocation parameter is also configurable for a desired location, local or remote of a configured connection

(VPN::Profiles::connections::r3::remote::revocation, in the example below). Certificate revocation policy for CRL revocation can be one of the following string options.

• strict - Fails if no revocation information is available, i.e. the certificate is not known to be unrevoked

• ifuri - Fails only if a CRL URI is available but certificate revocation checking fails, i.e. there should be revocation information available but it could not be obtained

• relaxed - Fails only if a certificate is revoked, i.e. it is explicitly known that it is bad

Component Testing Assurance Activities: None Defined

2.3.10 X.509 Certificate Authentication (NDcPP22e/VPNGW12:FIA_X509_EXT.2)

2.3.10.1 NDcPP22e/VPNGW12:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.10.2 NDcPP22e/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: 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 6.3 of the ST states the TOE validates Certificates when they are loaded onto the TOE, when they are used to authenticate the TOE to a peer, and when they are used to authenticate a peer to the TOE. If any of the checks fail (with the possible exception of the CRL check), the TSF rejects the certificate. The administrator configures the certificate to authenticate the TOE to a peer while configuring IPsec. The TOE uses the certificate presented by a peer to attempt to authenticate the peer.

The administrator has the option of choosing whether or not to accept the certificate. The administrator accomplishes this by setting the certificate revocation policy in the router configuration.

See the Guidance Assurance Activity below which addresses how the certificate to be used is configured.

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 3.3 of the AGD describes the import and export of certificates and keys.

Section 7.3 of the AGD details generating certificates and the associated keys.

Section 8.4 of the AGD contains the VPN configuration details including sample configuration using certificates previously imported.

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.

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

2.3.11.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.2 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 7.4.2 of the admin guide details generation of CSRs and requesting certificates from a CA. Section 7.4.2.1.1 of the admin guide states that when generating a CSR the Common Name, Organization, and Country can be edited as well as additional CSR 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.

2.4 Security management (FMT)

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

2.4.1.1 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 10.2 of the admin guide details local audit storage behavior. The Machete router maintains 4 separate log types and up to 10 files for each log type, one active file and 9 archive files. The current file for each log type is allowed to reach 50MB in size before it is rotated, when the router attempts to write an audit log message to an audit file which would increase the size beyond 50MB then that audit log type is rotated. Audit log rotation will delete the oldest archive file, if archiving the current file will create more than 9 archive files. Each archive file is renamed with a suffix .1.gz through .9.gz, indicating the age of the archive file, 1 being the newest and 9 being the oldest. Then the current audit log is compressed and renamed with the suffix .1.gz. Once rotation is completed, a new empty current file is created and the queued audit message is written to the file. This approach ensures that no audit log messages are discarded during the log rotation process and that the allocated storage capacity of 2GB is never exceeded.

The Machete router CLI does not implement any commands that allow the modification or deletion of audit records. No configuration is needed or possible for local storage of audit data or audit log rotation.

Section 10.3 of the admin guide details external audit storage behavior. The Machete router supports the transmission of audit event data to an external audit server using the syslog protocol, RFC 3164. The router transmits audit records to the syslog server in real-time, i.e. when an audit event is generated, is it simultaneously sent to both the external server and the local store. The router does not have the ability to cache or retransmit audit records if the syslog server is unavailable. If the connection to the syslog server is lost, audit logs continue to be stored locally on the router. When the connection is restored, the router automatically resumes sending new audit log messages, requiring no further action from the operator.

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 '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 with 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: Not applicable, as the ST claims [determine the behaviour of] the functions [transmission of audit data to an external IT entity] to Security Administrators.

Test 2: Not applicable, as the ST claims [determine the behaviour of] the functions [transmission of audit data to an external IT entity] to Security Administrators.

Test 3: The evaluator demonstrated that no settings or commands are available to an administrator or user prior to login in FIA_UIA_EXT.1 where it was found that there are no settings or commands available to an administrator to modify any functions at all prior to login.

Test 4: This test was performed in conjunction with FAU_STG_EXT.1. Determining the audit export configuration can be performed at the same time as configuration. The evaluator configured audit export as an administrator and was able to successfully determine the behavior of transmission of audit data in FAU_STG_EXT.1.

2.4.2 MANAGEMENT OF SECURITY FUNCTIONS BEHAVIOUR (NDCPP22E:FMT_MOF.1/MANUALUPDATE)

2.4.2.1 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.

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 3.4 details installing and updating the firmware including all steps necessary to perform manual updates.

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.

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

2.4.3.1 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 states only the authorized administrator can start and stop services on the TOE. This is performed via a CLI ‘services’ command. This includes the firewall, NTP, SSH server, and VPN services.

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 1.4.2 of the admin guide states the following services along with how to manage (start/stop/restart) any applicable services:

The Machete router provides a number of services. The services available include:

• ares - Manage ARES service (is expected to always be running)

• dhcp-server – Manage DHCP server service

v4 – Manage DHCPv4 server service

v6 – Manage DHCPv6 server service

• fake-hwclock - Control fake hardware clock (for systems with no real-time clock)

• firewall - Manage firewall service

• frr - Manage FRR service

• ipsec - Manage IPsec service

• lldp - Manage LLDP service

• ntpd - Manage NTP service

• sshd - Manage SSHD service

• vpn - Manage VPN service

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

2.4.4 MANAGEMENT OF TSF DATA (NDCPP22E:FMT_MTD.1/COREDATA)

2.4.4.1 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 6.4 in the ST states the TOE requires administrators to be successfully identified and authenticated prior to allowing the administrator to manage the TOE.

