Assurance Activity Report

Junos 22.2R1 for SRX Series

Document Reference: EFT-T022-AAR 1.0
Document Status: Release
Release Date: 22-Dec-2023
Customer: Juniper Networks
167 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
1.0	5-Dec-23	C. Randall	Initial Release.

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] PP-Module for Virtual Private Network (VPN) Gateways, Version: 1.1, 18 June 2020

[8] Supporting Document, Mandatory Technical Document, PP-Module for Virtual Private Network (VPN) Gateways, version 1.1, dated 2020-06-18

[9] PP-Module for Stateful Traffic Filter Firewalls, Version 1.4 + Errata 20200625, 25-June-2020

[10] Evaluation Activities for Stateful Traffic Filter Firewalls PP-Module, Version 1.4 +Errata 20200625, June-2020

[11] PP-Module for Intrusion Prevention Systems (IPS), Version 1.0, 11-May-2021

[12] Supporting Document, Mandatory Technical Document, PP-Module for Intrusion Prevention Systems (IPS), Version: 1.0, 2021-05-11

2.2 Evaluation Evidence

[13] Security Target for Junos OS 22.2R1 for SRX Series, Version 1.2, 11-Oct-2023

[14] Test Report: NDcPP (EFT-T022-TR-NDcPP), Version 1.1, 27-Sep-2023

[15] Test Report: MOD_VPNGW (EFT-T022-TR-MOD_VPNGW), Version 1.0, 28-June-2023

[16] Test Report: MOD_FW (EFT-T022-TR-MOD_FW), Version 1.0, 28-June-2023

[17] Test Report: MOD_IPS (EFT-T022-TR-MOD_IPS), Version 1.0, 28-June-2023

[18] Junos OS, Common Criteria Guide for SRX300, SRX320, SRX340, SRX345, SRX345-DUAL-AC and SRX380 Devices, Release 22.2R1, Published 2022-10-06

[19] Junos OS, Common Criteria Guide for SRX1500, SRX4100, SRX4200, and SRX4600 Devices, Release 22.2R1, Published 2022-10-07

[20] Junos OS, Common Criteria Guide for SRX5400, SRX5600, and SRX5800 Devices, Release 22.2R1, Published 2022-10-07

[21] Junos OS, Intrusion Detection and Prevention User Guide, Published 2022-01-20

[22] Junos OS, Flow-Based and Packet-Based Processing User Guide for Security Devices, Published 2021-12-14

[23] IPsec VPN User Guide for Security Devices, Published 2021-12-14

[24] Junos OS CLI User Guide, Published 2021-12-14

3 Introduction

This report documents the assurance activities performed by Teron Labs as part of the Common Criteria evaluation of Junos 22.2R1 for SRX Series 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
[MOD_VPNGW]	PP-Module for Virtual Private Network (VPN) Gateways, version 1.1 dated 18 June 2020
[MOD_FW]	PP-Module for Stateful Traffic Filter Firewalls, Version 1.4 + Errata 20200625 dated 25 June 2020
[MOD_IPS]	PP-Module for Intrusion Prevention Systems (IPS), Version 1.0 dated 11 May 2021

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

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

3.1 Evaluation Identifiers

Task Identifier	EFT-T022
TOE Name	Junos 22.2R1 for SRX Series
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. PP-Module for Virtual Private Network (VPN) Gateways, Version: 1.1, 18 June 2020

	PP-Module for Stateful Traffic Filter Firewalls, Version 1.4 + Errata 20200625, 25-June-2020 PP-Module for Intrusion Prevention Systems (IPS), Version 1.0, 11-May-2021
CC Version	3.1 Revision 5

3.2 ST Identifier

ST Title	Security Target Junos 22.2R1 for SRX Series
ST Version	1.2
ST Date	11-October-2023

3.3 TOE Overview

The Target of Evaluation (TOE) is Juniper Networks, Inc. Junos OS 22.2R1 operating system on Services Gateway appliances SRX Series. The TOE can operate in single mode or in cluster mode, also known as high availability (HA). In cluster mode, a pair of devices are connected and configured to operate like a single device to provide high availability. When configured as a chassis cluster, the two nodes back up each other, with one node acting as the primary device and the other as the secondary device, ensuring stateful failover of processes and services in the event of system or hardware failure. If the primary device fails, the secondary device takes over processing of traffic. Further details of the Cluster Mode configuration are provided below.

The TOE supports definition and enforcement of information flow policies among network nodes. The TOE implements stateful inspection of every packet that traverses the network and provides a central point of control to manage the network security policy.

All information flows from one network node to another pass through an instance of the TOE. Information flow is controlled on the basis of network node addresses, protocol, type of access requested, and services requested. In support of the information flow security functions, the TOE ensures that security-relevant activity is audited and that the TOE functions are protected from potential attacks. The TOE also provides tools to manage all security functions.

The TOE also implements multi-site virtual private network (VPN) gateway and Intrusion Prevention System functionalities, capable of monitoring information flows to detect potential attacks based on pre-defined attack signature and anomaly characteristics in the traffic.

Each TOE is a security product that supports a variety of high-speed interfaces for medium/large networks and network applications. Juniper Networks routers share common Junos firmware, features, and technology for compatibility across platforms.

Each TOE is physically self-contained. They house all software, firmware and hardware necessary to perform all functions. The hardware consists of two major components: the Services Gateway appliance itself and various PIC/PIMs which allow the appliances to

communicate with the different types of networks that may be required within the environment where the Services Gateway appliances are used.

Each instance of the TOE consists of the following major architectural components:

- The Routing Engine (RE) runs the Junos firmware and provides Layer 3 routing services and network management for all operations necessary for the configuration and operation of the TOE. The RE also controls the flow of information through the TOE, including Network Address Translation (NAT) and all operations necessary for the encryption/decryption of packets for secure communication via the IPsec protocol.
- The Packet Forwarding Engine (PFE) provides all operations necessary for transit packet forwarding.

The Routing Engine and Packet Forwarding Engine 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 Services Gateway appliances support numerous routing standards for flexibility and scalability as well as IETF SSHv2 and IPsec protocols. These functions can all be managed through the Junos firmware, either from a connected terminal console or via a network connection. Network management can be secured using IPsec and SSH protocols. All management, whether from a user connecting to a terminal or from the network, requires successful authentication.

The TOE supports intrusion detection and prevention functionality, which allows it to detect and react to potential attacks in real time. The detection component of the IPS can be based on attack signatures which specify the characteristics of the potentially malicious traffic based on a variety of packet header and payload data attributes. Anomaly detection based on deviation of the monitored traffic from expected values is also supported.

Services Gateway appliances accomplish routing through a Virtual Router (VR) mechanism. The TOE can divide its routing component into two or more VRs with each VR maintaining its own list of known networks in the form of a routing table, routing logic, and associated security zones.

In the evaluated deployment the TOE is managed and configured via Command Line Interface, either via a directly connected console or using SSH connections, which can be further protected using IPsec if required.

3.3.1 **Physical boundary**

There are two variants of the TOE hardware: SRX1500, SRX4100, SRX4200 and SRX4600 use a virtualised architecture where the hardware is abstracted through a virtual platform. The SRX300, SRX320, SRX340, SRX345, SRX345-DUAL-AC, SRX5400, SRX5600 and SRX5800 are bare metal variants where the OS kernel interacts directly with the hardware.

