Assurance Activity Report

Junos 22.2R1 for MX10003

Document Reference: EFT-T024-AAR 1.3
Document Status: Released
Release Date: 31-Jan-2024
Customer: Juniper Networks
98 pages including cover.

TERON LABS

Copyright Notice

Teron Labs AISEF

Unit 3, 10 Geils Court
Deakin, ACT 2600
Australia

+61 2 5114 4878
info@teronlabs.com
www.teronlabs.com

Table of Contents

1 Document Management

Version	Date	Author	Description
1.1	27-Sep-2023	Anantha Kandiah	Updates based on ACA review.
1.2	19-Oct-2023	Anantha Kandiah	Updates based on ACA review.
1.3	31-Jan-2024	Anantha Kandiah	Updates based on NIAP review.

2 References

2.1 Evaluation Requirements

[1] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, Version 3.1, Revision 5

[2] Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components, Version 3.1, Revision 5

[3] Common Criteria for Information Technology Security Evaluation Part 3: Security assurance components, Version 3.1 Revision 5

[4] Common Methodology for Information Technology Security Evaluation, Evaluation methodology, Version 3.1, Revision 5

[5] Collaborative Protection Profile for Network Devices (NDcPP), Version 2.2e, 23-Mar-2020

[6] Supporting Document, Evaluation Activities for Network Device cPP, Version 2.2, Dec-2019

[7] Network Device Collaborative Protection Profile (NDcPP) Extended Package MACsec Ethernet Encryption, Version 1.2, May-2016

2.2 Evaluation Evidence

[8] Security Target Junos OS 22.2R1 for MX10003, Version 1.3, 10-Oct-2023.

[9] Test Report: NDcPP (EFT-T024-TR-NDcPP), Version 1.1, 13-Dec-2022

[10] Test Report: MACsec (EFT-T024-TR-MACSEC), Version 1.1, 13-Dec-2022

[11] Junos OS Common Criteria Configuration Guide for MX10003 Devices, Date 18-Sep-2022

[12] MX10003 Universal Routing Platform Hardware Guide, 01-March-2023

3 Introduction

This report documents the assurance activities performed by Teron Labs as part of the Common Criteria evaluation of Junos 22.2R1 for MX10003 developed by Juniper Networks. The product was evaluated against the requirements of the following protection profiles:

[NDcPP] Collaborative Protection Profile for Network Devices, version 2.2e dated 23 March 2020

[MACsec] Network Device Collaborative Protection Profile (NDcPP) Extended Package MACsec Ethernet Encryption, version 1.2 dated 10 May 2016

The above requirements were revised as per the Technical Decisions (TDs) listed in the Section 5.1

Based on the results of these activities, Teron Labs determined that Junos 22.2R1 for MX10003 and supporting evidence documentation passes all requirements of the above listed protection profiles.

3.1 Evaluation Identifiers

Task Identifier	EFT-T024
TOE Name	Junos 22.2R1 for MX10003
TOE Version	22.2R1
Sponsor	Juniper Networks, Inc. 1133 Innovation Way, Sunnyvale California 94089 United States
Developer	Juniper Networks, Inc. 1133 Innovation Way, Sunnyvale California 94089 United States
Evaluation Facility	Teron Labs Unit 3, 10 Geils Court, Deakin, ACT 2600, Australia
Scheme	Australian Information Security Evaluation Program (AISEP)
PP(s)	Collaborative Protection Profile for Network Devices (NDcPP), Version 2.2e, 23-Mar-2020.s Network Device Collaborative Protection Profile (NDcPP) Extended Package MACsec Ethernet Encryption, Version 1.2, 10-May-2016.
CC Version	3.1 Revision 5

3.2 ST Identifier

ST Title	Security Target Junos 22.2R1 for MX10003
ST Version	1.3
ST Date	10-Oct-2023

3.3 TOE Overview

The Target of Evaluation (TOE) is Juniper Networks, Inc. Junos OS 22.2R1 executing on MX-Series 3D Universal Edge Router MX10003 with MACsec Line Cards. The supported next generation Routing Engines is:

- RE-S-1600x8 for MX10003

The line card containing the MACsec module, required for deployment in the TOE, is:

- JNP-MIC1-MACSEC in the MX10003

The MX10003 appliances is a secure network device that protect itself largely by offering only a minimal logical interface to the network and attached nodes. It is powered by the Junos OS firmware, Junos OS 22.2R1, which is a special purpose OS that provides no general-purpose computing capability. Junos OS provides both management and control functions as well as all IP routing.

The appliance primarily supports the definition of, and enforce, information flow policies among network nodes. All information flow from one network node to another passes through an instance of the TOE. Information flow is controlled on the basis of network node addresses and protocol. In support of the information flow security functions, the TOE ensures that security-relevant activity is audited and provides the tools to manage all of the security functions.

The portfolio of MX Series 3D Universal Edge Routers includes a wide range of physical and virtual platforms that share a common architecture and feature set. Juniper Networks MX10003 routing appliance is a complete routing system that supports a variety of high-speed interfaces (only Ethernet is within scope of the evaluation) for medium/large networks and network applications. Juniper Networks MX routers share common Junos firmware, features, and technology for compatibility across platforms.

The appliance is physically self-contained, housing the firmware and hardware necessary to perform all routing functions. The architecture components of the appliance are:

- Switch fabric – the switch fabric boards/modules provide a highly scalable, non-blocking, centralized switch fabric matrix through which all network data passes.
- Routing Engine (Control Board) – the Routine Engine (RE) runs the Junos firmware and provides Layer 3 routing services and Layer 2 switching services. The RE also provides network management for all operations necessary for the configuration and operation of the TOE and controls the flow of information through the TOE, including support for appliance interface control and control plane functions such as chassis component, system management and user access to the appliance.

- The Packet Forwarding Engine (PFE) – provides all operations necessary for transit packet forwarding. This is provided by the Modular Port Concentrators on the MX-Series appliances.
- Power – power supply bays to provide complete flexibility for provisioning and redundancy. The power supplies connect to the midplane, which distributes the different output voltages produced by the power supplies to the appliance components, depending on their voltage requirements.

The RE and PFE perform their primary tasks independently, while constantly communicating through a high-speed internal link. This arrangement provides streamlined forwarding and routing control and the capability to run Internet-scale networks at high speeds.

The appliance supports numerous routing and switching standards for flexibility and scalability.

The functions of the appliance can all be managed through the Junos firmware, either from a connected terminal console or via a network connection. Network management is secured using the SSH protocol. All management, whether from a user connecting to a terminal or from the network, requires successful authentication. In the evaluated deployment the TOE is managed and configured via Command Line Interface, either via a directly connected console or over the network secured using the SSH protocol.

The MACsec line card support MACsec between adjacent devices, all traffic communicated between the devices including frames for LLDP, DHCP, ARP, STP, Ethernet Control frames, etc (the exceptions to this protection are Destination MAC and Source MAC addresses in MACsec and MKA frames).

MACsec can be deployed in point-to-point mode or shared mode with multiple stations. In the evaluated configuration MACsec must be configured individually on each point-to-point Ethernet link, such that a pair of MACsec devices (connected by a physical medium) protect Ethernet frames switched or routed from one device to the other. The two MACsec devices are provided with a Connectivity Association Key (CAK) and utilize the MACsec Key Agreement (MKA) protocol to create a secure tunnel. MKA is used by the two MACsec devices to agree upon MACsec keys. MACsec must be configured to protect all traffic between the devices, with the exception of the MKA or Ethernet control traffic such as EAP over LAN (EAPOL) frames. The devices will first exchange MKA frames, which serve to determine the peer is an authorized peer, and agree upon a shared key and MACsec cipher suite used to set up a transmit (Tx) Security Association (SA) and a receive (Rx) SA. Once the SAs are set up, MACsec-protected frames traverse the unprotected link.

3.3.1 Physical boundary

The TOE is the Junos OS 22.2R1 firmware running on the appliance chassis listed in Table 2. Hence, the TOE is contained within the physical boundary of the specified appliance chassis, as shown in Figure 1. The physical boundary of the TOE is the entire chassis of the appliance (see Table 1 - TOE Hardware and Firmware Information below). The evaluated configuration supports JNP-MIC1-MACSEC line card for MX10003.

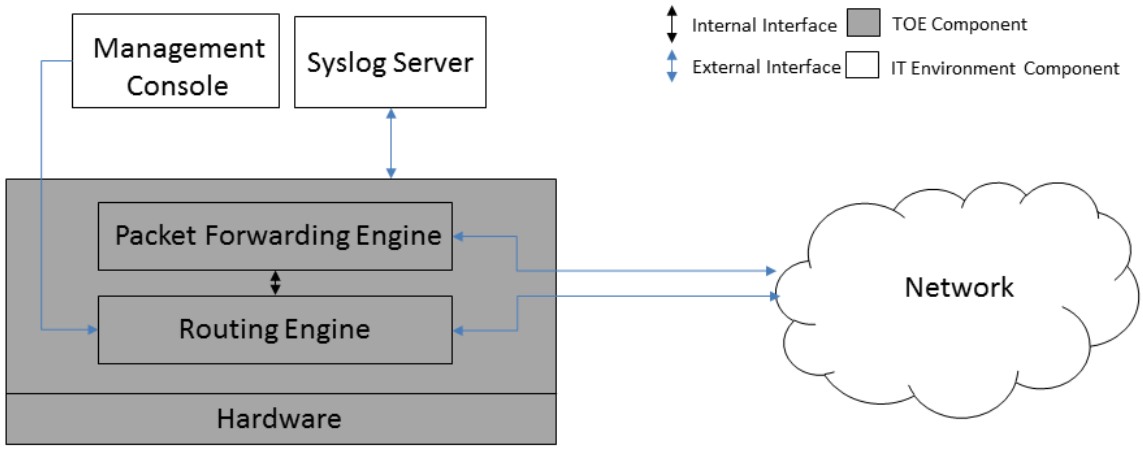

Figure 1 – MX10003 TOE Boundary

The TOE interfaces are comprised of the following:

- Network interfaces which pass traffic
- Management interface through which handle administrative actions.

Model	Routing Engine	Processor	Firmware
MX10003	RE-S-1600x8	Intel Xeon E5-2608L	Junos OS 22.2R1

Table 1 - TOE Hardware and Firmware Information

The MX-series appliance supports numerous combinations and permutations of line cards in the network ports. The interface options supported for each MX series routing appliance are described in the hardware guidance document (Ref. [12]).

Separate jinstall images are provided for MX-series, namely:

- MX10003: junos-vmhost-install-mx-x86-64-22.2R1.9.tgz

The firmware version reflects the detail reported for the components of the Junos OS when the “show version” command is executed on the appliance.

The guidance documents included as part of the TOE are:

> [ECG] Junos OS Common Criteria Configuration Guide for MX10003 Devices, Release 22.2R1

4 Cryptographic Algorithm Testing

The TOE implements cryptographic functions in a set of software crypto libraries, as detailed in the ST. The cryptographic algorithms implemented in these libraries and used by the TSF have been tested by the evaluators as per the cryptographic testing requirements specified in NDcPP-SD (Ref. [6]) and MACsec EP (Ref. [7]). The evaluators used a CAVS-like tool for the generation of test vectors and verification of responses. The table below lists the cryptographic algorithms tested along with the supported SFRs and a brief description of their usage.

Algorithm	SFR	Usage
AES CBC, CTR (128 and 256 bits)	FCS_COP.1/DataEncryption	Encryption/Decryption for SSH
AES CMAC (128 and 256 bits)	FCS_COP.1(1)/KeyedHashCMAC	MACsec key derivation
AES KW (128 and 256 bits)	FCS_COP.1(5)	MACsec Key Agreement (MKA)
RSA Digital Signature Algorithm (rDSA) (2048 bits)	FCS_COP.1/SigGen	SSH Authentication (signature generation and verification)
EC Digital Signature Algorithm (ECDSA) (P-256, P384, P-521)	FCS_COP.1/SigGen	SSH Authentication (signature generation and verification) Trusted update Signature verification
SHA-1, SHA2-256, SHA2-384, SHA2-512	FCS_COP.1/Hash	SSH Hashing
SHA2-256	FCS_COP.1/Hash	Primitive for DRBG
HMAC-SHA2-1, HMAC-SHA2-256, HMAC-SHA2-512]	FCS_COP.1/KeyedHash	SSH Keyed Hashing
HMAC-SHA1, HMAC-SHA2-256	FCS_COP.1/KeyedHash	Cryptographic hashing for password conditioning, password hashing, and self-testing (verifying integrity of system files)
HMAC-SHA2-256	FCS_COP.1/KeyedHash	Primitive for DRBG
HMAC_DRBG with SW-based noise with 256 bits min-entropy	FCS_RBG_EXT.1	Random bit generation with HMAC-DRBG, HMAC-SHA2-256
RSA Key Generation (2048 bits)	FCS_CKM.1	RSA Key Generation for SSH
ECC Key Generation (P-256, P-384, P-521)	FCS_CKM.1	ECC Key Generation for SSH KAS
FFC DH Group 14 key generation	FCS_CKM.1	DH Group 14 key generation for SSH KAS

Algorithm	SFR	Usage
ECC KAS (P-256, P-384, P-521)	FCS_CKM.2	SSH KAS
FFC KAS (DH Group 14)	FCS_CKM,2	SSH KAS

In addition, the TOE uses Broadcom chip BCM82391 for encrypting MACsec traffic using AES-GCM. This hardware implementation of AES-GCM was validated independently under the CAVP and was assigned the certificate number shown in the following table.

Algorithm	SFR	Usage
AES GCM (128 and 256 bits) (CAVP Cert. AES 4545)	FCS_COP1(5)	Encryption/Decryption MACsec

5 Functional Requirements Assurance Activities

This section describes the evaluation activities defined in the following references regarding TOE summary specification (TSS), guidance and functional testing requirements.