Section 6.3 in the ST states that the administrator configures the certificate to authenticate the TOE to a peer while configuring IPsec. The TOE also allows the administrator to generate certificate signing requests. The CSRs include the public key, common name, organization, and country. These fall under management functions and therefore require administrators to be successfully identified and authenticated as noted above.

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.

Section 6.5 of the admin guide details that prior to authentication the only unauthenticated services provided by the TOE are icmp echo and Display of the local or remote login banner.

Section 7 details certificate and PKI management. Section 3.3 describes the PKI trust store and how to import to it.

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.

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

2.4.5.1 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 5. This lists all cryptographic keys, 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 7.4.1 of the admin guide details generating private keys, section 7.4 also details generation of asymmetric keys and exporting public keys, and section 6.1.3 describes the configuration and generation of SSH keys.

Section 11.4 details key generation, key deletion, key export and key import with example commands.

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 (presented in t1) 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.

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

2.4.6.1 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_MTD.1/CryptoKeys.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.4.7.1 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 6.4 in the ST identifies all management functions consistent with those specified in FMT_SMF.1. All management functionality can be performed locally or remotely.

Section 2 in the Admin Guide states the operational environment includes the Machete router, one or more VPN connections to provide secure administrative services and protected communications. A trusted channel connection is used to provide a secure connection for communications between the Machete router and remote administrative services; such as Syslog and NTP. SSHv2 is used to provide remote login and Trusted Uploads (SFTP). A connection between the router and a VPN gateway is referred to as a Data VPN Channel and can be used for protected communications other than remote administrative services. Local administration is provided by a local terminal connected to the router through a RS-232 connection.

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

Section 2.4.2 details that the TOE can be accessed locally through the RS-232 port or remotely via SSHv2.

The Machete router command-line interface (CLI) can be accessed from a remote PC by either a serial console connection or by an SSH connection. When using the serial connection, set the parameters to 115200 baud, asynchronous mode with 8 data bits, no parity bit, and 1 stop bit, and with HW and SW flow control disabled.

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.

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

2.4.8.1 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.

See NDcPP22e:FMT_SMF.1

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.

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

2.4.9.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.9.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.9.3 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 6.4 in the ST explains the TOE maintains the security role of Security Administrator who can manage the TOE both remotely and locally. The TOE requires administrators to be successfully identified and authenticated prior to allowing the administrator to manage the TOE.

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.

Section 6.3 of the admin guide details local and remote administrative connections. The Machete Router can be administered both locally and remotely through CLI. Console access is always available for authorized local administrative users through an RS-232 connection. To establish a remote administrative session, SSH is required. No additional configuration is required to obscure password data.

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.

2.5 Packet Filtering (FPF)

2.5.1 Packet Filtering Rules - per TD0683 (VPNGW12:FPF_RUL_EXT.1)

2.5.1.1 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 6.5 of the ST provides a description of the TOE's startup and packet processing. Prior to enabling the kernel networking stack, the TOE assigns a drop rule to each interface, so no network traffic can be processed. As described below, the TOE does not perform packet switching in hardware, so the kernel networking stack is the only method of processing or forwarding packets. Following initialization, the TOE adjusts the rules assigned to each interface to enable the processing, routing, and filtering of network traffic. The TOE implements packet filtering and forwarding in the kernel, so any failure of the packet filtering rules is a kernel networking stack crash which prevents the TOE from processing any network traffic.

Guidance Assurance Activities: The operational guidance associated with this requirement is assessed in the subsequent test EAs.

This is addressed in the subsequent test EAs

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.

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.

2.5.1.2 VPNGW12:FPF_RUL_EXT.1.2

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

2.5.1.3 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

2.5.1.4 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 6.5 in the ST states The TOE supports the following protocols:

- IPv4 (RFC 791)
- IPv6 (RFC 2460)
- TCP (RFC 793)
- UDP (RFC 768)

The correctness of the TOE's implementation of these protocols is demonstrated through interoperability testing with other devices and network services.

The TSF supports packet filtering based on the following fields within each protocol:

- IPv4 (RFC 791)
  - Source address
  - Destination Address
  - Protocol
- IPv6 (RFC 2460)
  - Source address
  - Destination Address
  - Next Header (Protocol)
- TCP (RFC 793)
  - Source Port
  - Destination Port
- UDP (RFC 768)
  - Source Port
  - Destination Port

Each packet filtering rule is configured as a permit (pass packets), deny (drop packets silently), reject (drop packets and issue and ICMP response), or log rule. The TOE applies the packet filtering rules in the order they are configured by the administrator. If a packet does not match any of the configured rules, the TOE drops and logs the packet without sending an ICMP response.

The TOE allows packet filter rules to be associated with one or more interfaces. The TOE does not define any interface groups; however, the TOE allows wildcards to be used when specifying the interfaces, a rule applies to. INPUT and OUTPUT rules can be assigned to physical or virtual interfaces. FORWARD rules allow packets to be routed from one interface to another.

With the exception of IKE and ESP packets, all received packets destined for the TOE (based on IP address) are first subjected to the INPUT rules associated with the receiving interface. All routed packets are subjected to the FORWARD rules to determine if the packet should be passed on. All packets generated locally by the TOE are subjected to the OUTPUT rules associated with the sending interface.