The physical boundary of the virtualised TOE architecture illustrated in Figure 1 is the entire chassis of the TOE. It includes both the hardware and the firmware of the TOE. The virtualised variant of the TOE is the Junos OS 22.2R1 firmware running on the SRX1500, SRX4100, SRX4200 and SRX4600 chassis summarised in Table 1. The physical boundary of the TOE includes the firmware implementing the KVM Hypervisor, the firmware implementing the Routing

Engine and the Packet Forwarding Engine. The TOE is contained within the physical boundary of the appliance chassis.

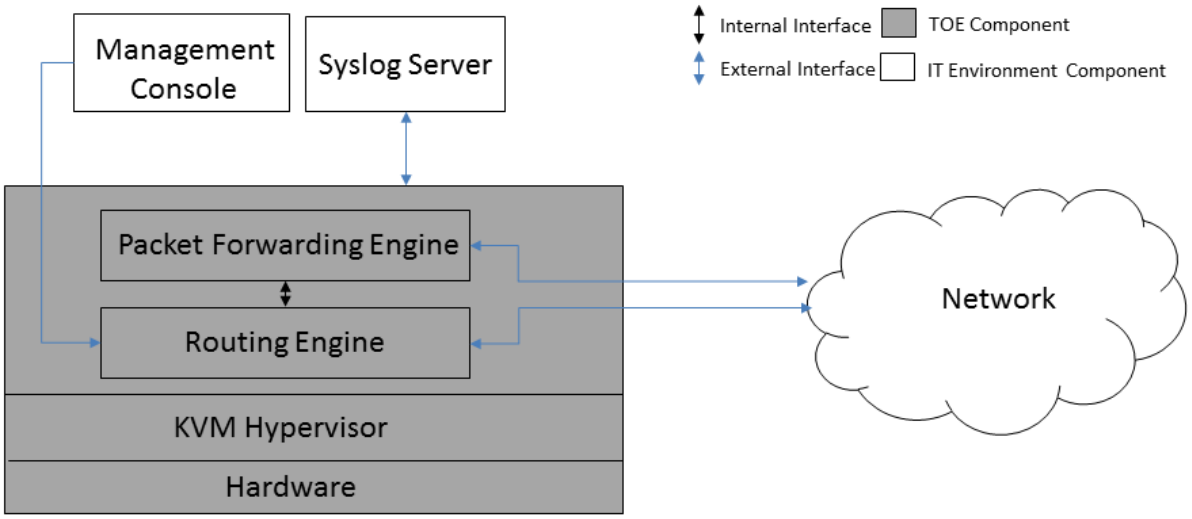

Figure 1 TOE Physical boundary for virtualised architecture

Model	Network Ports	Firmware
SRX1500	• 12x1Gb Ethernet LAN ports (RJ-45) • 4x1Gb Ethernet LAN ports (SFP) • 4x10Gb Ethernet LAN ports (SFP+) • 1x1Gb Out-of-band management port • 1x1Gb dedicated high availability port (SFP) • 2 PIM slots • 1 console (RJ45 + mini-USB) port • 1 USB 2.0 port (Type A)	Junos OS 22.2R1
SRX4100	• 8x1Gb/10Gb Ethernet LAN ports (SFP+) • 1x1Gb Out-of-band management port • 2x1Gb/10Gb dedicated high availability port (SFP/SFP+) • 1 console (RJ45) port • 2 USB 2.0 ports (Type A)	Junos OS 22.2R1

Model	Network Ports	Firmware
SRX4200	• 8x1Gb/10Gb Ethernet LAN ports (SFP+) • 1x1Gb Out-of-band management port • 2x1Gb/10Gb dedicated high availability port (SFP/SFP+) • 1 console (RJ45) port • 2 USB 2.0 port (Type A)	Junos OS 22.2R1
SRX4600	• 4x40Gb/100Gb Ethernet LAN ports (QSFP28) • 8x1Gb/10Gb Ethernet LAN ports (SFP+) • 1x1Gb Out-of-band management port • 4x1Gb/10Gb dedicated high availability port (SFP+) • 1 console (RJ45) port • 1 USB 2.0 port (Type A)	Junos OS 22.2R1

Table 1 TOE Physical Boundary Details for SRX1500, SRX4100, SRX4200 and SRX4600 models

Further information about the PIM slots available for the SRX1500 is given in Table 2.

Interface Model number	Description
SRX-MP-1SERIAL-R	Serial Mini-PIM provides physical connection for serial setup
SRX-MP-1T1E1-R	T1/E1 Mini-PIM provides the physical connection to T1 or E1 network media types
SRX-MP1VDSL2-R	VDSL2 Mini-PIM is part of the xDSL family of modem technologies, which provide faster data transmission over a single flat untwisted or twisted pair of copper wires
SRX-MP-LTE-AE (North America, EU) SRX-MP-LTE-AA (Asia, Australia)	Mini-PIM providing wireless WAN support. The Mini-PIM contains an integrated modem and operates over 3G and 4G networks. The Mini-PIM supports up to two SIM cards and can be installed in any of the Mini-PIM slots on the services gateways

Table 2 PIM details

The physical boundary of the bare metal TOE architecture illustrated in Figure 2 is the entire chassis of the TOE. It includes both the hardware and the firmware of the TOE. The bare metal variant of the TOE is the Junos OS 22.2R1 firmware running on the SRX300, SRX320, SRX340, SRX345, SRX345-DUAL-AC, SRX5400, SRX5600 and SRX5800 chassis summarised in Table 4. The physical boundary of the bare metal TOE architecture includes the firmware implementing the Routing Engine and the Packet Forwarding Engine. The TOE is contained within the physical boundary of the appliance chassis.

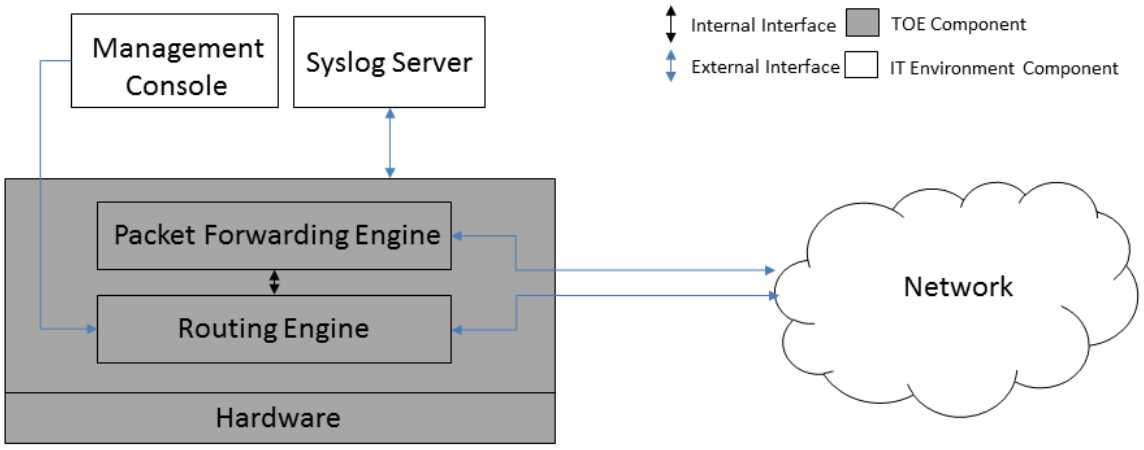

Figure 2 TOE Physical Boundary for the bare metal architecture

The TOE interfaces are comprised of the Network interfaces which pass traffic, and Management interface through which the administrative actions are handled.