[NDcPP_SD] Supporting Document, Evaluation Activities for Network Device cPP, December 2019, version 2.2

[MACsec] Network Device Collaborative Protection Profile (NDcPP) Extended Package MACsec Ethernet Encryption, version 1.2 dated 10 May 2016

The requirements are taken verbatim from the above sources and revised as per any applicable NIAP Technical Decisions listed in Section 5.1. Each requirement is formatted within a coloured box and is followed by the corresponding evaluation findings. Note that only evaluation activities applicable to the TOE are included. Specifically, as the TOE is not a distributed system, evaluation activities defined in the cPP supporting documents for distributed systems are omitted.

5.1 Technical Decisions

The following technical decisions (TDs) are applicable to the evaluated TOE.

ITEM	TITLE	REFERENCE	PUBLICATION DATE	Relevant to ST
TD0738*	NIT Technical Decision for Link to Allowed-With List	Chapter 2	2023.05.19	No
TD0670*	NIT Technical Decision for Mutual and Non-Mutual Auth TLSC Testing	FCS_TLSC_EXT.2.1	2022.09.16	No
TD0652*	MACsec CAK Lifetime in FMT_SMF.1	FMT_SMF.1	2022.08.31	Yes
TD0639*	NIT Technical Decision for Clarification for NTP MAC Keys	FCS_NTP_EXT.1.2, FAU_GEN.1, FCS_CKM.4, FPT_SKP_EXT.1	2022.08.26	No
TD0638*	NIT Technical Decision for Key Pair Generation for Authentication	FCS_CKM.1	2022.08.05	Yes
TD0636*	NIT Technical Decision for Clarification of Public Key User Authentication for SSH	FCS_SSHC_EXT.1	2022.03.21	No
TD0635*	NIT Technical Decision for TLS Server and Key Agreement Parameters	FCS_TLSS_EXT.1.3	2022.03.21	No
TD0634*	NIT Technical Decision for Clarification required for testing IPv6s	FCS_DTLSC_EXT.1.2, FCS_TLSC_EXT.1.2	2022.03.21	No

ITEM	TITLE	REFERENCE	PUBLICATION DATE	Relevant to ST
TD0633*	NIT Technical Decision for Ipsec IKE/SA Lifetimes Tolerance	FCS_IPSEC_EXT.1.7, FCS_IPSEC_EXT.1.8	2022.03.21	No
TD0632*	NIT Technical Decision for Consistency with Time Data for vNDs	FPT_STM_EXT.1.2	2022.03.21	No
TD0631*	NIT Technical Decision for Clarification of public key authentication for SSH Server	FCS_SSHS_EXT.1, FMT_SMF.1	2022.03.21	Yes
TD0618*	MACsec Key Agreement and conditional support for group CAK	FCS_MKA_EXT.1.2, FCS_MKA_EXT.1.5, FCS_MKA.1.8	2022.02.07	Yes
TD0592*	NIT Technical Decision for Local Storage of Audit Records	FAU_STG	2021.05.21	Yes
TD0591*	NIT Technical Decision for Virtual TOEs and hypervisors	A.LIMITED_FUNCTIONALITY, ACRONYMS	2021.05.21	No
TD0581*	NIT Technical Decision for Elliptic curve-based key establishment and NIST SP 800-56Arev3	FCS_CKM.2	2021.04.09	Yes
TD0580*	NIT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers	FCS_CKM.1.1, FCS_CKM.2.1	2021.04.09	Yes
TD0572*	NIT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers	FTP_ITC.1	2021.01.29	Yes
TD0571*	NIT Technical Decision for Guidance on how to handle FIA_AFL.1	FIA_UAU.1, FIA_PMG_EXT.1	2021.01.29	Yes
TD0570*	NIT Technical Decision for Clarification about FIA_AFL.1	FIA_AFL.1	2021.01.29	Yes
TD0569*	NIT Technical Decision for Session ID Usage Conflict in FCS_DTLSS_EXT.1.7	FCS_DTLSS_EXT.1.7, FCS_TLSS_EXT.1.4	2021.01.28	No
TD0564*	NIT Technical Decision for Vulnerability Analysis Search Criteria	AVA_VAN.1	2021.01.28	Yes
TD0563*	NIT Technical Decision for Clarification of audit date information	FAU_GEN.1.2	2021.01.28	Yes

ITEM	TITLE	REFERENCE	PUBLICATION DATE	Relevant to ST
TD0556*	NIT Technical Decision for RFC 5077 question	FCS_TLSS_EXT.1.4, Test 3	2020.11.06	No
TD0555*	NIT Technical Decision for RFC Reference incorrect in TLSS Test	FCS_TLSS_EXT.1.4, Test 3	2020.11.06	No
TD0553*	FCS_MACSEC_EXT.1.4 and MAC control frames	FCS_MACSEC_EXT.1.4	2020.12.18	Yes
TD0547	NIT Technical Decision for Clarification on developer disclosure of AVA_VAN	AVA_VAN.1	2020.10.15	Yes
TD0546*	NIT Technical Decision for DTLS – clarification of Application Note 63	FCS_DTLSC_EXT.1.1	2020.10.15	No
TD0538	NIT Technical Decision for Outdated link to allowed-with list	Section 2	2020.07.13	Yes
TD0537*	NIT Technical Decision for Incorrect reference to FCS_TLSC_EXT.2.3	FIA_X509_EXT.2.2	2020.07.13	No
TD0536	NIT Technical Decision for Update Verification Inconsistency	AGD_OPE.1	2020.07.13	Yes
TD0528	NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4	FCS_NTP_EXT.1.4	2020.07.13	No
TD0527*	Updates to Certificate Revocation Testing (FIA_X509_EXT.1)	FIA_X509_EXT.1/REV, FIA_X509_EXT.1/ITT	2020.07.01	Yes
TD0512*	Group CAKs for establishing multiple MKA connections is not mandated	FMT_SMF.1	2020.03.26	No
TD0509*	Correction to MACsec Audit	FAU_GEN.1	2020.03.02	Yes
TD0487*	Correction to Typo in FCS_MACSEC_EXT.4	FCS_MACSEC_EXT.4.4	2020.01.02	Yes
TD0466*	Selectable Key Sizes for AES Data Encryption/Decryption	FCS_COP.1.1	2019.11.15	Yes
TD0273*	Rekey after CAK expiration	FCS_MACSEC_EXT.4	2017.12.20	Yes
TD0190*	FPT_FLS.1(2)/SelfTest Failure with Preservation of Secure State and Modular Network Devices	FPT_FLS.1(2)/SelfTest	2017.04.11	Yes

ITEM	TITLE	REFERENCE	PUBLICATION DATE	Relevant to ST
TD0135*	SNMP in NDcPP MACsec EP v1.2	FMT_SNMP_EXT.1.1 , FCS_SNMP_EXT.1.1	2017.01.25	No

Table 2 Applicable NIAP Technical Decisions

5.2 Security Audit (FAU)

5.2.1 **FAU_GEN.1 Audit data generation**

TSS

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 (Ref. [6]).

Section 7.1 of the ST describes the details that are recorded for all administrative actions relating to cryptographic keys, including authentication keys generated by PKID, as well as ephemeral keys generated by SSH protocols. SSH host keys are referenced in the logs by the certificate id they are related to. For SSH ephemeral session keys, the PID is used as the key reference to relate the key generation and key destruction audit events. SSH keys generated for outbound trusted channels are identified by the public key filename and fingerprint. SSH keys imported for use in establishing outbound trusted channels are referenced in the log by the hash of the key imported and the username importing.

Guidance Documentation

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 examined the provided operational guidance (Ref. [11]) and determined that it lists all auditable events and provides a format for audit records.

All audit event types mandated by the cPP are described and the description of the field contains the required information as per FAU_GEN.1.2.

Table 7 in Chapter 7 of the guidance (Ref. [11]) describes each of the fields in the event logs. These include:

- Timestamp;
- Hostname;
- Process;
- Process ID;
- TAG;
- Username; and

- Message Text.

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 (Ref. [6]).

The evaluator has examined the provided operational guidance (Ref. [11]) and determined that it provides for sufficient details to implement the mechanisms in the TOE that are necessary to enforce the requirements specified in the cPP.

The Configuration Guide (Ref. [11]) provides the CLI commands and configuration examples necessary to place the device into its evaluated configuration and to enforce the requirements specified in the Security Target (Ref. [8]). The evaluator has found that the guide provides the necessary information for the TOE to operate in its evaluated configuration.

Tests

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.

Note that the testing here can be accomplished in conjunction with the testing of the security mechanisms directly (Ref. [6]).

The evaluator tested the ability of the TOE to generate audit records for the relevant events listed in the ST. The evaluator tested separately the logging of administrative clearing of logs and reboot of the system. To this end, the evaluator cleared the logs and requested a reboot of the system. After the reboot, the evaluator examined the logs and determined that the steps of clearing the logs and rebooting the TOE were logged.

The verification of audit-log functions for other events were tested throughout the rest of the test-plan.

5.2.2 **FAU_GEN.2 User identity association**

TSS & Guidance Documentation

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

Tests

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

5.2.3 **FAU_STG_EXT.1 Protected audit event storage**

TSS

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 (Ref. [6])

Section 7.1 of the ST indicates that Syslog can be configured to store the audit logs locally, and optionally to send them to one or more syslog log servers via Netconf over SSH.

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 (Ref. [6]).

As per Sect. 7.1 of the ST, “Local audit log are stored in /var/log/ in the underlying filesystem. Only a Security Administrator can read log files, or delete log and archive files through the CLI interface or through direct access to the filesystem having first authenticated as a Security Administrator. The syslogs are automatically deleted locally according to configurable limits on storage volume. The default maximum size is 1Gb. The default maximum size can be modified by the user, using the “size” argument for the “set system syslog” CLI command.”.

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 (Ref. [6]).

As per Sect. 3 of the ST, the TOE is not distributed. As per Sect. 7.1 of the ST, the TOE stores logs locally and optionally to send them to one or more external syslog log servers.

The evaluator shall examine the TSS to ensure that it details the behaviour 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 (Ref. [6]).

As per Sect. 7.1 of the ST: “The Junos OS defines an active log file and a number “The Junos OS defines an active log file and a number of “archive” files (10 by default, but configurable from 1 to 1000). When the active log file reaches its maximum size, the logging utility closes the file, compresses it, and names the compressed archive file ‘logfile.0.gz’. The logging utility then opens and writes to a new active log file. When the new active log file reaches the configured maximum size, ‘logfile.0.gz’ is renamed ‘logfile.1.gz’, and the active log file is closed, compressed, and renamed ‘logfile.0.gz’. When the maximum number of archive files is reached and when the size of the active file reaches the configured maximum size, the contents of the oldest archived file are deleted so the current active file can be archived”.

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 as well as acceptable frequency for the transfer of audit data (Ref. [6]).

As per Sect 7.1 of the ST “Syslog can be configured to store the audit logs locally (FAU_STG_EXT.1), and optionally to send them to one or more syslog log servers in real time via Netconf over SSH”.

Guidance Documentation

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.), configuration of the TOE needed to communicate with the audit server (Ref. [6]).

Chapter 5 of the guidance (Ref. [11]) provides the necessary documentation required to set up remote logging using SSH NETCONF.

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 (Ref. [6]).

The guidance (Ref. [11]) describes the relationship between the local audit data and the audit data that are sent to the audit log server. This is demonstrated via the following sentences in Chapter 5 of the guidance “A secure Junos OS environment requires the auditing of events and storing them in a local audit file. The recorded events are simultaneously sent to an external syslog 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 (Ref. [6]).

The evaluator has examined the provided operational guidance (Ref. [11]) and determined that it provides clarity on the fact that audit data is overwritten when space for audit data is full as per selection for FAU_STG_EXT.1.3 in the ST (Ref. [8]). According to the guidance (Ref. [11]) the user is able to configure the number of archived log files. These archived log files shall be overwritten when the maximum number of archived log files has been created.

Tests

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 (Ref. [6]).

The evaluator established an SSH connection to the TOE and executed an XML RPC to request the transmission of logs to a dedicated log-server. The evaluator, via monitoring of the packet-capture data, verified that all logging is transmitted encrypted between the TOE and the log-server. The evaluator also verified that connectivity between the log-server and the TOE is restored after the connection is physically interrupted.

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) (Ref. [6]).

The evaluators generated audit data and confirmed that these audit files were stored within the TOE file system. The evaluators confirmed that, upon exhausting the local storage space, the TOE deleted the oldest log file and created a new file to write to. This behaviour is consistent with FAU_STG_EXT.1.

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 (Ref. [6]).

The TOE does not claim compliance with FAU_STG_EXT.2/LocSpace.

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 (Ref. [6]).

The TOE is not distributed and, as such, this test is not applicable.

5.3 Cryptographic Support (FCS)

5.3.1 **FCS_CKM.1 Cryptographic Key Generation**

TSS

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 (Ref. [6]).

Sect. 7.2.1 of the ST indicates: “Asymmetric keys are also generated in accordance with FIPS PUB 186-4 Appendix B.3.3 for RSA Schemes and Appendix B.4.2 for ECC Schemes for SSH communications. The TOE implements Diffie-Hellman group 14, using the modulus and generator specified by Section 3 of RFC3526.”.

Guidance Documentation

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 (Ref. [6]).

The Evaluated Configuration Guide (Ref. [11]) describes how the administrator can configure SSH (in Chapter 4). As part of these configuration guides, the available cryptographic methods and associated key sizes are indicated with configuration examples for how to set these values appropriately.

Tests

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

Diffie-Hellman Group 14

FFC Schemes using "safe-prime" groups

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

(TD0631 is applied)

5.3.2 **FCS_CKM.2 Cryptographic Key Establishment**

TSS

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 (Ref. [6]).

Sect. 7.2.1 of the ST has a table (Table 13) which lists the following DH key exchange protocols for SSHv2:

- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521
- Diffie-Hellman group 14 (modp 2048)

This is consistent with FCS_CKM.1.1 and FCS_CKM.2.1. Further, it is stated that “The TOE implements Diffie-Hellman group 14, using the modulus and generator specified by Section 3 of RFC3526.”

Guidance Documentation

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s) (Ref. [6]).