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 9 of the admin guide details the firewall configuration. The Machete router supports packet filtering for the following 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

Each packet filtering rule is configured as a permit (pass packets), deny (drop packets silently), reject (drop packets and issue and ICMP response), or log rule. The Machete router applies the packet filtering rules in the order they are configured by the administrator. If a packet does not match any of the configured rules, the Machete router drops and logs the packet without sending an ICMP response.
The Machete router allows packet filter rules to be associated with one or more interfaces. The Machete router does not define any interface groups; however, the Machete router allows wildcards to be used when specifying the interfaces, a rule applies to. INPUT and OUTPUT rules can be assigned to physical or virtual interfaces. FORWARD rules allow packets to be routed from one interface to another.
With the exception of IKE and ESP packets, all received packets destined for the Machete router (based on IP address) are first subjected to the INPUT rules associated with the receiving interface. All routed packets are subjected to the FORWARD rules to determine if the packet should be passed on. All packets generated locally by the Machete router are subjected to the OUTPUT rules associated with the sending interface.

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).

2.5.1.5 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 6.2 of the ST states when a packet is received by the TSF and addressed to the TOE, the TSF first checks the packet against the INPUT filter rules as described in Section 6.5. Packets that match a “permit” rule are allowed to be processed further by the TOE. Next, the TSF determines if the packet is an IKE packet (UDP port 500 or 4500). If the packet is an IKE packet, the TSF checks the packet against the VPN session establishment rules. If the packet does not pass the VPN session establishment rules, the TSF drops the packet. If the packet passes the checks, the TSF processes the IKE packet.

All other packets must match a “permit” INPUT rule for the TOE to process them.

If the packet is an ESP packet (IPv4 protocol 50/IPv6 Next Header 50), the TSF examines the SPI and compares it to the SPIs associated with established Phase 2/Child SAs. If the TSF finds a matching SPI, it attempts decryption and processes the decrypted packet according to Section 6.5. The TSF drops the packet if it does not find a matching SPI or the decryption fails.

All packets not addressed to the TOE are first examined by the FORWARD filter rules, if the packet matches a “permit” rule, then the TOE continues processing the packet. Next, the TSF determines if the packet matches any of the configured IPsec SPD rules. If the packet does not match any SPD rules, the packet is processed as described in Section 6.5. If the packet matches one or more SPD rules, the TSF selects the most specific rule and applies the configured action (i.e. PROTECT, BYPASS, or DISCARD). The specificity of the rule is determined first by the specificity of the IP address/mask, then by the TCP or UDP port, and lastly by the transport layer protocol. Source and destination addresses and ports are given equal weights, so the administrator is instructed to ensure rules cannot have equal specificity. Packets matching a BYPASS SPD rule are forwarded without modification on the appropriate

network interface to reach their destination. If the VPN connection for a packet matching a PROTECT SPD rule is not currently established, the TSF sends an IKE init packet to the peer to attempt to establish a connection and the original packet is dropped; otherwise, the TSF forwards the packet through the established VPN connection. Packets that are forwarded through the VPN connection are subject to the OUTPUT filter rules described in Section 6.5.

Section 6.5 of the ST states Each packet filtering rule is configured as a permit (pass packets), deny (drop packets silently), reject (drop packets and issue and ICMP response), or log rule. The TOE applies the packet filtering rules in the order they are configured by the administrator. If a packet does not match any of the configured rules, the TOE drops and logs the packet without sending an ICMP response.

The TOE allows packet filter rules to be associated with one or more interfaces. The TOE does not define any interface groups; however, the TOE allows wildcards to be used when specifying the interfaces a rule applies to. INPUT and OUTPUT rules can be assigned to physical or virtual interfaces. FORWARD rules allow packets to be routed from one interface to another.

With the exception of IKE and ESP packets, all received packets destined for the TOE (based on IP address) are first subjected to the INPUT rules associated with the receiving interface. All routed packets are subjected to the FORWARD rules to determine if the packet should be passed on. All packets generated locally by the TOE are subjected to the OUTPUT rules associated with the sending interface.

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 9.2.3.1 of the admin guide details rule configuration and rule ordering. The rule priority is detailed along with all configurable options for processing packets.

The possible chain types are:

- filter: Used to filter packets. This is supported by the arp, bridge, ip, ip6 and inet table families.
- route: Used to reroute packets if any relevant IP header field or the packet mark is modified. This is supported by the ip, ip6 and inet table families.
- nat: Used to perform Networking Address Translation (NAT). Only the first packet of a given flow hits this chain; subsequent packets bypass it. Therefore, never use this chain for filtering. The nat chain type is supported by the ip, ip6 and inet table families.

The possible hooks that you can use when you configure your base chain are:

- ingress: Sees packets immediately after they are passed up from the NIC driver, before even prerouting.
- prerouting: Sees all incoming packets, before any routing decision has been made. Packets may be addressed to the local or remote systems.
- input: Sees incoming packets that are addressed to and have now been routed to the local system and processes running there.

- forward: Sees incoming packets that are not addressed to the local system.

- output: Sees packets that originated from processes in the local machine.
- postrouting: Sees all packets after routing, just before they leave the local system.

Each nftables base chain is assigned a priority that defines its ordering among other base chains, flowtables, and Netfilter internal operations at the same hook. For example, a chain on the prerouting hook with priority -300 will be placed before connection tracking operations. The priority may be a positive or negative integer, or it may use a keyword name for the default priority for that chain type.