Model	Network Ports	Firmware
SRX300	• Six Gigabit Ethernet LAN ports • Two 1G SFP ports	Junos OS 22.2R1
SRX320	• Two Mini PIM slots • Six Gigabit Ethernet LAN ports • Two 1G SFP ports	Junos OS 22.2R1
SRX340	• Four Mini PIM slots • Eight Gigabit Ethernet LAN ports • Eight 1G SFP ports • One Management RJ45 port	Junos OS 22.2R1
SRX345	• Four Mini PIM slots • Eight 1Gb Ethernet LAN ports (RJ-45) • Eight 1Gb Ethernet LAN ports (SFP) • No 10Gb SFP+ ports • One Management RJ45 port + mini-USB • One USB 3.0 port	Junos OS 22.2R1

Model	Network Ports	Firmware
SRX345-DUAL-AC[1]	• Four Mini PIM slots • Eight 1Gb Ethernet LAN ports (RJ-45) • Eight 1Gb Ethernet LAN ports (SFP) • No 10Gb SFP+ ports • One Management RJ45 port + mini-USB • One USB 3.0 port	Junos OS 22.2R1

Table 3 TOE Physical Boundary Details for SRX300, SRX320, SRX340, SRX345, and SRX345-DUAL-AC models

Model	Line Card Configuration	Firmware
SRX5400	• Four card cage slots: • one SCB • up to 3 SPCs or MPCs or IOCs	Junos OS 22.2R1
SRX5600	• Eight card cage slots: • 2 SCBs • up to 6 SPCs or MPCs or IOCs	Junos OS 22.2R1
SRX5800	• Fourteen card cage slots: • 2 or 3 SCBs • up to 12 SPCs or MPCs or IOCs	Junos OS 22.2R1

Table 4 TOE Physical Boundary Details for SRX5400, SRX5600 and SRX5800 models

The line cards and modules supported for by the SRX5400, SRX5600 and SRX5800 Services Gateways are described in the following reference documents:

- SRX5400 Services Gateway Hardware Guide
- SRX5600 Services Gateway Hardware Guide
- SRX5800 Services Gateway Hardware Guide

The line cards and modules listed in Table 5 through to Table 8 below are included in the scope of TOE.

[1] SRX345-DUAL-AC is a dual power supply version of SRX345. Identifier SRX345-DUAL-AC is as returned by the user command `show chassis hardware`.

Interface Model number	Description
SRX5K-SPC3 Services Processing Card	Contains two Services Processing Units (SPUs), which provide the processing power to run integrated services such as firewall, Ipsec, and IDP

Table 5 Services processing cards (SPCs)

Interface Model number	Description
Modular Port Concentrator (SRX5K-MPC)	Interface card with two slots that accept MICs
MIC with 20x1GE SFP Interfaces	Installed in the SRX-5K MPC to add twenty 1-Gigabit Ethernet small form-factor pluggable (SFP) Ethernet ports
MIC with 10x10GE SFP+ Interfaces	Installed in an MPC to add ten 10-Gigabit Ethernet SFP+ ports
MIC with 1x100GE CFP Interface	Installed in an MPC to add one 100-Gigabit Ethernet CFP port
MIC with 2x40GE QSFP+ Interfaces	Installed in an MPC to add two 40-Gigabit quad small form-factor pluggable (QSFP+) Ethernet ports
SRX5K-MPC3-40G10G	Provides 10 Gigabit Ethernet and 40 Gigabit Ethernet ports, with a Packet Forwarding Engine that provides a 240 Gbps line rate
SRX5K-MPC3-100G10G	Provides 100 Gigabit Ethernet and 10 Gigabit Ethernet ports, with a Packet Forwarding Engine that provides a 240 Gbps line rate
SRX5K-IOC4-10G	Provides 10 Gigabit Ethernet ports, with at Packet Forwarding Engine that provides 400-Gbps line rate
SRX5K-IOC4-MRAT	Provides 10 Gigabit Ethernet, 40 Gigabit Ethernet and 100 Gigabit Ethernet ports, with a Packet Forwarding Engine that provides a 480 Gbps line rate

Table 6 Modular Port Concentrators (MPCs) and Interface Cards (IOCs)

Interface Model number	Description
SRX5K-SCB3 Switch Control Board	Provides: • Gigabit Ethernet switch that is connected to the embedded CPU complex on all components • Switch fabric to provide the switching functions for the MPCs • Slot for RE
SRX5K-SCB4 Switch Control Board[2]	Provides • improved fabric performance and bandwidth capabilities for high-capacity line cards • Slot for RE

Table 7 Switch Control Boards (SCBs)

Interface Model number	Description
SRX5K-RE3-128G	Software processes that run on the Routing Engine maintain the routing tables, manage the routing protocols used on the device, control the device interfaces, control some chassis components, and provide the interface for system management and user access to the device

Table 8 Routing Engine (RE)

The install packages for the TOE software are the following:

- junos-srxsme-22.2R1.9.tgz
- junos-srxentedge-x86-64-22.2R1.9.tgz
- junos-srxmr-x86-64-22.2R1.9.tgz
- junos-srxhe-x86-64-22.2R1.9.tgz
- junos-vmhost-install-srx-x86-64-22.2R1.9.tgz

The guidance document included as part of the TOE is [18][19][20].

3.3.2 Cluster Mode Configuration

The Administrator of the TOE can set up the Cluster Mode for High Availability (HA). The two hosts constituting a chassis cluster must have identical configuration except for one being configured to node 0 and the other to node 1. The two nodes are connected via two links: the HA

[2] Only supported on SRX5600 and SRX5800 models.

control link and the HA fabric link. An example configuration of the TOE in Cluster Mode for high availability is depicted in Figure 3. Further details are given in [18][19][20].

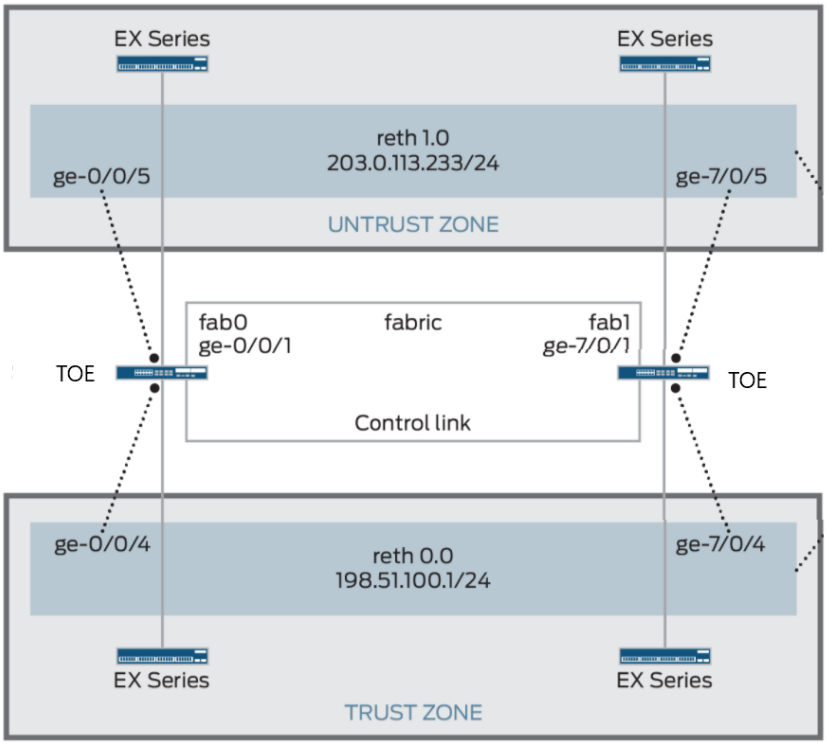

Figure 3 Example of the HA configuration of the TOE

Critical security parameter data sent over the control link between the two chassis in Cluster Mode is protected using IPsec from active and passive eavesdropping. Without the internal IPsec key, an attacker cannot gain privilege access or observe traffic.