The Evaluated Configuration Guide (Ref. [11]) describes how the administrator can configure SSH (in Chapter 4). As part of these configuration guides, the available cryptographic methods and associated key sizes are indicated with configuration examples for how to set these values appropriately.

Tests

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

SP800-56B Key Establishment Schemes

The evaluator shall verify the correctness of the TSF's implementation of RSAES-PKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses RSAES-PKCS1-v1_5 (Ref. [6]).

The TOE does not claim SP800-56B key establishment and, hence, this test is not applicable.

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. (Ref. [6]).

The evaluators performed this test as part of the test for FCS_SSHS_EXT.1, where the SSH on the TOE is configured to only use Diffie Hellman group 14 against a known, good implementation of the Diffie Hellman group 14 provided by the SSH client that is provided with Kali Linux.

5.3.3 FCS_CKM.4 Cryptographic Key Destruction

TSS

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 for[1]). 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).

[1] Where keys are stored encrypted or wrapped under another key then this may need to be explained in order to allow the evaluator to confirm the consistency of the description of keys with the TOE functions.

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 (Ref. [6]).

Sect. 7.2.1 of the ST has a table (Table 15) which provides all the required information. For each CSPs (SSH Private Host Key, SSH Session Keys, User Password, RNG State and MACsec keys), the table describes the following:

- usage,
- method of storage (plaintext, encrypted or hashed),
- storage location (SSD or memory);
- zeroization method (several methods).

The set of CSP covers all cryptographic uses specified in the ST.

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 (Ref. [6]).

See above.

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 (Ref. [6]).

See above.

Guidance Documentation

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 command2 and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance) (Ref. [6]).

Chapter 2 of the CC Guide (Ref. [11]) describes how the administrator can perform a zeroisation of the TOE. This will ensure that all Critical Security Parameters (CSPs) are wiped from the TOE. There are no instances where key destruction may be delayed at the physical layer.

Tests

None (Ref. [6]).

5.3.4 FCS_COP.1/DataEncryption Cryptographic Operation (AES Data Encryption/Decryption)

TSS

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 (Ref. [6]).

Table 13 indicates that the TOE implements the following for SSH:

- AES-CBC (128, 256) (Encrypt, Decrypt);
- AES-CTR (128, 256) (Encrypt, Decrypt);

and for MACsec

- AES-GCM (128, 256) (Encrypt, Decrypt);

and for MKA

- AES-CBC (128, 256) (Encrypt);

[2] 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).

Guidance Documentation

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 (Ref. [6]).

Chapter 4 of the CC Guidance (Ref. [12]) provides instructions on selecting the AES mode and key size that is to be used by SSH in the evaluated configuration. Furthermore, Chapter 8 of the same guidance provides information on how to choose the AES-GCM mode and the appropriate key length for MACsec.

Tests

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

KAT-4. To test the encrypt functionality of AES-CBC, the evaluator 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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

5.3.5 FCS_COP.1/SigGen Cryptographic Operation (Signature Generation and Verification

TSS

The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and key size supported by the TOE for signature services (Ref. [6]).

Table 13 indicates that the TOE supports:

- ECDSA (P-256 w/ SHA-256)
- ECDSA (P-384 w/ SHA-384)
- ECDSA (P-521 w/ SHA-521)
    - SigGen, SigVer, KeyGen, KeyVer for ECDH

Also, Sect. 7.2.1 describes that “Asymmetric keys are also generated in accordance with FIPS PUB 186-4 Appendix B.3.3 for RSA Schemes”.

Guidance Documentation

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 (Ref. [6]).

Chapter 4 of the Guidance (Ref. [11]) provides instructions on how to enable the use of ECDSA and RSA signatures for use in SSH.

Tests

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

5.3.6 FCS_COP.1/Hash Cryptographic Operation (Hash Algorithm)

TSS

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 (Ref. [6]).

Table 13 of the ST provides the mapping between primitives and hash functions. The mapping is complete.

Guidance Documentation

The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present (Ref. [6]).

The CC Guide (Ref. [11]) describes how the administrator can configure SSH (in Chapter 4). As part of these configuration guides, the available cryptographic methods and associated key sizes are indicated with configuration examples for how to set these values appropriately.

Tests

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

5.3.7 FCS_COP.1/KeyedHash Cryptographic Operation (Keyed Hash Algorithm)

TSS

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 (Ref. [6]).

Table 13 of the ST has a table that lists the supported HMAC functions, lengths, has functions, block sizes and output MACs.

Guidance Documentation

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 (Ref. [6]).

The Evaluated Configuration Guide (Ref. [11]) describes how the administrator can configure SSH (in Chapter 4). As part of these configuration guides, the available cryptographic methods and associated key sizes are indicated with configuration examples for how to set these values appropriately.

Tests

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

5.3.8 FCS_COP.1(1)/KeyedHashCMAC Cryptographic Operation (AES_CMAC Keyed Hash Algorithm)[3]

TSS

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

Section 7.2.3 of the ST indicates that AES-CMAC-128 or AES-CMAC-256 are used to derived SAKs and that the nonce value is generated from the TOE's RBG. It also describes that the TOE uses pre-shared CAKs, which are input as a string of up to 64 hexadecimal characters, allowing for a maximum security strength of 256 bits.

[3] Introduced by TD0466

Guidance Documentation

No additional guidance review activities are required (Ref. [6]).

Tests

The evaluator shall perform testing for AES-GCM as required by the NDcPP.

In addition to the tests specified in the NDcPP for this SFR, the evaluator shall perform the following tests:

CMAC Generation Test

To test the generation capability of AES-CMAC, the evaluator shall provide to the TSF, for each key length-message length-CMAC length tuple (in bytes), a set of 8 arbitrary key-plaintext tuples that will result in the generation of a known MAC value when encrypted. The evaluator will then verify that the correct MAC was generated in each case.

CMAC Verification Test

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

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

- Key length: values will include the following:
  - 16
  - 32
- Message length: values will include the following:
  - 0 (optional)
  - Largest value supported by the implementation (no greater than 65536)
  - Two values divisible by 16
  - Two values not divisible by 16
- CMAC length
  - Smallest value supported by the implementation (no less than 1)
  - 16
  - Any supported CMAC length between the minimum and maximum values (Ref. [6]).

The AES-CMAC implementation is as specified in the ST (Ref [8]). This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

5.3.9 FCS_COP.1(5) Cryptographic Operation (MACsec AES Data Encryption/Decryption)

TSS

The evaluator shall verify that the TSS describes the supported AES modes that are required for this EP in addition to the ones already required by the NDcPP (Ref. [7]).

Table 12 in the ST (Ref. [8]) describes the support for AES-GCM with key sizes of 128 and 256 bits. These algorithms, as per the ST, are implemented by the Broadcom PHY.

Guidance Documentation

No additional guidance review activities are required (Ref. [7]).

Tests

The evaluator shall perform testing for AES-GCM as required by the NDcPP.

In addition to the tests specified in the NDcPP for this SFR, the evaluator shall perform the following tests for AES Key Wrap in accordance with the NIST “Key Wrap Validation System”:

KW-AE Test

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

KW-AD Test

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

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

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

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

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

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

5.3.10 FCS_RBG_EXT.1 Extended: Cryptographic Operation (Random Bit Generation)

TSS

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 (Ref. [6]).

Section 7.2.1 specifies the DRBG of the TOE as HMAC-DRBG using SHA-256,

Paragraph 59 specifies the seeding information and refers to the guidance documentation for more detail.

The evaluator cited the Entropy Analysis Report provided by the vendor which provided detailed information on the design, the health tests, RBG, an overview of the primary entropy source, and an evaluation of the entropy produced by this source.

Guidance Documentation

The evaluator shall confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality (Ref. [6]).

The DRBG utilised by the TOE is non-configurable by the Administrator and is automatically used by the TOE. As such, this requirement is non-applicable.

Tests

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 (Ref. [6]).

This assurance activity is carried out using a CAVS-like testing tool. Further details are contained in Section 4.

5.3.11 **FCS_SSHS_EXT.1 SSH Server Protocol**

TSS

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 is applied)

Section 7.2.1 of the ST describes the acceptable public key algorithms for authentication, namely:

- ssh-rsa
- ecdsa-sha2-nistp256
- ecdsa-sha2-nistp384

- ecdsa-sha2-nistp521

Section 7.2.2 describes that the TOE supports password-based authentication for SSH and verify that the SSH client's presented public key matches one that is stored "in its known_hosts list of keys".

FCS_SSHS_EXT.1.3

The evaluator shall check that the TSS describes how "large packets" in terms of RFC 4253 are detected and handled (Ref. [6]).

Table 16, Section 7.2.2 of the ST, indicates that packets greater than 263Kbytes in an SSH transport connection are dropped and the connection is terminated by Junos OS.

FCS_SSHS_EXT.1.4

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 (Ref. [6]).

Table 16, Section 7.2.2 of the ST, describes the optional characteristics of SSH. The described supported encryption algorithms (AES-CBC-128, AES-CBC-256, AES-CTR-128, AES-CTR-256) correspond to those selected in FCS_SSHS_EXT.1.4.

FCS_SSHS_EXT.1.5

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 is applied)

Per Table 16, Section 7.2.2 of the ST, the TOE uses keys generated in accordance with "ssh-rsa", "ecdsa-sha2-nistp256", "ecdsa-sha2-nistp384" or "ecdsa-sha2-nistp521" to perform public-key based device authentication.

FCS_SSHS_EXT.1.6

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 (Ref. [6]).

As per Table 16, Section 7.2.2 of the ST, the TOE permits negotiation of HMAC-SHA1 in each direction for SSH transport, according to RFC 6668.

FCS_SSHS_EXT.1.7

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 (Ref. [6]).

As per Table 16, Section 7.2.2 of the ST, "key exchange is performed only using one of the supported key exchange algorithms, which are ordered as follows: ecdh-sha2-nistp256, ecdh-sha2-nistp384, ecdh-sha2-nistp521 (all specified in RFC 5656), diffie-hellman-group14-sha1 (specified in RFC 4253)." This list corresponds to the definition of FCS_SSHS_EXT.1.7.

FCS_SSHS_EXT.1.8

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 (Ref. [6]).

The TSS in Section 7.2.2 of the SP indicates that “for ciphers whose blocksize >= 16, the TOE rekeys every (2^32-1) bytes. The client may explicitly request a rekeying event as a valid SSHv2message at any time and the TOE will honor this request. Re-keying of SSH session keys can be configured using the sshd_config knob. The data-limit must be between 51200 and 4294967295 (2^32-1) bytes and the time-limit must be between 1 and 1440 minutes.

Guidance Documentation

FCS_SSHS_EXT.1.4

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) (Ref. [6]).

The evaluator was satisfied with the level of detail provided by the instructions in Chapter 4 of the CC Guide (Ref. [11]) on configuring the TOE for SSH. This chapter provided steps on:

- Configuring the host-key algorithms;
- Configuring the key-exchange algorithms for Diffie-Hellman;
- Configuring the message authentication codes;
- Configuring the ciphers;
- Configuring the maximum number of user-login attempts; and
- Configuring the backoff-factor – the amount of incremental delay that’s introduced for subsequent failed login attempts.

Sufficient detail was provided by these steps such that the SSH functionality conformed to the descriptions in the TSS.

FCS_SSHS_EXT.1.5

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) (Ref. [6]).

Refer to guidance evaluation activity for SSHS_EXT.1-4.

FCS_SSHS_EXT.1.6

The evaluator shall also check the guidance documentation to ensure that it contains instructions to the 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) (Ref. [6]).

Sufficient detail was provided by the steps in Chapter 4 of the CC Guide (Ref. [11]) such that the SSH functionality conformed to the descriptions in the TSS.

The guidance provides the following set of statements to configure the allowed data integrity algorithms:

```
set system services ssh macs hmac-sha1
set system services ssh macs hmac-sha2-256
set system services ssh macs hmac-sha2-512
```

FCS_SSHS_EXT.1.7

The evaluator shall also check the guidance documentation to ensure that it contains instructions to the administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE (Ref. [6]).

Sufficient detail was provided by the steps in Chapter 4 of the CC Guide (Ref. [11]) such that the SSH functionality conformed to the descriptions in the TSS.

The guidance provides the following set of statements to configure the allowed key-exchange algorithms:

```
set system services ssh key-exchange dh-group14-sha1
set system services ssh key-exchange ecdh-sha2-nistp256
set system services ssh key-exchange ecdh-sha2-nistp384
set system services ssh key-exchange ecdh-sha2-nistp521
```

FCS_SSHS_EXT.1.8

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 (Ref. [6]).

The guidance (Ref. [11]) provides for details on the configuration of thresholds for data-based and time-based rekeying for SSH.

Tests

FCS_SSHS_EXT.1.2

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. (TD0631 is applied)

The evaluator conducted the test and confirmed that the TOE permitted SSH connections for each supported client public-key algorithm when configured with the related public key.

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.

(TD0631 is applied)

The evaluator conducted the test and confirmed that the TOE did not permit the public-key-based SSH connection with the unrelated public key.

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 is applied)

Note: Test 1 and 2 for public key authentication is tested as part of testing for FCS_SSHS_EXT.1.5 (Ref. [6]).

The evaluator conducted the test and confirmed that the TOE permitted password-based authentication for SSH when using a valid password and rejects password-based authentication for SSH when using an invalid password.

FCS_SSHS_EXT.1.3

The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped (Ref. [6]).

The evaluators established an SSH between a client and the TOE. The evaluators then sent a packet of just over 263 kilo bytes in size and confirmed that the packet was dropped by the TOE.

FCS_SSHS_EXT.1.4

The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish a SSH connection. To verify this, the evaluator shall start session establishment for a 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 (Ref. [6]).

Per the guidance documentation, the evaluators configured the TOE to only offer those algorithms and cryptographic primitives specified in this requirement. The evaluators then commenced session establishment between a remote client and the TOE while monitoring network traffic between the two. The evaluators confirmed that the server KEXINIT packet, which as per RFC 4253 lists the cryptographic algorithms offered by the server for negotiation, contained only those algorithms specified in this requirement.