NOTE: If a packet is accepted and there is another chain, bearing the same hook type and with a later priority, then the packet will subsequently traverse this other chain. Hence, an accept verdict - be it by way of a rule or the default chain policy – is not necessarily final. However, the same is not true of packets that are subjected to a drop verdict. Instead, drops take immediate effect, with no further rules or chains being evaluated.

Each nftables base chain has a policy. This is the default verdict that will be applied to packets reaching the end of the chain (i.e, no more rules to be evaluated against).

Currently there are 2 policies: accept (default) or drop.

The accept verdict means that the packet will keep traversing the network stack (default).

The drop verdict means that the packet is discarded if the packet reaches the end of the base chain.

NOTE: If no policy is explicitly selected, the default policy accept will be used.

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.

2.5.1.6 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 6.5 of the ST states the TSF supports packet filtering based on the following fields within each protocol:

- IPv4 (RFC 791)
  - Source address
  - Destination Address
  - Protocol
- IPv6 (RFC 2460)
  - Source address
  - Destination Address
  - Next Header (Protocol)
- TCP (RFC 793)
  - Source Port
  - Destination Port
- UDP (RFC 768)
  - Source Port
  - Destination Port

Each packet filtering rule is configured as a permit (pass packets), deny (drop packets silently), reject (drop packets and issue and ICMP response), or log rule. The TOE applies the packet filtering rules in the order they are configured by the administrator. If a packet does not match any of the configured rules, the TOE drops and logs the packet without sending an ICMP response.

There are no unsupported IPv4/IPv6 protocols noted

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 9.2.3.1 of the admin guide states that there are 2 policies: accept (default) or drop.

The accept verdict means that the packet will keep traversing the network stack (default).

The drop verdict means that the packet is discarded if the packet reaches the end of the base chain.

NOTE: If no policy is explicitly selected, the default policy accept will be used.

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

2.6 Protection of the TSF (FPT)

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

2.6.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.1.2 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 6.6 of the ST states that the TOE stores administrator passwords in a non-plaintext form by hashing the password using SHA-512 prior to storage. The CLI does not implement any commands that allow the administrator to view the hashed passwords.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.2 Failure with Preservation of Secure State (Self-Test Failures) (VPNGW12:FPT_FLS.1/SelfTest)

2.6.2.1 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 6.6 of the ST states the TOE implements several self-protection mechanisms.

During power-up, the TSF runs the following self-tests to verify the correct operation of the TSF:

- Entropy Source Health Tests
  - KAT for the deterministic portion of the entropy source
  - Sliding window test of the first 65536 debiased bits from the entropy source
- RAM Tests on a random 64 MB sample of User space RAM:
  - Random Value – detect bad bits, which are permanently stuck
  - Sequential Increment – detect bad bits, which do not consistently hold a value
  - Walking Ones – detect bad bits, which are dependent on the current values of surrounding bits
  - Stuck Address – determine if the memory locations are addressed properly
- Firmware Integrity Test

  - ECDSA P-521 with SHA-512 signature verification of the firmware manifest and SHA-512 verification of each binary
- Cryptographic Library Tests
  - ECDSA P-224 Pairwise Consistency Test
  - AES GCM 256 Encrypt and Decrypt KATs
  - HMAC-SHA-256 KAT (implicitly tests SHA-256)
  - HMAC-SHA-384 KAT (implicitly tests SHA-384)
  - HMAC-SHA-512 KAT (implicitly tests SHA-512)
  - DRBG

If any self-testing generates a failure, the TOE immediately fails-secure by shutting down.

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 4.1 of the admin guide details that if any of the power-on self-tests fail, the router aborts the normal boot process and boots into a restricted diagnostic mode with no networking services. In this mode, the administrator can only log into the router via the serial connection with limited CLI capabilities including:

- view the self-test results to determine the cause of the error
- reboot/shutdown

Entropy health tests may fail if the entropy source is not producing quality entropy. This may be resolved by rebooting the router. RAM health test failure as well may be resolved by rebooting the router. If the self-test error persists then contact ATCorp's customer support.

Component Testing Assurance Activities: None Defined

2.6.3 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT_SKP_EXT.1)

2.6.3.1 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 6.6 of the ST states that the CLI does not implement any commands that allow the administrator to view pre-shared keys, symmetric keys, and private keys. Table 5 in Section 6.2 of the ST provides a detailed listing of where each key is stored. The TOE stores all secret and private cryptographic keys in plaintext.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.4 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT_STM_EXT.1)

2.6.4.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.4.2 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 6.6 of the ST states the TSF contains a real-time clock. The administrator has the option to either set the time on the clock manually or configure an external NTP server that can be queried periodically to update the clock to ensure that it is accurate and reliable. The TSF utilizes the time for the following security functions:

- Administrative session timeout checking
- Administrative remote authentication lockout timer
- Certificate expiration checking
- Phase 1/IKE SA rekey/expiration interval

- Phase 2/Child SA rekey/expiration interval
- Audit record timestamps
- VPN session inactivity checking
- VPN session establishment checking
- Log rotation

The TOE does not obtain time from the underlying virtualization system.

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 4.2 of the admin guide describes configuration of time and contains sample configurations of both manual and NTP configuration.

To set the time manually, enter configuration mode and use the CLI command:
admin@r0(config)> time [MMDDhhmm[[CC]YY][.ss]]
where:
MM = Month (01, 02…12)
DD = Day of month (01, 02…31)
hh = hour
mm = minute
CCYY = Year
ss = second

To synchronize time with external NTP servers or peers, sources can be listed as part of the ARES configuration after first enabling the SysConfig plugin.