Note the scope of the TOE includes both cluster mode and non-cluster mode. In other words, the security functionality of the TOE has been evaluated both when the TOE is configured in cluster mode, connected to a secondary or primary node, as well as when the TOE operates in single (non-cluster) mode.

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 MOD_VPNGW SD (Ref. [8]). The evaluators used the demo ACVTS environment to validate cryptographic capabilities of the branch platforms (SRX300, SRX320, SRX345, SRX345-DUAL-AC), and production ACVTS environment for the generation of test vectors and verification of responses for all other platforms. 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 CBC (128, 192, 256 bits), GCM (128, 256 bits)	FCS_COP.1/DataEncryption	Encryption/Decryption for IKE
AES CBC, GCM (128, 192, 256 bits)	FCS_COP.1/DataEncryption	Encryption/Decryption for IPsec
RSA Digital Signature Algorithm (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
RSA Digital Signature Algorithm (2048 bits)	FCS_COP.1/SigGen	IKE Authentication (signature generation and verification)
EC Digital Signature Algorithm (ECDSA) (P-256, P384, P-521)	FCS_COP.1/SigGen	IKE Authentication (signature generation and verification)
SHA-1, SHA2-256, SHA2-384, SHA2-512	FCS_COP.1/Hash	SSH Hashing
SHA-1, SHA2-256, SHA2-384, SHA2-512	FCS_COP.1/Hash	IKE Hashing
SHA-1, SHA2-256, SHA2-384, SHA-512	FCS_COP.1/Hash	IPsec 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-SHA2-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512	FCS_COP.1/KeyedHash	IKE Keyed Hashing

Algorithm	SFR	Usage
HMAC-SHA2-1, HMAC-SHA2-256, HMAC-SHA2-384, HMAC-SHA2-512	FCS_COP.1/KeyedHash	IPsec 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
RSA Key Generation (2048 bits)	FCS_CKM.1/IKE	RSA Key Generation for IKE
ECC Key Generation (P-256, P-384, P-521)	FCS_CKM.1/IKE	ECC Key Generation for IKE KAS
ECC KAS (P-256, P-384, P-521)	FCS_CKM.2	SSH KAS
FFC KAS (DH Group 14, 19, 20, 21)	FCS_CKM.2	SSH KAS
ECC KAS (P-256, P-384, P-521)	FCS_CKM.2	IKE KAS
FFC KAS (DH Group 14, 15, 16, 19, 20, 21, 24)	FCS_CKM.2	IKE KAS

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

[MOD_VPNGW] PP-Module for Virtual Private Network (VPN) Gateways, version 1.1 dated 18 June 2020

[MOD_FW] PP-Module for Stateful Traffic Filter Firewalls, Version 1.4 + Errata 20200625 dated 25 June 2020

[MOD_IPS] PP-Module for Intrusion Prevention Systems (IPS), Version 1.0 dated 11 May 2021

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
TD0800	Updated NIT Technical Decision for IPsec IKE/SA Lifetimes Tolerance	FCS_IPSEC_EXT.1.7, FCS_IPSEC_EXT.1.8, CPP_ND_V2.2-SD	2023.11.13	Yes
TD0792	NIT Technical Decision: FIA_PMG_EXT.1 - TSS EA not in line with SFR	FIA_PMG_EXT.1, CPP_ND_V2.2-SD	2023.09.27	Yes
TD0790	NIT Technical Decision: Clarification Required for testing IPv6	FCS_DTLSC_EXT.1.2, FCS_TLSC_EXT1.2	2023.09.27	No
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

ITEM	TITLE	REFERENCE	PUBLICATION DATE	Relevant to ST
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
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
TD0597	VPN GW Ipv6 Protocol Support	FPF_RUL_EXT.1.6, MOD VPNGW SD v1.1	2021.08.26	Yes
TD0595	Administrative corrections to IPS PP-Module	FAU_GEN.1.1/IPS, Table 4	2021.06.14	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
TD0590	Mapping of operational environment objectives	Section 4.3	2021.05.12	Yes
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

ITEM	TITLE	REFERENCE	PUBLICATION DATE	Relevant to ST
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
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
TD0551	NIT Technical Decision for Incomplete Mappings of Oes in FW Module v1.4+Errata	Sections 5.3.2 and 5.3.4	2020.10.15	Yes
TD0549	Consistency of Security Problem Definition update for MOD_VPNGW_v1.0 and MOD_VPNGW_v1.1	Section 6.1.2	2020.10.02	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
TD0545	NIT Technical Decision for Conflicting FW rules cannot be configured (extension of RfI#201837)	FFW_RUL_EXT.1.8	2020.10.15	Yes
TD0538	NIT Technical Decision for Outdated link to allowed-with list	Section 2	2020.07.13	Yes

ITEM	TITLE	REFERENCE	PUBLICATION DATE	Relevant to ST
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

Table 9 Applicable NIAP Technical Decisions

5.2 Security Audit (FAU)

5.2.1 **FAU_GEN.1/NDcPP 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 7.5 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. [18][19][20]) 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 5 in Chapter 8 of the guidance (Ref. [18][19][20]) 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. [18][19][20]) 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. [18][19][20]) 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. [13]). 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.1/ND (MOD_VPNGWv1.1)**

TSS

The evaluator shall verify that the TSS describes how the TSF can be configured to log network traffic associated with applicable rules. Note that this activity may be addressed in conjunction with the TSS Evaluation Activities for FPF_RUL_EXT.1

The evaluator shall verify that the TSS describes how the TOE behaves when one of its interfaces is overwhelmed by network traffic. It is acceptable for the TOE to drop packets that it cannot process, but under no circumstances is the TOE allowed to pass packets that do not satisfy a rule that allows the permit operation or belong to an allowed established session. It may not always be possible for the TOE to audit dropped packets due to implementation limitations. These limitations and circumstances in which the event of dropped packets is not audited shall be described in the TSS.

The evaluator also verifies that the TSS describes the auditable events for Ipsec peer session establishment that are required by the PP-Module.

Refer to the TSS explanation contained for FPF_RUL_EXT.1

The evaluator shall verify that the TSS describes how the TOE behaves when one of its interfaces is overwhelmed by network traffic. It is acceptable for the TOE to drop packets that it cannot process, but under no circumstances is the TOE allowed to pass packets that do not satisfy a rule that allows the permit operation or belong to an allowed established session. It may not always be possible for the TOE to audit dropped packets due to implementation limitations. These limitations and circumstances in which the event of dropped packets is not audited shall be described in the TSS.

Section 7.8 of the ST explains that in case of an interface getting overwhelmed, packets are dropped. This is recorded by the SNMP mibs as well as a log. When an interface gets overwhelmed with CPU utilization 99% then packets are dropped with syslog record as ‘CPU Utilization greater than 99, expect packet loss’

The evaluator also verifies that the TSS describes the auditable events for Ipsec peer session establishment that are required by the PP-Module.