FCS_SSHS_EXT.1.5

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.

The algorithms supported by the TOE are: ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521.

(TD0631 is applied)

The evaluators connected to the TOE from an SSH client using each of the public key algorithms specified in this requirement (ssh-rsa, rsa-sha2-256, rsa-sha2-512, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521). Via Wireshark analysis, the evaluators were able to confirm successful authentication, negotiation and establishment of an SSH session.

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 is applied)

The evaluators created a 1024-bit DSA key pair for use in public key authentication. Attempts to load this key onto the TOE for use in SSH public-key authentication were met with an error (as the TOE only permits RSA and ECDSA keys of sizes specified in FCS_SSHS_EXT.1).

FCS_SSHS_EXT.1.6

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 (Ref. [6]).

The evaluator opened multiple SSH connections to the TOE with each connection restricted to one of the supported integrity algorithms. The evaluator examined the packet-capture of the negotiated

connections and confirmed that the TOE permitted the exclusive use of hmac-sha1, hmac-sha2-256 and hmac-sha2-512 for SSH connections.

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 in 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 (Ref. [6]).

The evaluator attempted to establish an SSH connection to the TOE by restricting the connection attempt to the hmac-sha1-96 integrity algorithm. The TOE did not permit the use of the MAC algorithm.

FCS_SSHS_EXT.1.7

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 (Ref. [6]).

The evaluator configured an SSH client to only use diffie-hellman-group1-sha1 for key exchange and attempted to connect to the TOE. The evaluators confirmed that the TOE rejected this authentication attempt.

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 (Ref. [6]).

The evaluator configured an SSH client to use each of the specified key exchange methods (dh-group14-sha1, ecdh-sha2-nistp521, ecdh-sha2-nistp256 and ecdh-sha2-nistp384) in turn. The evaluator confirmed that, for each specified key exchange method, the TOE permitted the connection and successfully established an SSH session.

FCS_SSHS_EXT.1.8

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/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 (Ref. [6]).

The evaluator configured the TOE to have an SSH session rekey time of 60 minutes. The evaluators established a session from an SSH client and ensured that the session was kept alive for longer than 60 minutes. Via Wireshark traffic capture and audit log examination, the evaluators confirmed that a) the TOE initiated an SSH rekey upon reaching the 60-minute threshold; and b) an audit log was generated to indicate that the rekey event took place.

The evaluator then configured the TOE to have an SSH session rekey data limit of 1 gigabyte. The evaluator established a session from an SSH client and began to transmit traffic to exceed this threshold (via transfer of a large file). Via Wireshark traffic capture and audit log examination, the evaluators confirmed that a) the TOE initiated an SSH rekey upon reaching the 1 gigabyte threshold; and b) an audit log was generated to indicate that the rekey event took place.

The evaluator confirmed that both traffic-based and time-based thresholds for SSH rekey are configurable (as specified in the guidance documentation) and are only accessible to authorised administrators.

5.3.12 **FCS_MACSEC_EXT.1 MACsec**

TSS

The evaluator shall examine the TSS to verify that it describes the ability of the TSF to implement MACsec in accordance with IEEE 802.1AE-2006. The evaluator shall also determine that the TSS describes the ability of the TSF to derive SCI values from peer MAC address and port data and to reject traffic that does not have a valid SCI. Finally, the evaluator shall check the TSS for an

assertion that only EAPOL, MACsec Ethernet frames, and MAC control frames are accepted by the MACsec interface.

(TD0553 is applied).

Section 7.2.3 of the ST indicates that the TOE implements IEEE 802.1AE-2006. It also describes that each secure channel is identified by a SCI comprised of a globally unique MAC address and port Identifier, and the TOE can be configured to enforce SCI tagging such that traffic without a valid SCI is rejected. The same section asserts that only EAPOL, MACsec Ethernet frames and control frames (EtherType is 88-08) are accepted by the MACsec interface.

Guidance Documentation

There are no guidance activities for this SFR (Ref. [7]).

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall successfully establish a MACsec channel between the TOE and a MACsec-capable peer in the Operational Environment and verify that the TSF logs the communications. The evaluator shall capture the traffic between the TOE and the Operational Environment to determine the SCI that the TOE uses to identify the peer. The evaluator shall then configure a test system to capture traffic between the peer and the TOE to modify the SCI that is used to identify the peer. The evaluator then verifies that the TOE does not reply to this traffic and logs that the traffic was discarded.

Test 2: The evaluator shall send Ethernet traffic to the TOE's MAC address that iterates through the full range of supported EtherType values (refer to http://standards.ieee.org/develop/regauth/ethertype/eth.txt) and observes that traffic for all EtherType values is discarded by the TOE except for the traffic which has an EtherType value of 88-8E, 88-E5 or 8808. Note that there are a large number of EtherType values so the evaluator is encouraged to execute a script that automatically iterates through each value (Ref. [7]).

TD0553 is applied.

The evaluator confirmed that the the TSF logged the event when a MACsec connection was established. When a packet had a different SCI value, it was then discarded and logged. The evaluator confirmed that when sending Ethernet traffic to the TOE's MAC address that iterates through the full range of supported EtherType values, only MACsec, MKA frames and frames with ethertype of 8808 were accepted, all other Ethernet packet types were rejected.

5.3.13 **FCS_MACSEC_EXT.2 MACsec Integrity and Confidentiality**

TSS

The evaluator shall examine the TSS to verify that it describes the methods that the TOE implements to provide assurance of MACsec integrity, including the confidentiality offset(s) used, the use of an ICV (including the supported length), and generating the ICV with the SAK, using the SCI as the most significant bits of the IV and the 32 least significant bits of the PN as the IV (Ref. [7]).

Section 7.2.3 of the ST describes how MACsec PDUs are integrity protected using an ICV calculated over the destination address, source address, SecTAG, and payload. The ICV is calculated using SAK and IV composed by concatenation of the SCI and PN values, thus assuring uniqueness. The same section describes the confidentiality offsets supported.

Guidance Documentation

If any integrity verifications are configurable such as the confidentiality offset(s) used or the mechanism used to derive an ICK, the evaluator shall verify that instructions for performing these functions are documented (Ref. [7]).

The evaluator found that the mechanism to derive the ICK was fixed and could not be set. However, the confidentially offset was able to be configured. Examples of this were provided in Chapter 8 of the guide (Ref. [11]). For example, the "Configuring Static MACsec with ICMP traffic" section provides the following sample to set an offset of '30' octets that may be sent in unencrypted-plaintext:

```
set security macsec connectivity-association CA1 offset 30
```

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall transmit MACsec traffic to the TOE from a MACsec-capable peer in the Operational Environment. The evaluator shall verify via packet captures and/or audit logs that the frame bytes after the MACsec Tag values in the received traffic is not obviously predictable.

Test 2: The evaluator shall transmit valid MACsec traffic to the TOE from a MACsec-capable peer in the operational environment that is routed through a test system set up as a man-in-the-middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure (Ref. [7]).

The evaluator confirmed that the frame bytes after the MACsec Tag values of the captured MACsec packets were different from each other and showed no patterns.

The evaluator also confirmed that the TOE discarded frames which are modified in a man-in-the-middle setting.

5.3.14 **FCS_MACSEC_EXT.3 MACsec Randomness**

TSS

The evaluator shall examine the TSS to verify that it describes the method used to generate SAKs and nonces and that the strength of the CAK and the size of the CAK's key space are provided (Ref. [7]).

Section 7.2.3 of the ST indicates that AES-CMAC-128 or AES-CMAC-256 are used to derived SAKs and that the nonce value is generated from the TOE's RBG. It also describes that the TOE uses pre-shared CAKs, which are input as a string of up to 64 hexadecimal characters, allowing for a maximum-security strength of 256 bits.

Guidance Documentation

There are no guidance activities for this SFR (Ref. [7]).

Tests

Testing of the TOE's MACsec capabilities and verification of the DRBG is sufficient to demonstrate that this SFR has been satisfied (Ref. [7]).

Not applicable.

5.3.15 **FCS_MACSEC_EXT.4 MACsec Key Usage**

TSS

The evaluator shall check the TSS to ensure that it describes how the SAK is wrapped prior to being distributed using the AES implementation specified in this EP (Ref. [7]).

Section 7.2.3 of the ST indicates that the SAK is protected using AES Key Wrap with a key encryption key (KEK) derived from the CAK.

Guidance Documentation

If the method(s) of peer authentication is configurable, the evaluator shall verify that the guidance provides instructions on how to configure this. The evaluator shall also verify that the method of specifying a lifetime for CAKs is described (Ref. [7]).

As per guidance, the peer authentication is configurable. The guidance provides for commands on how to set the Connectivity Association Key (CAK). For example:

```
prompt security macsec connectivity-association pre-shared-key cak
```

The lifetime of an authentication may be configured through configuring the authentication key chain mechanism. The evaluator shall set-up an authentication key-chain where each key is provided with a 'start-time'. This is described in the form of examples provided in the section "Configuring MACsec with keychain using ICMP traffic" in the Evaluated Configuration Guide (Ref. [11]).

Tests

The evaluator shall perform the following tests:

Test 1: For each supported method of peer authentication in FCS_MACSEC_EXT.4.1, the evaluator shall follow the operational guidance to configure the supported method (if applicable). The evaluator shall set up a packet sniffer between the TOE and a MACsec-capable peer in the Operational Environment. The evaluator shall then initiate a connection between the TOE and the peer such that authentication occurs and a secure connection is established. The evaluator shall wait 1 minute and then disconnect the TOE from the peer and stop the sniffer. The evaluator shall use the packet captures to verify that the secure channel was established via the selected mechanism and that the EtherType of the first data frame sent between the TOE and the peer is 88-E5.

Test 2: The evaluator shall capture traffic between the TOE and a MACsec-capable peer in the Operational Environment. The evaluator shall then cause the TOE to distribute a SAK to that peer, capture the MKPDUs from that operation, and verify the key is wrapped in the captured MKPDUs.

Test 3: The evaluator shall set up an environment where the TOE is capable of communicating with two MACsec-capable peers in its Operational environment. The evaluator shall load a CAK into the TOE and the two peer devices, specifying a short lifetime, say, 10 minutes, and restore. The evaluator shall test two cases, one where the TOE is designated as the Key Server and principal actor, and one where it is the first peer (and also not a Key Server). The evaluator shall disconnect the second peer device, wait 10 minutes, and then reconnect the second peer. The evaluator shall verify in both cases that after 10 minutes, the Key Server will rekey the CA with the first peer, and then when the second peer is reconnected, the Key Server generates a new CAK that is distributed to that peer (Ref. [7]).

(TD0273 is applied)

The evaluator confirmed that the TOE successfully established a MACsec connection using the CAK-based method. After that the first packet, not including the MKA packets, all packets sent from the TOE to Alice had the EtherType of 88-E5.

The evaluator also confirmed that the SAK keys distributed from TOE to Alice were wrapped in ciphertexts.

For the final test, TD0273 is applied so it is not required.

5.3.16 **FCS_MKA_EXT.1 MACsec Key Agreement**

TSS

The evaluator shall examine the TSS to verify that it describes the methods that the TOE implements to provide assurance of MKA integrity, including the use of an ICV and the ability to use a KDF to derive an ICK. (Ref. [7]).

Section 7.2.3, paragraph 76 describes how ICVs are derived and used to provide MKA integrity.

The evaluator shall verify that the TSS describes the TOE's compliance with IEEE 802.1X-2010 and 802.1Xbx-2014 for MKA, including the values for MKA and Hello timeout limits and support for data delay protection. The evaluator shall also verify that the TSS describes the ability of the PAE of the TOE to establish unique CAs with individual peers and group CAs using a group CAK such that a new group SAK is distributed every time the group's membership changes. The evaluator shall also verify that the TSS describes the invalid MKPDUs that are discarded automatically by the TSF in a manner that is consistent with the SFR, and that valid MKPDUs are decoded in a manner consistent with IEEE 802.1X-2010 section 11.11.4 (Ref. [7]).

Section 7.2.3 of the ST indicates that the TOE enforces MKA timeouts according to IEEE 802.1X-2010 and 802.1Xbx-2014, listed in Table 17. As per FCS_MKA_EXT.1.6, the TOE uses pairwise CAKs and does not support group CAKs. The same section describes how the TOE decodes and automatically discard invalid MKPDUs as per Section 11.11.4 of 802.1X-2010

Guidance Documentation

The evaluator shall verify that the guidance documentation provides instructions on how to configure the TOE to act as the Key Server in an environment with multiple MACsec-capable devices (Ref. [7]).

The evaluator was able to verify that the document provides instruction on setting the priority of the TOE such that the TOE is deemed to act as the Key Server. For example, Step 7 of the "Configuring MACsec with keychain using ICMP Traffic section" provides the following command to ensure that the TOE is ensured to be the Key Server:

```
security-administrator@hostname:fips# set security macsec connectivity-
association CA1 mka
key-server-priority 1
```

Tests

FCS_MKA_EXT.1.2

The test below requires the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with. Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the Key Server and principal actor.

The evaluator shall then perform the following test: The evaluator shall use a peer device to send traffic to the TOE, arbitrarily inducing artificial delays in their transmission using a man-in-the-middle setup. The evaluator shall observe that traffic delayed longer than 2.0 seconds is rejected.

(TD0618 is applied).

The evaluator used the custom MACsec client to conduct the test and confirmed that when MACsec was enabled, traffic delayed longer than 2.0 seconds was rejected.

FCS_MKA_EXT.1.4

The evaluator shall perform the following tests:

Test 1: The evaluator shall transmit MKA traffic (MKPDUs) to the TOE from a MKA-capable 21 peer in the Operational Environment. The evaluator shall verify via packet captures and/or audit logs that the last 16 octets of the MKPDUs in the received traffic do not appear to be predictable. Ref. [7]

The evaluator used the custom MACsec client to conduct the test and confirmed that the last 16 octets of the MKPDUs in the received traffic did not appear to be predictable.