The NTP service only synchronizes the local clock if the local clock time is within 1000 seconds of the reference time. If the time difference is greater than 1000 seconds, the NTP service exits and logs the error. If the time difference is less than 1000 seconds, the NTP service steps the local clock until it matches the reference time and creates a log message.

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 - Not applicable, as the TOE does not obtain its time from an underlying VS.

2.6.5 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.6.5.1 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 6.6 of the ST states the TOE implements several self-protection mechanisms.

During power-up, the TSF runs the following self-tests to verify the correct operation of the TSF:

- Entropy Source Health Tests
  - KAT for the deterministic portion of the entropy source
  - Sliding window test of the first 65536 debiased bits from the entropy source
- RAM Tests on a random 64 MB sample of User space RAM:
  - Random Value – detect bad bits, which are permanently stuck
  - Sequential Increment – detect bad bits, which do not consistently hold a value
  - Walking Ones – detect bad bits, which are dependent on the current values of surrounding bits
  - Stuck Address – determine if the memory locations are addressed properly
- Firmware Integrity Test
  - ECDSA P-521 with SHA-512 signature verification of the firmware manifest and SHA-512 verification of each binary
- Cryptographic Library Tests
  - ECDSA P-224 Pairwise Consistency Test
  - AES GCM 256 Encrypt and Decrypt KATs
  - HMAC-SHA-256 KAT (implicitly tests SHA-256)
  - HMAC-SHA-384 KAT (implicitly tests SHA-384)
  - HMAC-SHA-512 KAT (implicitly tests SHA-512)
  - DRBG

If any self-testing generates a failure, the TOE immediately fails-secure by shutting down.

The TOE is not distributed.

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 4.1 of the admin guide details self-tests.

The router runs a suite of the following power-on self-tests to ensure correct operation:

- Entropy source health tests
- RAM health tests
- Firmware integrity test
- Cryptographic algorithm tests

If any of the power-on self-tests fail, the router aborts the normal boot process and boots into a restricted diagnostic mode with no networking services. In this mode, the administrator can only log into the router via the serial connection with limited CLI capabilities including:

- view the self-test results to determine the cause of the error
- reboot/shutdown

Entropy health tests may fail if the entropy source is not producing quality entropy. This may be resolved by rebooting the router. RAM health test failure as well may be resolved by rebooting the router. If the self-test error persists then contact ATCorp's customer support.

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 Power on self-tests during boot. The evaluator booted the TOE and reviewed the results of the self-tests in the console messages.

2.6.6 TSF Testing (VPNGW12:FPT_TST_EXT.1)

2.6.6.1 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

2.6.7 Self-Test with Defined Methods (VPNGW12:FPT_TST_EXT.3)

2.6.7.1 VPNGW12:FPT_TST_EXT.3.1

TSS Assurance Activities: None Defined **Guidance Assurance Activities**: None Defined **Testing Assurance Activities**: None Defined

2.6.7.2 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 6.6 of the ST states the TOE implements several self-protection mechanisms.

During power-up, the TSF runs the following self-tests to verify the correct operation of the TSF:

- Entropy Source Health Tests
  - KAT for the deterministic portion of the entropy source
  - Sliding window test of the first 65536 debiased bits from the entropy source
- RAM Tests on a random 64 MB sample of User space RAM:
  - Random Value – detect bad bits, which are permanently stuck
  - Sequential Increment – detect bad bits, which do not consistently hold a value
  - Walking Ones – detect bad bits, which are dependent on the current values of surrounding bits
  - Stuck Address – determine if the memory locations are addressed properly
- Firmware Integrity Test
  - ECDSA P-521 with SHA-512 signature verification of the firmware manifest and SHA-512 verification of each binary
- Cryptographic Library Tests
  - ECDSA P-224 Pairwise Consistency Test

- AES GCM 256 Encrypt and Decrypt KATs
- HMAC-SHA-256 KAT (implicitly tests SHA-256)
- HMAC-SHA-384 KAT (implicitly tests SHA-384)
- HMAC-SHA-512 KAT (implicitly tests SHA-512)
- DRBG

If any self-testing generates a failure, the TOE immediately fails-secure by shutting down.

This is consistent with what is described in the SFR.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.8 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.6.8.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.3 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 6.6 of the ST states the TSF allows the administrator to query the version of TOE firmware by running the “show version” command.

The TSF supports a manual update process initiated by the administrator. First the administrator must obtain a candidate update from ATCorp on optical media. Then the administrator transfers the update to the TOE, and the TOE verifies the ECDSA P-521 with SHA-512 signature of the update using an ATCorp public key. If the verification fails, the TOE stops the update process and deletes update. If the verification succeeds; the TOE places the update in a staging area, updates the manifest using SHA-512 and the FW Integrity ECDSA Key, and immediately restarts.

The TOE does not support automatic updates or the automatic checking of updates.

The TOE is not distributed.

The TOE does not support published hash.

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 3.5 of the admin guide details the commands for upgrading the TOE firmware including the commands for displaying the current version, uploading new firmware images, displaying current images that are uploaded but not installed, and installing and verifying the image.

The router indicates success or failure of the installation of the new firmware. Loss of power or interruption of the firmware update process puts the Machete router in an unusable state. Any attempt to update the firmware generates a syslog message indicating success or failure of the update.

Should the router fail to process a firmware update for any reason, the failure and reason for failure is recorded in syslog. Successful processing of the firmware update is also recorded in syslog.