Section 7.5 of [ST] describes the auditable events for the TOE, including those related to Ipsec per session establishment.

Guidance Documentation

The evaluator shall verify that the operational guidance describes how to configure the TSF to result in applicable network traffic logging. Note that this activity may be addressed in conjunction with the guidance Evaluation Activities for FPF_RUL_EXT.1.

Chapter 11 of the Evaluated Configuration Guide (Ref. [18][19][20]) provides guidance on how to configure network traffic with applicable rules. Traffic filter rules can be configured on a device to

enforce validation against protocols attributes and direct traffic accordingly to the configured attributes. These rules are based on zones on which network interfaces are bound. As specified in the examples in this chapter, traffic can be logged when a security policy is applied to a packet.

Tests

Test 1: The evaluator shall attempt to flood the TOE with network packets such that the TOE will be unable to process all the packets. This may require the evaluator to configure the TOE to limit the bandwidth the TOE is capable to handling (e.g., use of a 10 MB interface). The evaluator shall then review the audit logs to verify that the TOE correctly records that it is unable to process all of the received packets and verify that the TOE logging behavior is consistent with the TSS.

Test 2: The evaluator shall use a remote VPN client to establish an IPsec session with the TOE and observe that the event is logged in accordance with the expectations of the PP-Module.

The evaluator set-up the IPS rules to enable logging of the excessive traffic to be sent. From Alice, the evaluator performed a hping3 command to Bob with the “flood” and “random source” option. After a few seconds, the traffic flow was stopped, and the TOE logs examined. The logs showed that the TOE was unable to process all the packets. The logging was reflective of the explanation provided in the TSS for this requirement.

The remainder of the testing associated with this SFR is assessed by the tests for FCS_IPSEC_EXT.1 and FIA_X509_EXT.1/Rev.

5.2.3 **FAU_GEN.1/IPS Audit Data Generation (IPS)**

TSS

The evaluator shall verify that the TSS describes how the TOE can be configured to log IPS data associated with applicable policies.

The evaluator shall verify that the TSS describes what (similar) IPS event types the TOE will combine into a single audit record along with the conditions (e.g., thresholds and time periods) for so doing. The TSS shall also describe to what extent (if any) that may be configurable.

For IPS_SBD_EXT.1, for each field, the evaluator shall verify that the TSS describes how the field is inspected and if logging is not applicable, any other mechanism such as counting that is deployed. (Ref. [6]).

Section 7.9 of ST(Ref.[13]) explains that IDP policies consist of rule bases, and each rule base contains a set of rules that specify rule parameters, including logging requirements. It also refers to the Junos OS guidance documents for more information about configuring IDP event logging.

Section 7.5 of ST(Ref.[13]) list the IPS related events that TOE logs and also describes what IPS events can be combined into a single audit record, using the Junos OS log suppression mechanism. Log suppression is configurable, based on attributes, including source/destination addresses and number of log occurrences after which log suppression begins.

Section 7.9 of ST(Ref.[13]) list all the header fields that can be used by Junos to match traffic in Junos IDP policies and explains how logging can be enabled in the definition each IDP policy rule.

Guidance Documentation

The evaluator shall verify that the operational guidance describes how to configure the TOE to result in applicable IPS data logging.

As per Intrusion Detection and Prevention User Guide that is linked to in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18,19,20]) logging for signature-based detection is enabled by including the following configuration statement in IDP policy:

```
set security idp idp-policy base-policy rulebase-ips rule <rule-name>
then notification log-attacks alert
```

Attacks-screen such as Ping-of-Death and TCP SYN-FIN are automatically logged.

The evaluator shall verify that the operational guidance provides instructions for any configuration that may be done in regard to logging similar events (e.g., setting thresholds, defining time windows, etc.). (Ref. [6]).

As per Intrusion Detection and Prevention User Guide that is referenced in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18][19][20]f) to set a threshold to begin log suppression, the following command is used:

```
set security idp sensor-configuration log suppression start-log <total
events>
```

To set the threshold at which a log entry is generated, the following command is used:

```
set security idp sensor-configuration log suppression max-time-report
<total events>
```

Tests

The evaluator shall test that the interfaces used to configure the IPS polices yield expected IPS data in association with the IPS policies. A number of IPS policy combination and ordering scenarios need to be configured and tested by attempting to pass both allowed and anomalous network traffic matching configured IPS policies in order to trigger all required IPS events. Note the following: This activity should have been addressed with a combination of the Test EAs for the other IPS requirements. As part of testing this activity, the evaluator shall also ensure that the audit data generated to address this SFR can be handled in the manner that FAU_STG_EXT.1 requires for all audit data (Ref. [6]).

The evaluator shall restart the TOE and verify that the TOE records audit logs of the shut-down and start-up of IPS functions. Configure the TOE with an IPS signature policy and verify that the TOE records audit logs of the modification and enabling of IPS policies. Send multiple network

packets through the TOE that trigger the IPS policy and verify that the TOE records audit logs of total counts of similar events. Disable the network interface with the IPS policy and verify that the TOE records audit logs of the interface being disabled.

5.2.4 **FAU_STG.1/NDcPP Protected audit trail storage**

TSS

The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally and how these records are protected against unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records. (Ref. [6]).

Section 7.5 of the ST (Ref[13]) states that the logs are stored in underlying filesystems and accessible only to authenticated Security administrators through the CLI interface or through direct access to the filesystem.

Guidance Documentation

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

There are no configuration required to protect locally stored audit data.

Tests

The evaluator shall perform the following tests:

a) Test 1: The evaluator shall access the audit trail without authentication as Security Administrator (either by authentication as a non- administrative user, if supported, or without authentication at all) and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to access the audit trail can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

b) Test 2: The evaluator shall access the audit trail as an authorized administrator and attempt to delete the audit records. The evaluator shall verify that these attempts succeed. The evaluator shall verify that only the records authorized for deletion are deleted. (Ref. [6]).

The access control mechanisms prevent non-authorised users from accessing any part of the TOE.

The evaluator shall log in to the TOE using a non-administrative user's credentials via the console. From the console, the evaluator shall then attempt to view the log file.

The evaluator connects via SSH to the TOE as an authorised administrator. From the CLI, the evaluator enters the command to clear the logs. This will provide the evaluator with a list of files to be cleared. The evaluator verifies, through accessing the shell on the TOE, that the files previously listed for the evaluator, have been removed from the TOE.

5.2.5 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.

5.2.6 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 7.5 of the ST(Ref.[13]) 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. 7.5 of the ST(Ref.[13]), “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(Ref.[13]), the TOE is not distributed. As per Sect. 7.5 of the ST(Ref.[13]), 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. 7.5 of the ST(Ref.[13]): “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 7.5 of the ST(Ref.[13]) “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 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 6 of the guidance (Ref. [18][19][20]) 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. [18][19][20]) 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 6 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. [18][19][20]) 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. [13]). According to the guidance (Ref. [18][19][20]) 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/NDcPP 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. 7.1.1 of the ST(Ref.[13]) indicates: “Asymmetric keys are also generated in accordance with FIPS PUB 186-4 Appendix B.3 for RSA Schemes and Appendix B.4 for ECC Schemes for SSH communications. The TOE implements Diffie-Hellman group 14, using the modulus and generator specified by Section 3 of RFC3526”. Table 8 ST(Ref.[13])specifies the different key generation schemes and key sizes used in each protocol implemented by the TOE.