Test 2: The evaluator shall transmit valid MKA traffic to the TOE from a MKA-capable peer in the operational environment that is routed through a test system set up as a man-in-the- middle. The evaluator shall use the test system to intercept this traffic to modify one bit in a packet payload

before retransmitting to the TOE. The evaluator shall verify that the traffic is discarded due to an integrity failure.

The evaluator used the custom MACsec client to conduct the test and confirmed that modified MKA traffic was discarded.

FCS_MKA_EXT.1.5

The tests below require the TOE to be deployed in an environment with two MACsec-capable peers, identified as devices B and C, that the TOE can communicate with.

Prior to performing these tests, the evaluator shall follow the steps in the guidance documentation to configure the TOE as the Key Server and principal actor (peer). The evaluator shall then perform the following tests using a traffic sniffer to capture this traffic.

Test 1: The evaluator shall send a fresh SAK that includes both peers as active participants. The evaluator shall start an MKA session between the TOE and the two active participant peers and send MKPDUs. The evaluator shall verify from packet captures that MKPDUs are sent at least once every half-second.

(TD0618 is applied).

The evaluator used the custom MACsec client to conduct the test and confirmed that MKPDUs were sent at least once every half-second.

Test 2: Disconnect one of the peers. Using a man-in-the-middle device, arbitrarily introduce an artificial delay in sending a fresh SAK following the change in the Live Peer List. Repeat Test 1 delaying a fresh SAK for MKA Lifetime traffic and observe that the timeout of 6.0 seconds is enforced by the TSF.

(TD0618 is applied).

The evaluator used the custom MACsec client to conduct the test and confirmed that the timeout of 6.0 seconds was enforced for MKA Lifetime traffic by the TSF.

FCS_MKA_EXT.1.8

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

Test 1: [conditional]: If the TOE supports group CAKs, the evaluator shall perform the following steps

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

2. Generate a group CAK for the group of 3 devices using ieee8021XKayCreateNewGroup.

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

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

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

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

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

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

9. Invoke ieee8021XKayCreateNewGroup again.

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

(TD0618 is applied).

Note: Tests 3 and 4 (from the original EP) are not performed due to the application of TD0618. Test 1 for FCS_MKA_EXT 1.8 (as specified in TD0618) is not performed as group CAK is not claimed.

Test 2: The evaluator shall start an MKA session between the TOE and the two environmental MACsec peers and then perform the following steps:

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

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

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

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

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

(TD0618 is applied).

The evaluator used the custom MACsec client and sent five different malformed MKA frames as specified in Test 2 (of TD0618). All malformed MKA frames were rejected by the TOE.

5.4 Identification and Authentication (FIA)

5.4.1 **FIA_AFL.1 Authentication Failure Management**

TSS

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) (Ref. [6]).

Section 7.3 of the ST describes how successive unsuccessful authentication attempt are detected and tracked. It explains how the TOE can be configured to specify the action to be taken if the administrator fails to enter valid username/password credentials for password authentication when attempting to authenticate via remote access.

Guidance Documentation

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 (Ref. [6]).

The evaluator has examined the guidance (Ref. [11]) to ensure that instruction for configuring the number of successive unsuccessful authentication attempts for authentication are provided. The “Limiting the Number of User Logins Attempts for SSH Sessions” section in Chapter 4 of the ECG (Ref. [11]) provides detailed commands for the administrator to enter in order to set the correct lock-out period; tries before disconnection; back-off threshold (the delay after an unsuccessful attempt); back-off factor (the factor by which the delay increases after each unsuccessful attempt); and whether the root user may be allowed to login via SSH.

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 (Ref. [6]).

The evaluator has found that the guidance (Ref. [11]) 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. Page 45 states that “The local administrator access will be maintained even if the remote administration is made permanently or temporarily

unavailable due to the multiple failed login attempts. The console login for local administration will be available to the users during the lockout period.”.

Tests

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).

c) 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 (Ref. [6]).

The evaluator configured the TOE to lock-out access for a user after 3 failed attempts at authentication via SSH. The evaluator also configured the number of minutes for a lockout period to be 5 minutes. This configuration was then validated by the evaluator by attempting a login via SSH with incorrect password credentials 3 times. Having been locked out, the evaluator was only able to log in with the correct credentials after 5 minutes, or via the serial console using the credentials of the user that’s locked out. Since the locking of a user-account only applies to SSH connections, the evaluator was able to still access the system via the serial console.

5.4.2 **FIA_PMG_EXT.1 Password Management**

TSS

The evaluator shall examine the TSS to determine that it contains the lists of the supported special character(s) and minimum and maximum number of charters supported for administrator passwords (Ref. [6]).

Section 7.3 of the ST describes that authentication data for fixed password authentication is a case-sensitive, alphanumeric value. The password has a minimum length of 10 characters and maximum length of 20 characters, and must contain characters from at least two different character sets (upper, lower, numeric, punctuation), and can be up to 20 ASCII characters in length (control characters are not recommended). Any standard ASCII, extended ASCII and Unicode characters can be selected when choosing a password.

Guidance Documentation

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 (Ref. [6]).

Chapter 2 of the Evaluated Configuration Guide (Ref. [11]) identifies the character requirements that may be used in passwords. It states that:

"Passwords must contain at least three of the following five defined character sets:

- Uppercase letters
- Lowercase letters
- Digits
- Punctuation marks
- Keyboard characters not included in the other four sets-such as the percent sign(%) and the ampersand (&)"

Furthermore, Chapter 3 of the guide states that passwords must contain "both alphanumeric and punctuation characters, composed of any combination of upper and lowercase letters, numbers, and special characters such as,"!", "@","#", "$", "%", "A", "&","*","(", and")". There should be at least a change in one case, one or more digits, and one or more punctuation marks."

Chapter 3 of the guide also provides commands for administrators to set the password policy via the following commands:

```
set system login password minimum-length 10
set system login password change-type character-sets
set system login password minimum-changes 3
set system login password change-type character-sets
```

In addition to this, the guide states that hashing algorithm for user passwords can be either SHA256 or SHA512 and provides the following command to set the default hashing algorithm:

```
set system login password format sha512
```

Tests

The evaluator shall perform the following tests.

a) Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to 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 (Ref. [6]).

To perform this test, the evaluator configured the maximum and minimum length of the accepted passwords and the minimum change in character sets as per guidance documentation. The evaluator then tried different sets of passwords that are expected to pass and fail the password requirements enforced by the TOE and confirmed that the TOE behaved as expected.

5.4.3 FIA_UIA_EXT.1 User Identification and Authentication

TSS

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” (Ref. [6]).

Section 7.3 of the ST describes the logon process for each logon method allowed (local console and SSH). The Authentication process and library are login() and PAM Library module.

Following TOE initialization, the login() process is listening and can be accessed through either direct connection to the local console or following successful establishment of a remote management connection over SSH, when a login prompt is displayed. For password authentication, login() interacts with a user to request a username and password to establish and verify the user’s identity. The SSH daemon also supports public key authentication of clients.

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 (Ref. [6]).

Section 7.3 of the SP specifies that prior to authentication, the only Junos OS managed responses provided to the administrator are:

- Negotiation of SSH session
- Display of the access banner
- ICMP echo responses.

Guidance Documentation

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 (Ref. [6]).

The Evaluated Configuration Guide provides:

- Guidance on configuring administrative credentials and privileges (Chapter 3); and
- Guidance on configuring SSH and Console Connections (Chapter 4).

An administrator successfully authenticates to the TOE by providing a username and password combination matching the stored credentials (for both console and SSH).

There is no configuration required to limit services available prior to login.

Tests

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:

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 (Ref. [6]).

The evaluator tested all methods of authentication to the TOE: console, password-based SSH login and public-key based SSH login. The evaluator carried out a successful login using each one of these methods. The evaluator also carried out an unsuccessful attempt at login using each one of these methods. A login-banner was displayed where appropriate. Furthermore, all attempts at login were recorded in the TOE's syslog in the appropriate format.

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 (Ref. [6]).

The evaluator carried out an nmap scan on the IP address of the TOE for all available protocols and ports. The only protocols identified were TCP and ICMP. The only TCP service that was available on the TOE was SSH and NETCONF (830) as expected.

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 (Ref. [6]).

The only other service prior to the local administrator logging in to the TOE is the OAM shell. The evaluator accessed the shell and confirmed that no services are available to the administrator before logging in.

5.4.4 FIA_UAU_EXT.2 Password-based Authentication Mechanism

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 (Ref. [6]).

The TOE requires users to provide unique identification and authentication data (passwords/public key) before any access to the system is granted.

5.4.5 FIA_UAU.7 Protected Authentication Feedback

TSS

None (Ref. [6]).

The username entered by the administrator at the username prompt is reflected to the screen, but no feedback to screen is provided while the entry made by the administrator at the password prompt until the Enter key is pressed (FIA_UAU.7).

Guidance Documentation

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 (Ref. [6]).

The TOE does not require any preparatory steps to ensure that authentication data is not revealed while entering for each login. No feedback is provided to the user.

Tests

The evaluator shall perform the following test for each method of local login allowed:

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 (Ref. [6])

The evaluator confirmed that the TOE does not provide any display output/feedback when passwords are entered as part of the user login process.

5.5 Security Management (FMT)

5.5.1 **FMT_MOF.1/ManualUpdate Management of security functions behaviour**

TSS

None (Ref. [6]).

N/A

Guidance Documentation

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) (Ref. [6]).

As per “Installing Software on the Device with Single Routing Engine” section in Chapter 2, the new package was able to be installed using the command:

```
request vmhost software add <package>
```

Tests

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 shall 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 (Ref. [6]).

To execute this test the evaluator copied the firmware image on to a USB flash drive. This was then mounted on the TOE by a user with admin privileges. The evaluator then connected to the TOE, via SSH, as a user with no administrator privileges and attempted to install the update by entering the corresponding CLI commands. The attempt failed.

5.5.2 **FMT_MOF.1/Services Management of security functions behaviour**

TSS

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 (Ref. [6]).

The Security Administrator has the capability to:

- Manage Functions:
    - Transmission of audit data to an external IT entity, including Start/stop and modify the behaviour of the trusted communication channel to external syslog server (netconf over SSH) and the trusted path for remote Administrative sessions (SSH) (FMT_MOF.1/Functions, FMT_MOF.1/Services, FMT_SMF.1)
    - Handling of audit data, including setting limits of log file size (FMT_MOF.1/Functions)

Guidance Documentation

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 (Ref. [6]).

The Guidance (Ref. [11]) provides information on how to configure the trusted communication channel to an external syslog server using NETCONF over SSH in Chapter 5. Details on how to configure SSH are provided in Chapter 4. Instruction on how to configure the handling of audit data is provided in Chapter 6 of the guidance.

Tests

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 (Ref. [6]).

The evaluators authenticated to the device as a user that does not have security administrator privileges and attempted to run the commands to execute the FIPS self-tests and carry out a reboot. The evaluator confirmed that the attempt failed.

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 (Ref. [6]).

The evaluator authenticated to the device as a security administrator and attempted to run the commands to execute the FIPS self-tests and carry out a reboot. The evaluator confirmed that the attempt was successful.

5.5.3 FMT_MOF.1/Functions Management of security functions behaviour

TSS

None (Ref. [6]).

The Security Administrator has the capability to:

- Manage Functions:
  - Transmission of audit data to an external IT entity, including Start/stop and modify the behaviour of the trusted communication channel to external syslog server (netconf over SSH) and the trusted path for remote Administrative sessions (SSH) (FMT_MOF.1/Functions, FMT_MOF.1/Services, FMT_SMF.1)
  - Handling of audit data, including setting limits of log file size (FMT_MOF.1/Functions)

Guidance Documentation

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 (Ref. [6]).

As per the references provided in the Guidance (Ref. [11]) documentation “To prevent log files from growing too large, by default the Junos OS system logging utility writes messages to a sequence of files of a defined size. The files in the sequence are referred to as archive files to distinguish them from the active file to which messages are currently being written.”. It goes on to state “When an active log file called logfile reaches the maximum size, the logging utility closes the file, compresses it, and names the compressed archive file logfile.0.gz. The logging utility then opens and writes to a new active file called logfile. This process is also known as file rotation. When the new logfile reaches the configured maximum size, logfile.0.gz is renamed logfile.1.gz, and the new logfile is closed, compressed, and renamed logfile.0.gz. By default, the logging utility creates up to 10 archive files in this manner. When the maximum number of archive files is reached and when the size of the active file reaches the configured maximum size, the contents of the last archived file are overwritten by the current active file.”. The maximum size of each file and the maximum number of files before logs are overwritten can be specified by the `set system archive <file number> <size>` command.

Tests

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 (Ref. [6]).

No access to TOE services and TSF data is permitted prior to authentication as a Security Administrator. As a non-authenticated user, access extends as far as the login prompt and the user must successfully authenticate before any further access is granted.

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 (Ref. [6]).

The evaluator authenticated to the TOE as a Security Administrator and confirmed that, when in configuration mode, the functionality was provided to alter security-related parameters (e.g. cipher suites, authentication methods) for transmission of audit logs to an external entity. Evaluators confirmed that, upon committing the configuration and establishing the secure tunnel for audit log transmission, the revised configuration was used.

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 (Ref. [6]).

No access to TOE services and TSF data is permitted prior to authentication as a Security Administrator. As a non-authenticated user, access extends as far as the login prompt and the user must successfully authenticate before any further access is granted.

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 (Ref. [6]).

The evaluator authenticated to the TOE as a Security Administrator and confirmed that, when in configuration mode, the functionality was provided to alter security-related parameters (such as audit log size, the number of audit log files to be stored on the device, etc.) related to audit log storage. The evaluator confirmed that, once these parameters had been adjusted, the TOE took action (e.g. creating new log files, deleting older log files, etc.) as expected.

5.5.4 **FMT_MTD.1/CoreData Management of TSF Data**

TSS

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 (Ref. [6]).

Prior to authentication, the only Junos OS managed responses provided to the administrator are:

- Negotiation of SSH session
- Display of the access banner
- ICMP echo responses.