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.

2.6.9 Trusted Update (VPNGW12:FPT_TUD_EXT.1)

2.6.9.1 VPNGW12:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.9.2 VPNGW12:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.9.3 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

2.7 TOE ACCESS (FTA)

2.7.1 TSF-INITIATED TERMINATION (NDCPP22E:FTA_SSL.3)

2.7.1.1 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 6.7 of the ST states The TOE terminates remote administrative sessions after an administrator configured period of inactivity. The inactivity period can be configured to be 1 to 3600 seconds with a default of 300 seconds.

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 5.1.2 describes the remote inactivity timeout configuration. The timeout value is specified in seconds by default or may use any of the time units denoted by appending the first letter; either lower-case or uppercase ‘s’ for seconds, ‘m’ for minutes, ‘h’ for hours, ‘d’ for days, or ‘w’ for weeks. Removing this parameter or specifying a zero value disables the inactivity timeout.

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 for each administrative method.

2.7.2 TSF-Initiated Termination (VPN Headend) - per TD0656 (VPNGW12:FTA_SSL.3/VPN)

2.7.2.1 VPNGW12:FTA_SSL.3.1/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 verify that it describes the ability of the TSF to terminate an inactive VPN client session.

Section 6.7 of the ST states the TOE monitors the client VPN activity within the Phase 2/Child SA. If no packets are sent for an administrator configured period of time, the TOE terminates the client VPN. The inactivity period can be configured to be 5 – 480 minutes.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it provides instructions to the administrator on how to configure the time limit for termination of an active VPN client session.

Section 8.4.4.3 of the admin guide states that child connections can be configured to be terminated after a period of inactivity between 5-480 minutes. This can be configured by adding the “inactivity” parameter to the child connection.

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

Test 1: The evaluator shall follow the steps provided in the operational guidance to set the inactivity timer for five minutes. The evaluator shall then connect a VPN client to the TOE, let it sit idle for four minutes and fifty seconds, and observe that the VPN client is still connected at this time by performing an action that would require VPN access. The evaluator shall then disconnect the client, reconnect it, wait five minutes and ten seconds, attempt the same action, and observe that it does not succeed. The evaluator shall then verify using audit log data that the VPN client session lasted for exactly five minutes.

Test 2: The evaluator shall configure the inactivity timer to ten minutes and repeat Test 1, adjusting the waiting periods and expected audit log data accordingly.

Test 1: The evaluator configured the inactivity timer for five minutes, then connected a VPN client to the TOE and waited for four minutes and fifty seconds. The evaluator then sent a ping from the TOE to the VPN client and verified the session did not close. The evaluator then established a connection and let the idle timeout expire and verified that the TOE terminated the session as expected. The VPN client log showed that the connection was terminated after 5 minutes of inactivity. The evaluator verified using audit log data that the VPN client session

lasted five minutes.

Test 2: The evaluator repeated Test 1 with a time period of 10 minutes (adjusting the waiting periods and audits accordingly).

2.7.3 User-initiated Termination (NDcPP22e:FTA_SSL.4)

2.7.3.1 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 6.7 of the ST states the TOE provides the function to logout (or terminate) both local and remote user sessions as directed by the user.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Section 5.2 of the admin guide details the administrator-initiated termination using the command logout. This applies to both local and remote sessions.

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 issued the exit command, 'logout' 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 issued the exit command, 'logout' per the command identified in the Guidance. 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.

2.7.4 TSF-initiated Session Locking (NDcPP22e:FTA_SSL_EXT.1)

2.7.4.1 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 6.7 of the ST states the TOE locks local administrative sessions after an administrator configured period of inactivity. The inactivity period can be configured to be 1 to 3600 seconds with a default of 300 seconds. The TSF blanks the local console when it locks the session and requires the administrator to re-authenticate before allowing administrative access.

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 6.4 of the admin guide describes authentication failure management including the configurable session locking duration. By default accounts have 3 attempts to successfully authenticate before being locked for the duration of 300 seconds. Once an account is locked, the account remains locked for a configurable integer between 1 and 3600, inclusively, duration of seconds. Remotely locked accounts can always authenticate through the local console.

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 1,3, and 5 minutes. The evaluator then performed a login from the console and no further activity until the session was automatically closed by the TOE.

2.7.5 Default TOE Access Banners (NDcPP22e:FTA_TAB.1)

2.7.5.1 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 6.7 of the ST states whether connecting to the CLI locally or remotely (SSH), 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 5.3 of the AG details how to configure the banner message.

Component Testing Assurance Activities: The evaluator shall also perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure a notice and consent warning message. The evaluator shall then, for each method of access specified in the TSS, establish a session with the TOE. The evaluator shall verify that the notice and consent warning message is displayed in each instance.

Test 1: The evaluator configured the TOE for login banners and verified with each method of access that the banner was displayed.

2.7.6 TOE Session Establishment - per TD0656 (VPNGW12:FTA_TSE.1)

2.7.6.1 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 6.7 of the ST states for client VPN connections, the TOE authenticates the connection if the session meets the session establishment rules by verifying (as configured):

- The client IP address is on the allowed list
- Connections are allowed during the current time of day
- Connections are allowed on the current date
- Connections are allowed during the current day of week
- Connections are allowed during the current day of month

If one of the session establishment rules above is not met the connection is rejected.

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 9.3 of the admin guide describes the deny session establishment configuration.