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.[18][19][20]) describes how the administrator can configure SSH (in Chapter 5). 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 the demo ACVTS environment for branch platforms. Production ACVTS environment was used to test the remaining platforms and the certificates A3693 and A4386 were issued. 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 the demo ACVTS environment for branch platforms. Production ACVTS environment was used to test the remaining platforms and the certificates A3693 and A4386 were issued. Further details are contained in Section 4.

Key Generation for Finite-Field Cryptography (FCC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic

prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.
The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:
- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:
- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x:
- len(q) bit output of RBG where 1 <=x <= q-1
- len(q) + 64 bit output of RBG, followed by a mod q-1 operation and a +1 operation, where 1<= x<=q-1.

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

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

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm
- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

for each FFC parameter set and key pair. (Ref. [6]).

This assurance activity is carried out using the demo ACVTS environment for branch platforms. Production ACVTS environment was used to test the remaining platforms and the certificates A3610 and A3945 were issued. Further details are contained in Section 4.

Diffie-Hellman Group 14 and 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)

N/A

5.3.2 FCS_CKM.1/IKE Cryptographic Key Generation (for IKE Peer Authentication)

TSS

The evaluator shall check to ensure that the TSS describes how the key-pairs are generated.

In order to show that the TSF implementation complies with FIPS PUB 186-4, the evaluator shall ensure that the TSS contains the following information:

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

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

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

Any TOE-specific extensions, processing that is not included in the Appendices, or alternative implementations allowed by the Appendices that may impact the security requirements the TOE is to enforce shall be described. (Ref. [6]).

Section 7.1.1 of the ST(Ref. [18][19][20]) indicates that the TOE implements all of the “shall” and “should” requirements and none of the “shall not” or “should not” from FIPS PUB 186-4 Appendix B3 and B4.

Guidance Documentation

The evaluator shall check that the operational guidance describes how the key generation functionality is invoked, and describes the inputs and outputs associated with the process for each signature scheme supported. The evaluator shall also check that guidance is provided regarding the format and location of the output of the key generation process. (Ref. [6]).

The evaluator examined the Evaluated Configuration Guide (Ref. [18][19][20]) and found that it provided a section regarding the “Configuring IPsec VPN with RSA Signature as IKE Authentication on the Initiator or Responder” on page 87 of the document (page 89 for Ref. [3]). This section provided additional links to the Juniper website on how the RSA key generation functionality is invoked and the outputs of this process. The Evaluated Configuration Guide (Ref. [2]) also provided a section entitled “Configuring IPsec VPN with ECDSA signature IKE authentication on the Initiator” and “Configuring IPsec VPN with ECDSA signature IKE authentication on the Responder” on page 91-94 of the document (page 93-96 for [3]). These sections provided additional links to the Juniper website on how the ECDSA key generation functionality is invoked and the outputs of this process.

Tests

For FFC Schemes using “safe-prime” groups:

Testing for FFC Schemes using safe-prime groups is done as part of testing in FCS_CKM.2. (Ref. [6]).

N/A

For all other selections:

The evaluator shall perform the corresponding tests for FCS_CKM.1 specified in the NDcPP SD, based on the selections chosen for this SFR. If IKE key generation is implemented by a different algorithm than the NDcPP key generation function, the evaluator shall ensure this testing is performed using the correct implementation. (Ref. [6]).

Testing for this requirement is carried out through testing of the NDcPP requirements.

5.3.3 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. 7.1.1 of the ST(Ref.[13]), Table 15, 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. [18][19][20]) describes how the administrator can configure SSH (in Chapter 5). 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 the demo ACVTS environment for branch platforms. Production ACVTS environment was used to test the remaining platforms and the certificates A3610 and A3945 were issued. Further details are contained in Section 4.

RSA-based key establishment

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

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

Diffie-Hellman Group 14

The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses Diffie-Hellman group 14. (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.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safe- prime groups by using a known good implementation for each protocol selected in FTP_TRP.1/Admin,

FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses. (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.

5.3.4 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[3]). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

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

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

Where the TSS identifies keys that are stored in a non-plaintext form, the evaluator shall check that the TSS identifies the encryption method and the key-encrypting-key used, and that the key-encrypting-key is either itself stored in an encrypted form or that it is destroyed by a method included under FCS_CKM.4 (Ref. [6]).

Sect. 7.1.1 of the ST(Ref. [13]), Table 16, lists all keys/CSPs applicable to the module and their corresponding:

- CSP;
- Description;
- Method of storage;
- Storage location; and
- Zeroization method.

The list corresponds to the protocols (SSH and IKE/IPsec) used by the TOE, as well the password-based authentication method used for user authentication.

[3] 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.

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]).

The TSS does not define the use of such value for key/CSP erasure.

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 command4 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 Evaluated Configuration Guide (Ref. [18][19][20]) 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.

[4] 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).

Tests

None (Ref. [6]).

5.3.5 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]).

Section 7.1.1 of the ST(Ref. [18][19][20]) details that The TOE utilises AES in CBC, GCM and CTR modes with 128-bit, 192-bit and 256-bit keys.

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]).

The guidance documentation (Ref. [18][19][20]) describes how the administrator can configure SSH and IPsec. As part of these configuration guides, the available encryption/decryption modes and associated key sizes are indicated with configuration examples for how to set these values appropriately for data encryption/decryption. Furthermore, Chapter 9 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 the demo ACVTS environment. ACVTS was used to test the remaining platforms and the certificates A3569, A3370, A3693 and A3493 were issued. Further details are contained in Section 4.

5.3.6 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]).

Section 7.1.4 of the ST(Ref.) indicates that both RSA and ECDSA are supported for use with X.509v3 certificates that conform to RFC 4945 and pre-shared Keys for Ipsec support. This is consistent with FCS_COP.1/SigGen.

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]).

The guidance documentation (Ref. [18,19,20]) describes how the administrator can configure SSH and IPsec. As part of these configuration guides, the available cryptographic algorithms and associated key sizes are indicated with configuration examples for how to set these values appropriately for signature services.

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 the demo ACVTS environment. Production ACVTS environment was used to test the remaining platforms and the certificates A3693, A3370, A4386, and A3945 were issued. 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 the demo ACVTS environment. Production ACVTS environment was used to test the remaining platforms and the certificates A4386 and A3693 were issued. Further details are contained in Section 4.

5.3.7 **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 14 of ST(Ref. [13]) documents the association of the hash functions and other TSF cryptographic functions. Each hash function used as a component of another cryptographic primitives listed in Table 14 is accounted for within the same crypto module/library.

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 Evaluated Configuration Guide (Ref. [18,19,20]) 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 the demo ACVTS environment. Production ACVTS environment was used to test the remaining platforms and the certificates A3493, A3367, A3693, A3370, A3569, A3493 and A3368 were issued. Further details are contained in Section 4.

5.3.8 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]).

Sect. 7.1.1 of the ST(Ref. [13]) states the key length, block size and output size for each HMAC function defined in FCS_COP.1/KeyedHash, namely HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384 and HMAC-SHA-512

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 guidance documentation (Ref. [18,19,20]) describes how the administrator can configure SSH and IPsec. As part of these configuration guides, the available cryptographic algorithms and associated key sizes are indicated with configuration examples for how to set these values appropriately for keyed hash functions.

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 the demo ACVTS environment. Production ACVTS environment was used to test the remaining platforms and the certificates A3569, A3370, A3693, A3367 and A3493 were issued. Further details are contained in Section 4.