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 (Ref. [6]).

The TOE does not support X.509 certificates, therefore this requirement is not applicable.

Tests

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 (Ref. [6]).

No separate testing for FMT_MTD.1/CoreData is required because all management functions have already been exercised under any other SFR.

Guidance Documentation

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 (Ref. [6]).

The documentation (Ref. [11]) groups functionality into chapters (administrative credentials and privileges, SSH, event logging, etc.), which allows for simple identification of which functions are applicable to the requirements of the cPP.

The TOE implements a single role, that of the authorised administrator. As such, no configuration is required to restrict access to TOE functions and TSF data.

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 (Ref. [6]).

5.5.5 The TOE does not support the handling of X509v3 certificates.FMT_MTD.1/CryptoKeys Management of TSF Data

TSS

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. (Ref. [6]).

The Security Administrator has the capability to:

Manage crypto keys (***FMT_MTD.1/CryptoKeys***):

o SSH key generation (ecdsa, ssh-rsa)

Guidance Documentation

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 (Ref. [6]).

The evaluator found the guidance document (Ref. [11]) referred to documentation on how to manage SSH keys.

Tests

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 (Ref. [6]).

The evaluator was able to confirm that no operation can be carried out without prior authentication in Junos, and that a user without security administrator privileges is not able to enter the configuration mode. Therefore, this user is unable to set any configuration material relevant to the SSH public keys for a user.

The evaluator was able to verify that when a user with administrator privileges is authenticated, the user can generate SSH host authentication keys.

5.5.6 FMT_SMF.1 Specification of Management Functions

TSS (containing also requirements on Guidance Documentation)

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 (Ref. [6]).

The evaluator shall verify that the TSS describes the ability of the TOE to provide the management functions defined in this SFR in addition to the management functions required by the base NDcPP (TD0652 is applied).

Section 7.4 of the ST specifies the security management functions available via the serial port on the device or remotely over SSH. These functions correspond to those described in the guidance documentation as well as those observed by the evaluators during exercising of the TOE. The distinction between the local and remote administrative interfaces is made clear both in the TSS and guidance documentation.

The documentation (Ref. [11]) groups functionality into chapters (administrative credentials and privileges, SSH, event logging, etc.), which allows for simple identification of which functions are applicable to the requirements of the functions specified in FMT_SMF.1

Section 7.4 of the ST also describes the ability of the TOE to provide the management functions defined additionally for MACsec in this SFR.

Tests

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 (Ref. [6]).

These management functions are addressed under other SFRs.

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

In addition to the tests specified in the NDcPP for this SFR, the evaluator shall follow the relevant operational guidance to perform the tests listed below. Note that if the TOE claims multiple management interfaces, the tests should be performed for each interface that supports the functions

Test 1: The evaluator shall connect to the PAE of the TOE and install a PSK. The evaluator shall then specify a CKN and that the PSK is to be used as a CAK.

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

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

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

(TD0652 is applied).

The evaluator conducted the test confirmed that the Pre-shared-key and CKN values input to the TOE were handled correctly.

Test 2: The evaluator will test the ability of the TOE to enable and disable MKA participants using the management function specified in the ST. The evaluator shall install pre-shared keys in devices B and C, and take any necessary additional steps to create corresponding MKA participants. The evaluator shall disable the MKA participant on device C, then observe that the TOE can communicate with B but neither the TOE nor B can communicate with device C. The evaluator shall re-enable the MKA participant of device B and observe that the TOE is now able to communicate with devices B and C.

(TD0652 is applied).

When enabling and disabling MKA participants using the management function specified in the ST, the evaluator observed that the TOE still behaved as expectedly.

Test 3: For TOEs using only PSKs, the TOE should be the Key Server in both tests and only one peer (B) needs to be tested. The tests are:

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

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

(TD0652 is applied).

The evaluator conducted the test and observed that the TOE behaved correctly when MACsec keys were rolled over.

Test 4: If “Cause Key Server to generate a new group CAK...” is selected, the evaluator shall connect to the PAE of the TOE, set the management function specified in the ST (e.g., set ieee8021XKeyCreateNewGroup to true), and observe that the TOE distributes a new group CAK.

(TD0652 is applied).

As “Cause Key Server to generate a new group CAK...” is not selected, this test is not relevant.

5.5.7 **FMT_SMR.2 Restrictions on security roles**

TSS

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 (Ref. [6]).

Accounts assigned to the Security Administrator role are used to manage Junos OS in accordance with [NDcPP v2.2e].

Guidance Documentation

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 (Ref. [6]).

The TOE is administered locally via the console port or remotely via SSH. The evaluator found the guidance document (Ref. [11]) to provide all the necessary instructions for administering the TOE both locally and remotely.

Tests

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 (Ref. [6]).

The evaluator used both the local console and SSH throughout the testing of the TOE. Therefore, this test is addressed by all other tests carried out by the evaluator.

5.6 Protection of the TSF (FPT)

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

TSS

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 (Ref. [6]).

As per Section 7.2.1, the TOE "does not provide a CLI interface to permit the viewing of keys. Cryptographic keys are protected through the enforcement of kernel-level file access rights, limiting access to the contents of cryptographic key containers to processes with cryptographic rights or shell users with root permission."

Guidance Documentation

None.

Tests

None.

5.6.2 FPT_APW_EXT.1 Protection of Administrator Passwords

TSS

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 (Ref. [6])

As per Section 7.5 of the ST, passwords are hashed when stored using hmac-sha1, sha256 or sha512. Authentication data for public key-based authentication methods (SSH) are in the filesystem are protected through the enforcement of kernel-level file access rights.

Guidance Documentation

None.

Tests

None

5.6.3 **FPT_TST_EXT.1 TSF Testing**

TSS

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 (Ref. [6]).

Section 7.5 of the ST describes the self-tests that the TOE runs. The description of the self-test is sufficient to demonstrate the correct operation of the TSF. Specifically, the self-tests ensure that only authorized executables are allowed to run thus ensuring the correct operation of the TOE.

Guidance Documentation

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 (Ref. [6]).

Chapter 9 of the guidance documentation (Ref. [11]) describes the self-tests that are present on the TOE and the possible errors that may result from these tests. These self-tests are executed as part of the TOE being configured to be in FIPS mode. These tests test the integrity of the TOE firmware and perform tests on the cryptographic algorithm implementations. Any failure of these tests results in a reboot of the TOE.

Tests

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 fulfil 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 (Ref. [6]).

The evaluator confirmed that when the TOE is rebooted, the FIPS self-tests, including firmware integrity and cryptographic functions self-tests, are performed during the reboot process.

5.6.4 **FPT_TUD_EXT.1 Trusted Update**

TSS

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 (Ref. [6]).

Section 7.5 of the ST, indicates that “Security Administrators are able to query the current version of the TOE firmware using the CLI command “show version” (FPT_TUD_EXT.1) and, if a new version of the TOE firmware is available, initiate an update of the TOE firmware. Junos OS does not provide partial updates for the TOE, customers requiring updates must migrate to a subsequent release. Updates are downloaded and applied manually (there is no automatic updating of the Junos OS). The installable firmware package containing the Junos OS has a digital signature that is checked when the Security Administrator attempts to install the package.”

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 (Ref. [6]).

Not applicable.

Guidance Documentation

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 (Ref. [6]).

The TOE does not support delayed activation. However, as per “Installing Software on the Device with Single Routing Engine” section in Chapter 2 of the CLI guide (Ref. [11]) the currently running version of the TOE can be queried via the show version command.

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 (Ref. [6]).

As per Chapter 1 of the Junos OS Software Installation and Upgrade Guide (Ref. [11]):

“Juniper Networks routing platforms run only binaries supplied by Juniper Networks, and currently do not support third-party binaries. Each Junos OS image includes a digitally signed manifest of executables that are registered with the system only if the signature can be validated. Junos OS will not execute any binary without a registered signature.”

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 (Ref. [6]).

The TOE does not use published hashes to protect the trusted update mechanism. As such, this requirement is not applicable.

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 user; it does not need to give information about the internal communication that takes place when applying updates (Ref. [6]).

The TOE is not in a distributed form. As such, this requirement is not applicable.

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 (Ref. [6]).

The TOE is not in a distributed form. As such, this requirement is not applicable.

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 (Ref. [6]).

The ST author does not indicate that a certificate-based mechanism is used for software update digital signature verification.

Tests

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 (Ref. [6]).

The evaluator executed the 'show version' command and confirmed that the TOE output the current version of the firmware. The evaluator loaded a legitimate update file onto the device via USB and, using the commands specified in the Installation and Upgrade Guide, which is referenced in the guidance documentation (Ref. [11]), confirmed that the TOE successfully installed the new firmware image. The TOE does not support delayed activation of updates.

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 (Ref. [6]).

The evaluator executed the 'show system version' command via the CLI and confirmed that it indicated a version different to that of the update file to be applied. The evaluator attempted to apply modified updates (modified via hex editor, unsigned firmware file or signed with an invalid development key) and confirmed that, in each instance, the TOE rejected the update file. The TOE does not support delayed activation of updates.

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 user 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 (Ref. [6]).

The TOE does not use published hashes to authenticate firmware updates. Hence this test is not applicable.

5.6.5 **FPT_STM_EXT.1 Reliable Time Stamps**

TSS

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 (Ref. [6]).

Section 7.1 of the ST explains that the clock function of Junos OS provides a source of date and time information for the appliance, used in audit timestamps, which is maintained using the hardware Time Stamp Counter as the clock source.

Guidance Documentation

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 (Ref. [6]).

Page 37 of the ECG (Ref. [11]) provides information on how to set the time. These instructions also disable NTP.

Tests

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 (Ref. [6]).

The evaluator manually set the time on the TOE using the operational guidance (Ref [11]). The evaluator waited for a period of 3 minutes and confirmed that the time is in agreement with respect to the value that was originally set. The evaluator confirmed that the TOE supports the direct setting of time by the security administrator. The time was maintained.

5.6.6 FPT_CAK_EXT.1 Protection of CAK Data

TSS

The evaluator shall examine the TSS to determine that it details how CAKs are stored and that they are unable to be viewed through an interface designed specifically for that purpose. If these values are not stored in plaintext, the TSS shall describe how they are protected or obscured (Ref. [6]).

Section 7.2.3 of the ST indicates that CAKs are store in encrypted form using AES with the system master password.

Guidance Documentation

There are no guidance activities for this requirement (Ref. [6]).

Tests

There are no test activities for this requirement (Ref. [6]).

5.6.7 FPT_FLS.1(2)/SelfTest Fail Secure with Preservation of Secure State

TSS

The evaluator shall examine the TSS to determine that it indicates that the TSF will shut down in the event that a self-test failure is detected. For TOEs with redundant failover capability, the evaluator shall examine the TSS to determine that it indicates that the failed components will shut down in the event that a self-test failure is detected.

(TD0190 is applied).

Section 7.5 of the ST indicates that when any self-test fails, the TOE halts in an error state and no command line input or traffic to any interface is processed. The TOE can only be returned to operation by rebooting it.

Note that the TOE does not have any redundant failover capability.

Guidance Documentation

The evaluator shall examine the operational guidance to verify that it describes the behavior of the TOE following a self-test failure and actions that an administrator should take if it occurs.

(TD0190 is applied).

Chapter 9 of the ECG (Ref. [11]) describes the behaviour of the TOE incase of self-test failure. It states that “If one of the KATs fail, the device panics and reboot continuously. The device can be recovered using USB install.”

Tests

The following test may require the vendor to provide access to a test platform that provides the evaluator with the ability to modify the TOE internals in a manner that is not provided to end customers:

Test 1: The evaluator shall modify the TSF in a way that will cause a self-test failure to occur. The evaluator shall determine that the TSF shuts down and that the behavior of the TOE is consistent with the operational guidance. The evaluator shall repeat this test for each type of self-test that can be deliberately induced to fail. For TOEs with redundant failover capability, the evaluator shall determine that the failed components shut down and the behavior of the TOE is consistent with the operational guidance. For each component, the evaluator shall repeat each type of self-test that can be deliberately induced to fail.

(TD0190 is applied).

The evaluator confirmed that when a self-test failure occurs, the TSF shut down and that the behaviour of the TOE is consistent with the operational guidance.

5.6.8 **FPT_RPL.1 Replay Detection**

TSS

The evaluator shall examine the TSS to determine that it describes how replay is detected for MPDUs and how replayed MPDUs are handled by the TSF (Ref. [6]).

Section 7.2.3 of the ST explains that the packet number (PN) field is used to protect against replayed packets. Whenever a packet is received that has a PN value lower than the minimum acceptable PN value that the TOE keeps, the MPDU is dropped.

Guidance Documentation

There are no guidance activities for this requirement (Ref. [6]).

Tests

The evaluator shall perform the following tests:

Before performing each test the evaluator shall successfully establish a MACsec 27 channel between the TOE and a MACsec-capable peer in the Operational Environment sending enough traffic to see it working and verify the PN values increase for each direction

Test 1: The evaluator shall set up a MACsec connection with an entity in the Operational Environment. The evaluator shall then capture traffic sent from this remote entity to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the remote entity

where the PN values in the SecTag of these packets are less than the lowest acceptable PN for the SA. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

The evaluator shall establish a MACsec connection between the TOE and a test system. The evaluator shall then capture traffic sent from test system to the TOE. The evaluator shall retransmit copies of this traffic to the TOE in order to impersonate the remote entity where the PN values in the SecTag of these packets are less than the lowest acceptable PN for the SA. The evaluator shall observe that the TSF does not take action in response to receiving these packets and that the audit log indicates that the replayed traffic was discarded.

Test 2: The evaluator will capture frames during a MKA session and record the lowest PN observed in a particular time range. The evaluator will then send a frame with a lower PN, and then verify that this frame is dropped. The evaluator will verify that the device logged this event (Ref. [6]).

Test 1:

The evaluator confirmed that the TOE discarded replayed MACsec frames and logged the events.