The following configuration should be added to the input chain before all of the accept statements:

iifname <interface> <day|hour|time> drop

Where:

<interface> is the name of the interface that should drop packets.

<day|hour> can be either the word “day” followed by the day of the week, or “hour” followed by the range of hours to drop packets. Note that the “meta” keyword is required after the interface for the time specifier.

The following example will deny session establishment on the weekend and throughout the week between the hours 6:00PM and 7:00AM for the eth0 interface and also reject packets between the dates “2026-01-01 00:00:00” and “2038-01-19 03:14:07”

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 the current day. The evaluator then attempted to reconnect using the same VPN client, and observed that it was not successful.

Test 3: 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 the current time. The evaluator then attempted to reconnect using the same VPN client, and observed that it was not successful.

Test 4: No other attributes are identified by the requirement; therefore, this conditional test is not applicable.

2.7.7 VPN Client Management - per TD0656 (VPNGW12:FTA_VCM_EXT.1)

2.7.7.1 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 6.7 of the ST states if the client VPN connection is authenticated and meets the session establishment rules, the TSF assigns the client a private IP address out of an administrator configured pool.

Component Guidance Assurance Activities: There are no operational guidance EAs for this component.

No guidance activities

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.

2.8 Trusted path/channels (FTP)

2.8.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP_ITC.1)

2.8.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.3 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 6.8 of the ST states the TOE uses IPsec or SSH when exporting audit records to a third-party syslog server and IPsec when communicating with an NTP server. The TOE also uses IPsec when communicating with IPsec clients and peers.

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 10.3.1 of the admin guide details protected syslog via trusted channel. Section 4.2.2.1 describes configuration for NTP protection via IPsec.

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 NTP and syslog over the tunneled IPsec network. The also repeated this test configured the TOE for syslog over transmitted over SSHC.

Test 2 - This was tested through the FCS_IPSEC_EXT.1 / FCS_SSHC_EXT.1 tests as well as test 4 below. The TOE can be configured to connect as a client to a peer and will then automatically 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 (IPsec & SSH) and/or NTP server (IPsec only). 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 IPsec MAC layer timeout, the evaluator physically disrupted the connection for about 30 seconds. For the IPsec APP layer timeout, the evaluator physically disrupted the connection for about 3 minutes.

For the SSH MAC layer timeout, the evaluator physically disrupted the connection for about 15 seconds. For the SSH APP layer timeout, the evaluator physically disrupted the connection for about 4 minutes.

2.8.2 Inter-TSF Trusted Channel (VPN Communications) (VPNGW12:FTP_ITC.1/VPN)

2.8.2.1 VPNGW12:FTP_ITC.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.2 VPNGW12:FTP_ITC.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.3 VPNGW12:FTP_ITC.1.3/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

2.8.3 Trusted Path - per TD0639 (NDcPP22e:FTP_TRP.1/Admin)

2.8.3.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.3.3 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 6.8 of the ST states the TOE uses SSH and optionally IPsec to provide a trusted path for remote management interfaces to protect the communication from disclosure and modification. 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 3 of the AGD details that The Machete router supports either SSH or IPsec to provide a trusted communication channel between itself and all authorized IT entities that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data. The router uses SSH or IPsec to provide the trusted path with remote administrative users as well.

Section 8.4 Describes IPsec configuration and section 8.3 describes SSH configuration.

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 also demonstrated that the management SSH traffic could be tunnel through an IPsec connection. This connection was also successful and the traffic was encrypted.

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 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 1.4 of the admin guide describes the operational environment, states the administrator must ensure the guidance items in the admin guide are enforced, and identifies all services outside of the common criteria evaluation. Section 1.4.2 identifies all services available in the evaluated configuration. Section 2 of the admin guide details the evaluated configuration.

SSH algorithm configuration is detailed in section 8.3.7. IPsec cryptographic algorithm configuration is described in section 8.4.5.

Section 3.5 details the install/update firmware process including acquiring images and how digital signatures are verified.

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 evaluator had the Admin Guide to use when configuring the TOE. The completeness of the Admin Guide is addressed by its 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 Admin Guide 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 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:

- MACHETE-FIT2 running Ares 2.0 (CRR) –Referred to as “CRR”
- MACHETE-V1 running Ares 2.0 (Machete-V) – Referred to as “Machete-V”

Figure 1 of the DTR depicts the standard test network used during evaluation testing. The figure shows two Devices Under Test (DUTs), which represents the location of the TOEs in the network.