5.3.9 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 7.1.2 of ST(Ref. [13]), states that OS Junos uses two sources of entropy: RANDOM_NET, a software-based noise source, and RANDOM_PURE, which is hardware-based noise source. Sect. 7.1.2 refers to a document about Entropy for more details.

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 the demo ACVTS environment. Production ACVTS environment was used to test the remaining platforms and the certificates A3370, A3693 and A3493 were issued. Further details are contained in Section 4.

5.3.10 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 7.1.3 of the ST(Ref. [13]) specifies "ssh-rsa", "ecdsa-sha2-nistp256", "ecdsa-sha2-nistp384" or "ecdsa-sha2-nistp521" as the public key algorithms accepted by the TOE.

Section 7.1.3 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]).

Per Section 7.1.3 of the ST(Ref. [13]), packets greater than 256Kbytes in an SSH transport connection are dropped and the connection is terminated by the TOE.

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]).

Section 7.1.3 of the ST(Ref. [13]) provides a description of the implementation of SSH, including optional characteristics and encryption algorithms supported: aes128-cbc and aes256-cbc, aes128-ctr, and aes256-ctr, which correspond with those listed 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)

Section 7.1.3 of the ST(Ref. [13]) specifies "ssh-rsa", "ecdsa-sha2-nistp256", "ecdsa-sha2-nistp384" and "ecdsa-sha2-nistp521" as the public key algorithms accepted by the TOE, which is consistent with statement of security requirements in the ST.

Table 17 also describes how the TOE authenticates a user using public keys.

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]).

Section 7.1.3 of the ST(Ref. [13]) lists the supported data integrity algorithms (hmac-sha1, hmac-sha2-256 and hmac-sha2-512, which corresponds with FCS_SSHS_EXT.1.6)

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]).

Section 7.1.3 of the ST(Ref. [13]) lists the supported key exchange mechanisms (diffie-hellman-group14-sha1, ecdh-sha2-nistp256, ecdh-sha2-nistp384 and ecdh-sha2-nistp521). This list however matches the match the list in 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]).

Table 17 in Section 7.1.3 of the ST(Ref. [13]) states that TOE rekeys every (2^32-1) bytes, but can be configured to a data limit between 51200 and (2^32-1) or time-limit within 1 and 60 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 Evaluated Configuration Guide (Ref. [18,19,20]) 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;
- 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 most of 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.1-1.

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 Evaluated Configuration Guide (Ref. [18,19,20]) 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 macs [ hmac-sha1 hmac-sha2-256 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 Evaluated Configuration Guide (Ref. [18,19,20]) 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 key-exchange [ ecdh-sha2-nistp256 ecdh-sha2-nistp384
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. [18][19][20]) 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)

For each supported public-key algorithm, the evaluator configures the client to successfully establish an SSH connection and asks the TOE to authenticate using that public-key algorithm only.

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 evaluator configures the client to only accept an unsupported SSH server host public key algorithm and attempts to establish a connection with the TOE.

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 shall open multiple SSH connections to the TOE with each connection restricted to one of the supported integrity algorithms. Each connection shall test a different integrity algorithm. The evaluator shall examine packet-capture logs of the negotiated connections to confirm the use of the selected integrity algorithm.

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 shall attempt to establish an SSH connection to the TOE by restricting the connection attempt to the hmac-sha1-96 integrity 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 shall ensure that the TOE is configured to only allow for the key-exchange algorithms specified by the guidance document for SSH. Having done so, the evaluator attempts to connect to the TOE via SSH by restricting the key-exchange algorithm to only Diffie Hellman group 1. The attempt failed.

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 shall ensure that the TOE is configured to only allow for the key-exchange algorithms specified by the guidance document for SSH. Having done so, the evaluator attempts to connect to the TOE via SSH multiple times. For each attempted connection, the evaluator shall restrict the key-exchange algorithm to a different supported exchange algorithm.

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 shall attempt to set the time-based threshold value via a user-account with non-administrator privileges. Having ensured that the values may only be configured by a user with administrator privileges, the evaluator configures the time-based threshold value to a period of 1 hour. After reconnecting to the TOE via an SSH connection, the evaluator waits for a period of 1 hour and 1 minute and confirms that re-keying has occurred by inspecting the TOE's logs.

The evaluator shall attempt to set the time-based threshold value via a user-account with non-administrator privileges. Having ensured that the values may only be configured by a user with administrator privileges, the evaluator configures the sized-based threshold value to a value of 1GiB. After reconnecting to the TOE via an SSH connection, the evaluator transmits 1GiB of traffic. When the transfer is complete, the evaluator verifies that re-keying has occurred by inspecting the TOE's logs.

5.3.11 **FCS_IPSEC_EXT.1 IPsec Protocol**

TSS

FCS_IPSEC_EXT.1.1

The evaluator shall examine the TSS and determine that it describes what takes place when a packet is processed by the TOE, e.g., the algorithm used to process the packet.

The TSS describes how the SPD is implemented and the rules for processing both inbound and outbound packets in terms of the IPsec policy. The TSS describes the rules that are available and the resulting actions available after matching a rule.

The TSS describes how those rules and actions form the SPD in terms of the BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) actions defined in RFC 4301.

As noted in section 4.4.1 of RFC 4301, the processing of entries in the SPD is non-trivial and the evaluator shall determine that the description in the TSS is sufficient to determine which rules will be applied given the rule structure implemented by the TOE. For example, if the TOE allows specification of ranges, conditional rules, etc., the evaluator shall determine that the description of rule processing (for both inbound and outbound packets) is sufficient to determine the action that will be applied, especially in the case where two different rules may apply. This description shall cover both the initial packets (that is, no SA is established on the interface or for that particular packet) as well as packets that are part of an established SA. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) describes how packets are processed. Each packet is compared against entries in the security policy ruleset in sequential order until one is found that matches the specification in the policy, or until the end of the rule set is reached, in which case the implicit default policy is implemented and the packet is discarded.

Traffic that is allowed by the stateful firewall and routed through an IPsec tunnel is PROTECTED. Traffic that is explicitly "denied" by a firewall rule or does not match any rule in the firewall ruleset

is DISCARDED. Traffic that is permitted by a firewall rule, but not routed by a defined IPsec tunnel is BYPASSED traffic.

FCS_IPSEC_EXT.1.3

The evaluator checks the TSS to ensure it states that the VPN can be established to operate in transport mode and/or tunnel mode (as identified in FCS_IPSEC_EXT.1.3) (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) indicates that the TOE supports tunnel mode only, which matches with the statement in FCS_IPSEC_EXT.1.3.

FCS_IPSEC_EXT.1.4

The evaluator shall examine the TSS to verify that the selected algorithms are implemented. In addition, the evaluator ensures that the SHA-based HMAC algorithm conforms to the algorithms specified in FCS_COP.1/KeyedHash Cryptographic Operations (for keyed-hash message authentication) and if the SHA-based HMAC function truncated output is utilized it must also be described. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) lists the cryptographic algorithms selected in FCS_IPSEC_EXT.1.4, namely AES-GCM-128, AES-GCM-192 and AES-GCM-256, and AES-CBC-128, AES-CBC-192 or AES-CBC-256 using HMAC SHA-256 for ESP protection. HMAC SHA-256 is selected in FCS_COP.1/KeyedHash.