Test 2:

The evaluator confirmed that the TOE discarded MACsec frames that have PNs lower than the lowest acceptable PN and logged the events.

5.7 TOE Access (FTA)

5.7.1 **FTA_SSL_EXT.1 TSF-initiated Session Locking**

TSS

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 (Ref. [6]).

The Security Administrator can set the TOE so that a user session is terminated after a period of inactivity.

Guidance Documentation

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 (Ref. [6]).

The evaluator confirmed that the section “Configure Session Termination” on page 38 of the GD demonstrates how to configure the inactivity time period.

Tests

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 re-authentication is needed when trying to unlock the session (Ref. [6]).

The evaluator configured several idle timeout periods for the local console connection. The evaluators confirmed that, for each time period defined, the TOE terminated the session after the period of inactivity had expired. The evaluators confirmed that, once a session had been terminated, re-authentication was required before access to the TOE was restored.

5.7.2 **FTA_SSL.3 TSF-initiated Termination**

TSS

The evaluator shall examine the TSS to determine that it details the administrative remote session termination and the related inactivity time period (Ref. [6]).

The Security Administrator can set the TOE so that a user session is terminated after a period of inactivity.

Guidance Documentation

The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination (Ref. [6]).

The evaluator confirmed that the section “Configure Session Termination” on page 38 of the GD demonstrates how to configure the inactivity time period.

Tests

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 (Ref. [6]).

The evaluators configured several idle timeout periods for the remote SSH connection. The evaluators confirmed that, for each time period defined, the TOE terminated the session after the period of inactivity had expired. The evaluators confirmed that, once a session had been terminated, re-authentication was required before access to the TOE was restored.

5.7.3 **FTA_SSL.4 User-initiated Termination**

TSS

The evaluator shall examine the TSS to determine that it details how the local and remote administrative sessions are terminated (Ref. [6]).

User sessions (local and remote) can be terminated by users.

Guidance Documentation

The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session (Ref. [6]).

The evaluator confirmed that the section “Sample Output for User Initiated Termination” on page 40 of the guidance documentation demonstrates how to terminate a local or remote interactive session.

Tests

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 (Ref. [6]).

The evaluator established an administrative session via the local console. Once the session had been established, evaluators executed the ‘exit’ command and confirmed that the session was terminated.

The evaluator established an administrative session via the remote SSH. Once the session had been established, evaluators executed the ‘exit’ command and confirmed that the session was terminated.

5.7.4 **FTA_TAB.1 Default TOE Access Banners**

TSS

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) (Ref. [6]).

As per Section 7.6 of the ST, "Junos enables Security Administrators to configure an access banner for local and remote SSH connections provided with the authentication prompt. The banner can provide warnings against unauthorized access to the secure switch as well as any other information that the Security Administrator wishes to communicate."

Guidance Documentation

The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message (Ref. [6]).

As per Evaluated Configuration Guide (Ref. [11]) the login banner can be set via the set system login message login-message-banner-text command.

Tests

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 (Ref. [6]).

The evaluator configured a warning and consent message using the command specified in the guidance documentation. The evaluator confirmed that the configured message was displayed when connecting to the TOE via both local and remote administrative channels.

5.8 Trusted path/channels (FTP)

5.8.1 **FTP_ITC.1 Inter-TSF trusted channel**

TSS

The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each communications mechanism is identified in terms of the allowed protocols for that IT entity, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST (Ref. [6]).

As described in Section 7.2.2, the TOE provides an SSH server to support Trusted Channels using SSHv2 protocol which ensures the confidentiality and integrity of communication with the remote audit server. Export of audit information to a secure, remote server is achieved by setting up an event trace monitor that sends event log messages by using NETCONF over SSH to the remote system event logging server. This description matches with the cryptographic SFRs in the ST.

Guidance Documentation

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 (Ref. [6]).

The TOE utilises SSH for communication between itself and remote identities. External logging is supported are transferred over SSH to a remote server using NETCONF.

The Evaluated Configuration Guide (Ref. [11]) provides instructions for configuring SSH and the transfer of logs using NETCONF via SSH. In the event that the connections are broken, the TOE shall attempt to reconnect to the remote device.

Tests

The vendor 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 (Ref. [6]).

Testing of SSH is performed as part of other evaluation activities.

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 (Ref. [6]).

The evaluator confirmed by observing the traffic data using Wireshark that the TOE can initiate communication via the trusted channel (NETCONF over SSH) to send audit records to the syslog server and that the data is encrypted.

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 (Ref. [6]).

The evaluator established an SSH connection between the TOE and a peer before physically interrupting communications. The connection was able to be recovered after a physical interruption when the disconnection period is shorter than the application layer timeout. The connection is terminated by the TOE if the disconnection duration is greater than the application layer timeout. No data was transmitted during the periods of disconnection.

5.8.2 **FTP_TRP.1/Admin Trusted Path**

TSS

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 (Ref. [6]).

Section 7.2.2 of the ST indicates the TOE implements SSHv2 protocol as an administrative trusted path, in order to ensure the confidentiality and integrity of user remote sessions. This matches the cryptographic SFRs defined in the ST.

Guidance Documentation

The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method (Ref. [6]).

The TOE only supports SSH for establishing the remote administrative sessions. For SSH connections, a valid username and password or SSH key must be provided to access the TSF. The SSH client that is used must support the ciphers/key exchange methods used by the TOE in its evaluated configuration. This is confirmed by the Evaluated Configuration Guide (Ref. [11]).

Tests

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 (Ref. [6]).

Testing of remote administration via SSH is performed as part of other evaluation activities.

b) Test 2: The evaluator shall ensure, for each communication channel, the channel data is not sent in plaintext (Ref. [6]).

The evaluators performed Wireshark monitoring for SSH and confirmed that data sent via this channel was not transmitted in plaintext.

6 Evaluation Activities for SARs

This Section covers the evaluation activities for the Security Assurance Requirements (SARs) included in NDcPP v2.1.

6.1 ADV: Development

6.1.1 **Basic Functional Specification (ADV_FSP.1)**

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 3, 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’ (Ref. [6]).

Relevant TSFIs, per SD (Ref. [6]), are those used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates), as well as interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs).

According to the guidance documentation (Ref. [11]) and ST (Ref. [8]) the relevant TSFIs are the Junos CLI, which can be accessed by administrators either via a directly attached serial connection or remotely via SSH;

The purpose of these interface is clear from the ST and guidance documentation. Furthermore, the guidance documentation contains detailed instructions on how to administer the TOE via the CLI.

6.2 AGD: Guidance Documents

6.2.1 **Operational User Guidance (AGD_OPE.1)**

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 (Ref. [6]).

The evaluator was able to verify that there is a reasonable guarantee that administrators and users will be made aware of the existence and role of the documentation in maintaining an evaluated configuration.

The documentation relating to maintaining an evaluated configuration is made publicly available on Juniper's website.

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 (Ref. [6]).

The Evaluated Configuration Guide (Ref. [11]) requires the TOE to operate in FIPS mode and covers all Operational Environment that the product supports.

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 (Ref. [6]).

In order to satisfy the evaluated configuration, the TOE may only be operated in FIPS mode. When configured to operate in FIPS mode as per guide (Ref. [11]) a "fips" indicator shall be present on the CLI prompt. As per guide, once configured to operate in FIPS mode, the only means of not operating in FIPS mode is to zeroize the TOE. This will remove all CSPs and revert the device to factory setting.

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 (Ref. [6]).

The evaluator was able to ensure that the operational guidance was clear in stating the functionality and interfaces that were assessed and tested. The guide (Ref. [11]) covers the subjects of: authentication methods, administrator credentials and privileges, SSH and console connection, remote logging, audit and event logging options and the carrying out of self-tests on the TOE.

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 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:

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 (Ref. [6]).

a) In order to satisfy the evaluated configuration, the TOE may only be operated in FIPS mode. When configured to operate in FIPS mode as per guide (Ref. [11]) a “fips” indicator shall be present on the CLI prompt. As per guide, once configured to operate in FIPS mode, the only means of not operating in FIPS mode is to zeroize the TOE. This will remove all CSPs and revert the device to factory setting.

b)
1. The “Downloading Software Packages from Juniper Networks” section of the of the Evaluated Configuration Guide (Ref. [11]) provides instructions on how an update can be obtained. The “Installing Software on the Device with Single Routing Engine” section in Chapter 2 outlines the steps on how the update can be installed.
2. The Installation and Upgrade guide (referenced on Page 18 of Ref. [11]) provides detailed instructions on initiating the update process as well as examples on how to determine if the upgrade was successful or otherwise. The provided examples allow for the user to check the messages that are output on successful validation of the digital signature.

c) The Evaluated Configuration Guide (Ref. [11]) is clear in stating to the reader that it provides for “the steps required to duplicate the configuration of the device running Junos OS when the device is evaluated.” That is, only the security functionality that is covered by the guide is covered by the Evaluation Activities.

6.2.2 Preparative Procedures (AGD_PRE.1)

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) (Ref. [6]).

The evaluator was able to identify the following methods on how the administrator is able to verify that the operational environment can fulfil its role to support the security objectives of the TOE:

- OE.PHYSICAL: The MX10003 Universal Routing Platform Hardware Guide (Ref. [12]) addresses this in detail.

- OE.NO_GENERAL_PURPOSE: this requirement is met implicitly as the TOE is a dedicated router platform and does not permit the installation of any general-purpose software.

- OE.NO_THRU_TRAFFIC_PROTECTION: as stated in the Security Target (Ref. [8]) the TOE does not provide any protection of the traffic that traverses it.

- OE.TRUSTED_ADMIN: This requirement is met when the administrator configures the TOE according to the Evaluated Configuration Guide (Ref. [11]). Maintenance of the TOE (for e.g. the removal of expired certificates) is subject to the update procedures and frequency employed by the administrator.

- OE.UPDATES: The Evaluated Configuration Guide (Ref. [11]) and the Junos OS Software Installation and Upgrade Guide (referred in Ref. [11]) provide a great level of detail on how the administrator can provide updates to the TOE.

- OE.ADMIN_CREDENTIALS_SECURE: The procedures that are provided as part of the Evaluated Configuration Guide (Ref. [11]) ensure that the behaviour of the administrator and the overall security of the environment.

- OE.RESIDUAL_INFORMATION: The zeroisation procedures that are provided in the Evaluated Configuration Guide (Ref. [11]) support this objective.

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 (Ref. [6]).

The evaluator was able to find that the preparative procedures are provided for every Operational Environment that the product supports.

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment (Ref. [6]).

The evaluator was able to ensure that the Evaluated Configuration Guide (Ref. [11]) provided the procedures to successfully install the TSF in the supported Operating Environments.

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 (Ref. [6]).

The instructions provided in the Evaluated Configuration Guide (Ref. [11]) allowed for the security of the TSF to be managed as a component of the 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 (Ref. [6]).

a) The “Configuring Administrative Credentials and Privileges’ section of the Evaluated Configuration Guide (Ref. [11]) provides instructions on configuring protected administrative capability.
b) The TOE does not have default values for passwords. However, as per the MX10003 Universal Routing Platform Hardware Guide (Ref. [12]) the TOE requires the root password to be configured before any changes are made to the configuration. The Quick Start guide (referenced in [12]) provides instructions on how the root password may be set.

6.3 ALC: Life-cycle Support

6.3.1 **Labelling of the TOE (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 (Ref. [6]).

During testing, the evaluators issued a *show system information* command to the TOE in its operational mode. The output provided by the TOE indicated that the TOE’s model was MX10003, and Junos version was 22.2R1. This output was consistent with the expected TOE’s references.

6.3.2 **TOE CM coverage (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 (Ref. [6]).

The evaluators examined the Configuration List provided in Section 1.6.2 of the Security Target (Ref. [8]) to determine whether it includes the TOE and its associated evaluation evidence.

In addition to the TOE, the configuration list includes the following documentation:

- Common Criteria Configuration Guide for MX10003 Devices, Release 22.2R1

6.4 ATE: Tests

6.4.1 **Independent Testing – Conformance (ATE_IND.1)**

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 of [NDcPP-SD].

The evaluator should consult Appendix B of the [NDcPP-SD] when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation (Ref. [6]).

The evaluators conducted independent testing as per ATE_IND.1. The detailed description of the test cases and results are documented in the detailed test report (Ref. [9]) produced by the evaluators and submitted to the Australasian Certification Authority (ACA). The test report includes all the required tests specified in [NDcPP_SD] with all revisions specified by the relevant technical

decisions listed in the ST. A summary of the test cases can be found in Section 5 of this AAR. All independent tests passed. Some of the tests for cryptographic functionality of the TOE were carried out via CAVP-like test harness while others used ACVTS testing.

Testing was performed at Teron Labs' evaluation facility in Canberra under the oversight of the ACA. The test environment used by the evaluators is depicted in the diagram below.

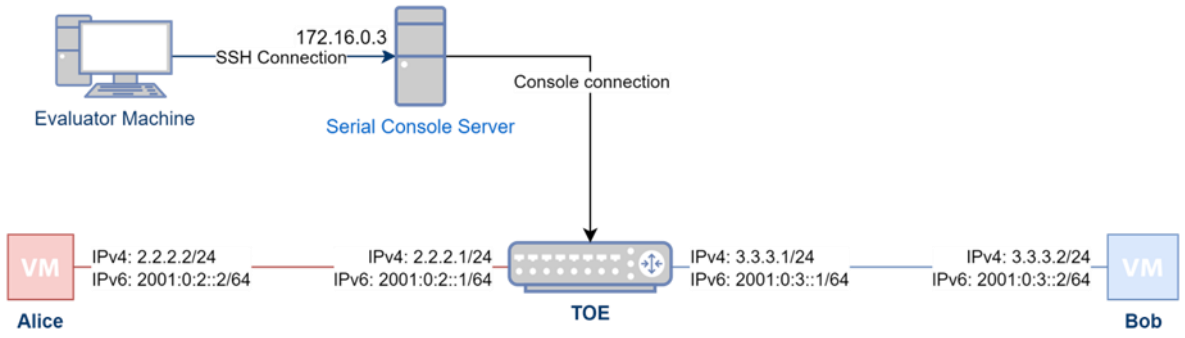

Figure 2 - Testing Environment

The test environment meets the requirements set out in [NDcPP-SD]. It consists of virtual machines Alice, Bob, a serial console server, the physical machine from which the TOE may be accessed via the console server, and the TOE itself. Alice is instantiated on the VMWare ESXi 6.7.0 server running on a Dell Poweredge R810 hardware. The network connection between the TOE and the VM is managed by a virtual switch that is also configured on the VMWare ESXi server on which Alice and Bob is hosted. Alice and Bob's prime function is to generate or instigate network traffic.