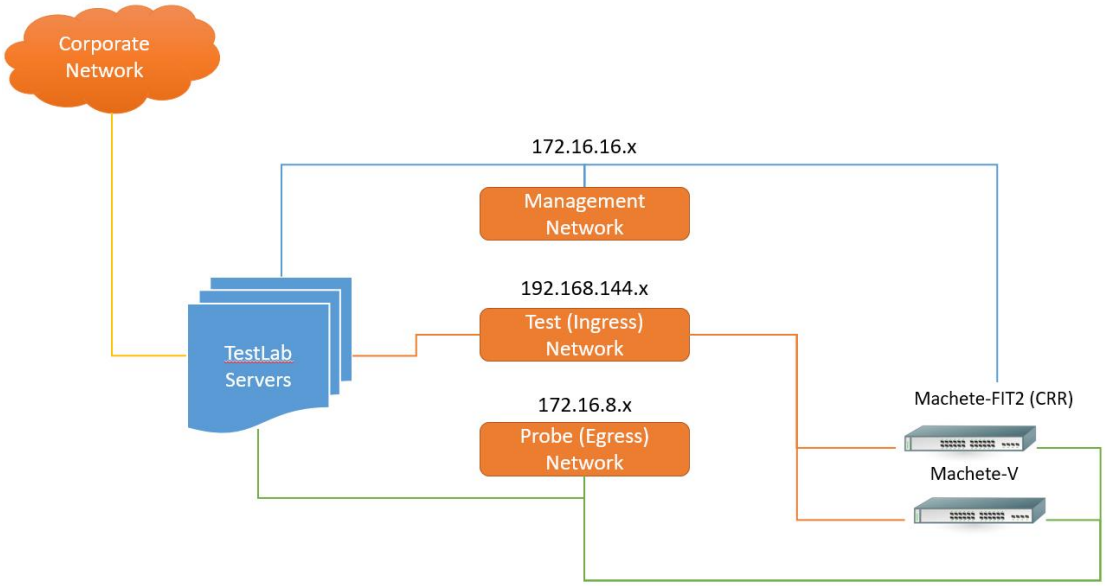
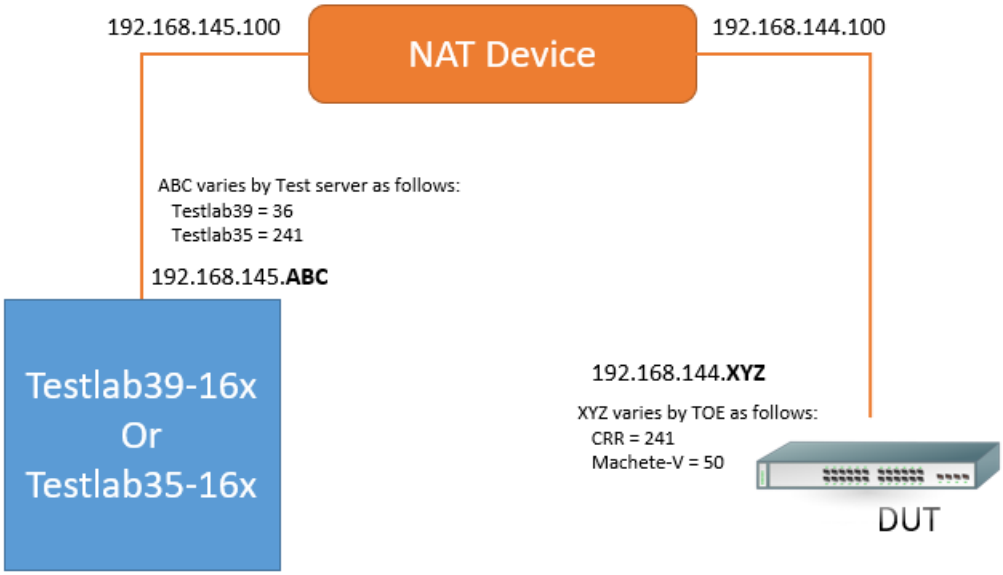


Figure 2 of the DTR depicts the test network used during evaluation testing for NAT.



The Gossamer Test servers utilized both a windows and Ubuntu environment. The Windows supporting software included the following.

- 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 included the following tools.

- Openssh version 7.2p2
- Openssh version 8.9p1 (for SSH key exchange testing)
- Big Packet Putty version 6.2
- Nmap version 7.01
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Openssl version 1.0.2g
- Strongswan version 5.3.5
- Rsyslog version 8.16.0

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 fuss testing. None of the public search for vulnerabilities or the fuzz testing uncovered any residual vulnerability.

The evaluator searched the following sources for vulnerabilities on 1/30/24

- 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 Engin (http://www.exploitsearch.net),
- Offensive Security Exploit Database (https://www.exploit-db.com/)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search),

Each site was searched using the following terms: "Architecture Technology Corporation", "Machete Router ", "ATCorp", "ARES", "OpenSSL", "Intel Atom x7-E3950", "AMD Ryzen V1605B", "Intel Core i5-8365U", "Intel Atom x6425E", "Ryzen 4600G", "TCP".

3.5.2 Additional Flaw Hypothesis (AVA_VAN.1)

Assurance Activities: The following additional tests shall be performed:1.) [Conditional]: If the TOE is a TLS server and supports ciphersuites that use RSA transport (e.g. supporting TLS_RSA_WITH_* ciphers) the following test shall be performed. Where RSA Key Establishment schemes are claimed and especially when PKCS#1 v1.5* padding is used, the evaluators shall test for implementation flaws allowing Bleichenbacher and Klima et al. style attacks, including Bock et al's ROBOT attacks of 2017 in the flaw analysis. Even though Bleichenbacher's original paper is two decades old, Bock et al. found these attacks to still be effective in weakening the security of RSA key establishment in current products. Bleichenbacher and Klima et al. style attacks are complex and may be difficult to detect, but a number of software testing tools have been created to assist in that process. The iTC strongly recommends that at least one of the tools mentioned in Bock et al's ROBOT attacks of 2017 webpage or paper, as effective to detect padding oracle attacks, be used to test TOE communications channels using RSA based Key Establishment (related sources: http://archiv.infsec.ethz.ch/education/fs08/secsem/bleichenbacher98.pdf, https://eprint.iacr.org/2003/052, https://robotattack.org/). Network Device Equivalency Consideration.

The TOE does not support a TLS server implementation and therefore is not subject to Bleichenbacher attacks.