Identifier	Software	Hardware	Purpose
Alice	Kali Linux 2021.1	Dell Poweredge R810	• Network traffic generation • Syslog server
Bob	Kali Linux 2021.1	Dell Poweredge R810	• Network traffic generation • Syslog server
TOE	Junos 22.2R1	MX10003	• Router
Switch 1	Virtual Switch on VMWare ESX 6.7	Dell Poweredge R810	• Switch
Evaluator Machine	Windows 10 Professional	Dell Latitude 5590	• Console access • Report generation

Table 3 - Test Hardware

The software used in the test environment is considered in the table below.

Name	Type	Source
Junos 22.2R1	TOE Operating System	Vendor
Scapy 2.4.4	Packet generation software	Kali Linux (pre-installed)
Python 3.9	Language run-time (used by Scapy)	Kali Linux (pre-installed)
Wireshark 3.4.4	Packet sniffing software	Kali Linux (pre-installed)
ESXi-6.0.0-20200204001-standard	Virtualisation Host	Internet

Table 4 - Test Software

6.5 AVA: Vulnerability Assessment

6.5.1 **Vulnerability Survey (AVA_VAN.1)**

The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

The developer shall provide documentation identifying the list of software and hardware components that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside of the TOE), for example a web server, protocol or cryptographic libraries, (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components and will be used by the evaluators in formulating vulnerability hypotheses during their analysis.

The evaluator formulates hypotheses in accordance with process defined in Appendix A of [NDcPP-SD]. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3 of [NDcPP_SD]. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2 of [NDcPP_SD]. The results of the analysis shall be documented in the report according to Appendix A.3 of [NDcPP_SD]. (Ref. [6]).

The evaluator followed the flaw hypothesis methodology. Accordingly, four types of flaw hypotheses have been considered.

Type 1 Hypotheses – Public-Vulnerability-Based (the Technical Decision TD0564 is applied)

The evaluators shall perform a search on the sources listed in Section A.4 to determine a list of potential flaw hypotheses that are specific to the TOE and its components.

The search terms used were as follows:

- MX10003
- RE-S-1600x8
- JNP-MIC1-MACSEC
- EX9253
- EX9253-RE2
- EX9253-6Q12C-M
- bcm82391
- Junos OS 22.2R1
- Quicksec 7.0
- OpenSSL 1.1.1
- LibMD Open SSL 1.1.1
- FreeBSD 12.1-Stable
- MACsec Junos

The relevant public vulnerabilities are listed in Table 5 - Public Vulnerabilities with some analysis. As the consequence of the analysis, there are not vulnerability tests are added to this evaluation.

Type 2 Hypotheses – iTC-Sourced

No Type 2 hypotheses have been defined by the iTC for any the cPPs involved.

Type 3 Hypotheses – Evaluation-Team-Generated

As per the NDcPP SD, Type 3 flaws are formulated by the evaluator based on information presented by the product (through on-line help, product documentation and user guides, etc.) and product behaviour during the (functional) testing activities.

During functional testing of the TOE, the evaluators have not observed any behavior that would point to anomalous functionality or vulnerability. Similarly, the evaluators have not found elements in the product documentation that would be indicative of potential vulnerabilities, beyond what was already explored in the conducted Type 1 survey.

Nonetheless, based on behavior observed in previous evaluations of Juniper devices performed by the evaluators, the evaluators have decided to conduct the following tests:

- Vulnerability scanning by Nessus

- Race condition check: the evaluator shall attempt to take advantage of any race conditions in the TOE's boot-up process. This shall involve the evaluator attempting to access the system via the serial console without providing a password.

Additionally, the firmware shall be analysed to determine whether any sensitive information has been included, such as private keys or unsecured superuser accounts.

Type 4 Hypotheses – Tool-Generated

The following protocol fuzz vulnerability testing has been considered by the evaluators:

- Examine the effects of sending mutated packets carrying each 'Type' and 'Code' value that is undefined in the relevant RFC for each of ICMPv4 (RFC 792) and ICMPv6 (RFC 4443).
- Examine the effects of mutated packets carrying each 'Transport Layer Protocol' value that is undefined in the respective RFC for IPv4 (RFC 791) IPv6 (RFC 2460) should also be covered if it is supported and claimed by the TOE.
- Examine the effect of fuzzing the remaining fields in the required protocol headers.

Vuln ID	Publication Date	CVSS Severity	Analysis
CVE-2021-0290	July 15, 2021	*V3.1:* 6.5 MEDIUM *V2.0:* 3.3 LOW	Not applicable to this TOE version
CVE-2022-1292	May 03, 2022	*V3.1:* 9.8 CRITICAL *V2.0:* 10.0 HIGH	Vendor has confirmed that it is not applicable because script (c_rehash) that accompanies OpenSSL but is not shipped with Junos.
CVE-2022-2097	July 05, 2022	*V3.1:* 5.3 MEDIUM *V2.0:* 5.0 MEDIUM	AES OCB mode is out of the scope
CVE-2022-2068	June 21, 2022;	*V3.1:* 9.8 CRITICAL *V2.0:* 10.0 HIGH	Vendor has confirmed that it is not applicable because script (c_rehash) that accompanies OpenSSL but is not shipped with Junos.
CVE-2019-0190	January 30, 2019	*V3.1:* 7.5 HIGH *V2.0:* 5.0 MEDIUM	HTTP is out of the scope
CVE-2020-7463	March 26, 2021	*V3.1:* 5.5 MEDIUM *V2.0:* 4.9 MEDIUM	SCTP is out of the scope
CVE-2020-7461	March 26, 2021	*V3.1:* 7.3 HIGH *V2.0:* 7.5 HIGH	DHCP is out of the scope
CVE-2020-7460	August 06, 2020	*V3.1:* 7.0 HIGH *V2.0:* 4.4 MEDIUM	The flaw is exploitable at least only above Basic Attack Potential as an additional malicious program is required to run on the TOE.
CVE-2020-7459	August 06, 2020	*V3.1:* 6.8 MEDIUM *V2.0:* 4.6 MEDIUM	The exploitation requires a malicious USB device plugged into the system, which is out of scope as per A.PHYSICAL_PROTECTION,

			A.TRUSTED_ADMINSTRATOR and A.ADMIN_CREDENTIALS_SECURE assumptions in the Security Target.
CVE-2020-7458	July 09, 2020	*V3.1:* 9.8 CRITICAL *V2.0:* 7.5 HIGH	The flaw is exploitable at least only above Basic Attack Potential. The evaluator has not been able to find the affected services on the TOE.
CVE-2020-7457	July 09, 2020	*V3.1:* 8.1 HIGH *V2.0:* 6.8 MEDIUM	The flaw is exploitable at least only above Basic Attack Potential, as a malicious application is required on run on the TOE. That condition is not easy to be satisfied under A.PHYSICAL_PROTECTION, A.TRUSTED_ADMINSTRATOR and A.ADMIN_CREDENTIALS_SECURE assumptions in the Security Target.
CVE-2020-7456	June 09, 2020	*V3.1:* 6.8 MEDIUM *V2.0:* 7.2 HIGH	The exploitation requires a malicious USB device plugged into the system, which is out of scope as per A.PHYSICAL_PROTECTION, A.TRUSTED_ADMINSTRATOR and A.ADMIN_CREDENTIALS_SECURE assumptions in the Security Target.
CVE-2020-7455	May 13, 2020;	*V3.1:* 7.5 HIGH *V2.0:* 5.0 MEDIUM	FTP connections to and from the TOE, the affected service, is out of scope.
CVE-2020-7454	May 13, 2020;	*V3.1:* 9.8 CRITICAL *V2.0:* 7.5 HIGH	Vendor has confirmed that it is not applicable, as the TOE does not use any libalias functionality
CVE-2019-15880	May 13, 2020;	*V3.1:* 9.8 CRITICAL *V2.0:* 7.5 HIGH	The cryptodev module, the affected library, has not been found by the evaluator on the TOE. Moreover, the vendor has described that the cryptographic functionalities on the TOE are handled by other libraries. Also, as per the advisor at https://www.freebsd.org/security/advisories/FreeBSD-SA-20:15.cryptodev.asc, this library is rarely used.
CVE-2019-15879	May 13, 2020;	*V3.1:* 7.4 HIGH *V2.0:* 5.8 MEDIUM	The cryptodev module, the affected library, has not been found by the evaluator on the TOE. Moreover, the vendor has described that the cryptographic functionalities on the TOE are handled by other libraries. Also, as per the advisor at https://www.freebsd.org/security/advisories/FreeBSD-SA-20:15.cryptodev.asc, this library is rarely used.
CVE-2019-15878	May 13, 2020;	*V3.1:* 7.8 HIGH *V2.0:* 4.6 MEDIUM	SCTP, the affected protocol, is out of scope
CVE-2020-7453	April 28, 2020; 8	*V3.1:* 6.0 MEDIUM *V2.0:* 3.3 LOW	The issue is exploitable only with a *superuser* access, which is assumed to be trusted and trustworthy as per A.TRUSTED_ADMINSTRATOR

			and A.ADMIN_CREDENTIALS_SECURE assumptions in the Security Target.
CVE-2020-7452	April 28, 2020	*V3.1:* **9.1 CRITICAL** *V2.0:* **9.0 HIGH**	As per the advisory at https://security.FreeBSD.org/advisories/FreeBSD-SA-20:07.epair.asc , the issue is exploitable only with a root access, which is assumed to be trusted and trustworthy as per A.TRUSTED_ADMINSTRATOR and A.ADMIN_CREDENTIALS_SECURE assumptions in the Security Target.
CVE-2019-5614	April 28, 2020	*V3.1:* **9.8 CRITICAL** *V2.0:* **7.5 HIGH**	The vendor has confirmed that it is not applicable not applicable, as ipfw is not used in Junos
CVE-2019-15874	April 28, 2020	*V3.1:* **9.8 CRITICAL** *V2.0:* **7.5 HIGH**	The vendor has confirmed that it is not applicable not applicable, as ipfw is not used in Junos
CVE-2020-7451	April 28, 2020	*V3.1:* **5.3 MEDIUM** *V2.0:* **5.0 MEDIUM**	The vendor has confirmed that it is not applicable
CVE-2019-15877	April 28, 2020;	*V3.1:* **5.5 MEDIUM** *V2.0:* **2.1 LOW**	The flaw is exploitable at least only above Basic Attack Potential, as it requires an attacker run a malicious executable on the TOE.
CVE-2019-15876	April 28, 2020;	*V3.1:* **5.5 MEDIUM** *V2.0:* **2.1 LOW**	The flaw is exploitable at least only above Basic Attack Potential, as it requires an attacker run a malicious executable on the TOE.
CVE-2020-7450	February 18, 2020;	*V3.1:* **9.8 CRITICAL** *V2.0:* **7.5 HIGH**	This vulnerability is out of scope as the TOE is not supposed to fetch any untrustworthy URLs.
CVE-2019-15875	February 18, 2020;	*V3.1:* **3.3 LOW** *V2.0:* **2.1 LOW**	The flaw is exploitable at least only above Basic Attack Potential, as it requires an attacker to have access to a core dump file and try to find the leaked data, it is not clear how harmful the leaked data is.
CVE-2018-0021	April 11, 2018	*V3.0:* **8.8 HIGH** *V2.0:* **3.3 LOW**	Not applicable to this TOE version
CVE-2017-2342	July 17, 2017	*V3.0:* **8.1 HIGH** *V2.0:* **4.3 MEDIUM**	Not applicable to this TOEplatform

Table 5 - Public Vulnerabilities

7 Glossary

Acronym/Term	Description
AAR	Assurance Activity Report
ACA	Australian Certification Authority
AES	Advanced Encryption Standard
AISEF	Australian Information Security Evaluation Facility
ANSI	American National Standards Institute
CA	Certificate Authority
CAVP	Cryptographic Algorithm Validation Program
CAVS	Cryptographic Algorithm Validation System
CBC	Cipher Block Chaining
CLI	Command Line Interface
CMAC	Cipher-based Message Authentication Code
cPP	Collaborative Protection Profile
CRL	Certificate Revocation List
CSP	Critical security parameter
DH	Diffie Hellman
DRBG	Deterministic Random Bit Generator
DSA	Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EP	Extended Package
ESP	Encapsulating Security Payload
FFC	Finite Field Cryptography
FIPS 140-2	Federal Information Processing Standard 140-2
FTP	File Transfer Protocol
GCM	Galois Counter Mode
HMAC	Hash-based Message Authentication Code
ICMP	Internet Control Message Protocol

ICMPv6	Internet Control Message Protocol version 6
IDP	Intrusion Detection and Prevention
IPS	Intrusion Prevention System
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
IPv6	Internet Protocol version 6
MACsec	Media Access Control Security
NAT	Network Address Translation
NDcPP	Network Device collaborative Protection Profile
NTP	Network Time Protocol
RFC	Request for Comment
RIP	Routing Information Protocol
RNG	Random Number Generator
RSA	Rivest-Shamir-Adleman
SA	Security Association
SHA	Secure Hash Algorithm
SHAVS	Secure Hash Standard Validation System
SFR	Security Functional Requirement
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SSH	Secure Shell
ST	Security Target
TCP	Transmission Control Protocol
TSF	TOE Security Functionality
TSFI	TSF Interface
TOE	Target of Evaluation
UDP	User Datagram Protocol
VPN	Virtual Private Network