FCS_IPSEC_EXT.1.5

The evaluator shall examine the TSS to verify that IKEv1 and/or IKEv2 are implemented. For IKEv1 implementations, the evaluator shall examine the TSS to ensure that, in the description of the Ipsec protocol, it states that aggressive mode is not used for IKEv1 Phase 1 exchanges, and that only main mode is used. It may be that this is a configurable option. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) indicates that the TOE supports both IKEv1 and IKEv2 and that IKEv1 aggressive mode is not supported.

FCS_IPSEC_EXT.1.6

The evaluator shall ensure the TSS identifies the algorithms used for encrypting the IKEv1 and/or IKEv2 payload, and that the algorithms chosen in the selection of the requirement are included in the TSS discussion. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) lists the cryptographic algorithms for payload protection in IKEv1 and IKEv2 selected in the SFR, namely AES-CBC-128, AES-CBC-192, AES-CBC-256, AES-GCM-128 and AES-GCM-256.

FCS_IPSEC_EXT.1.7

The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 1 SA lifetime and/or the IKEv2 SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.7. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) states that the TOE permits configuration of the IKE and Ipsec lifetime exchanges in terms of number of (kilo)bytes (64 to 4294967294 kilo bytes) or length of time (180 to 86400 seconds). This matches the selection in the FCS_IPSEC_EXT.1.7.

FCS_IPSEC_EXT.1.8

The evaluator shall ensure the TSS identifies the lifetime configuration method used for limiting the IKEv1 Phase 2 SA lifetime and/or the IKEv2 Child SA lifetime. The evaluator shall verify that the selection made here corresponds to the selection in FCS_IPSEC_EXT.1.8. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) states that the TOE permits configuration of the IKE and Ipsec lifetime exchanges in terms of number of (kilo)bytes (64 to 4294967294 kilo bytes) or length of time (180 seconds to 8 hours).

FCS_IPSEC_EXT.1.9

The evaluator shall check to ensure that, for each DH group supported, the TSS describes the process for generating "x". The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of "x" meets the stipulations in the requirement. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) indicates that the DH exponents are chosen randomly using HMAC-SHA-256 DRBG and specifies the exponent lengths for the supported DH groups, as follows: 224 bits (for DH Group 14), 256 bits (for DH Groups 19 and 24) and 384 bits (for DH Group 20). This is consistent with the selections in the requirement.

FCS_IPSEC_EXT.1.10

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

If the second selection is chosen, the evaluator shall check to ensure that, for each PRF hash supported, the TSS describes the process for generating each nonce. The evaluator shall verify that the TSS indicates that the random number generated that meets the requirements in this PP is used, and that the length of the nonces meet the stipulations in the requirement. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) indicates that the DH exponents are chosen randomly using HMAC-SHA-256 DRBG and specifies the exponent lengths for the supported DH groups, as follows: 224 bits (for DH Group 14), 256 bits (for DH Groups 19 and 24) and 384 bits (for DH Group 20). This is consistent with the selections in the requirement.

FCS_IPSEC_EXT.1.11

The evaluator shall check to ensure that the DH groups specified in the requirement are listed as being supported in the TSS. If there is more than one DH group supported, the evaluator checks to ensure the TSS describes how a particular DH group is specified/negotiated with a peer. (Ref. [6]).

As per Section 7.1.4 of the ST(Ref. [13]), the TOE supports DH Groups 14, 19, 20 and 24. The way how a particular DH group is negotiated with a peer is also described. When the TOE receives an IKE proposal, it will select the first DH group that matches the acceptable DH groups configured

in the TOE (one or more of DH Groups 14, 19, 20 or 24) and the negotiation will fail if there is no match. Similarly, when the peer initiates the IKE protocol, the TOE will select the first match from the IKE proposal sent by the peer and the negotiation fails is no acceptable match is found.

FCS_IPSEC_EXT.1.12

The evaluator shall check that the TSS describes the potential strengths (in terms of the number of bits in the symmetric key) of the algorithms that are allowed for the IKE and ESP exchanges. The TSS shall also describe the checks that are done when negotiating IKEv1 Phase 2 and/or IKEv2 CHILD_SA suites to ensure that the strength (in terms of the number of bits of key in the symmetric algorithm) of the negotiated algorithm is less than or equal to that of the IKE SA this is protecting the negotiation. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) describes the possible strengths as 128, 192 or 256 bits. It also states that the TOE ensures that the strength of the symmetric algorithm negotiated to protect the IKEv1 Phase 1, IKEv2 IKE_SA connection is greater than or equal to the strength of the symmetric algorithm negotiated to protect the IKEv1 Phase 2, IKEv2 CHILD_SA connection.

FCS_IPSEC_EXT.1.13

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

If pre-shared keys are chosen in the selection, the evaluator shall check to ensure that the TSS describes how pre-shared keys are established and used in authentication of Ipsec connections. The description in the TSS shall also indicate how pre-shared key establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) indicates that both RSA and ECDSA are supported for use with X.509v3 certificates that conform to RFC 4945 and pre-shared Keys for Ipsec support. This is consistent with FCS_COP.1/SigGen.

Section 7.1.4 of the ST(Ref. [13]) describes how pre-shared keys are established and used in IPsec. The TOE accepts pre-shared text keys and converts the text string into an authentication value using the PRF that is configured as the hash algorithm for the IKE exchanges as per RFC 2409 for IKEv1 or RFC 4306 for IKEv2.

FCS_IPSEC_EXT.1.14

The evaluator shall ensure that the TSS describes how the TOE compares the peer's presented identifier to the reference identifier. This description shall include which field(s) of the certificate are used as the presented identifier (DN, Common Name, or SAN). If the TOE simultaneously supports the same identifier type in the CN and SAN, the TSS shall describe how the TOE prioritizes the comparisons (e.g. the result of comparison if CN matches but SAN does not). If the location (e.g. CN or SAN) of non-DN identifier types must explicitly be configured as part of the reference identifier, the TSS shall state this. If the ST author assigned an additional identifier type, the TSS description shall also include a description of that type and the method by which that type is compared to the peer's presented certificate, including what field(s) are compared and which fields take precedence in the comparison. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) describes how the TOE compares the peer's presented identifier against the reference identifier. The identifiers can be expressed as distinguished names, fully qualified domain name (FQDN), user FQDN or IP address. If the certificate does not validate, or the contents do not match the configured identity, then the SA will not be established.

Guidance Documentation

FCS_IPSEC_EXT.1.1

The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases – a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet. (Ref. [6]).

Chapter 5 of the Flow-based and Packet-based Processing User Guide (Ref. [22]) along with the Chapter 8 of the IDP User Guide (Ref. [21]) details how to specify rules for packet processing, including the three cases specified complementing the level of detail within the ST(Ref. [13]).

FCS_IPSEC_EXT.1.3

The evaluator shall confirm that the guidance documentation contains instructions on how to configure the connection in each mode selected. (Ref. [6]).

Chapter 10 of the guidance documents (Ref. [18,19,20]) details how to configure the VPN IPsec in tunnel mode.

FCS_IPSEC_EXT.1.4

The evaluator checks the guidance documentation to ensure it provides instructions on how to configure the TOE to use the algorithms selected. (Ref. [6]).

Chapter 5 of the VPN IPsec guidance (Ref. [23]), details how to configure the algorithms selected for IPsec VPNs.

FCS_IPSEC_EXT.1.5

The evaluator shall check the guidance documentation to ensure it instructs the administrator how to configure the TOE to use IKEv1 and/or IKEv2 (as selected), and how to configure the TOE to perform NAT traversal (if selected). If the IKEv1 Phase 1 mode requires configuration of the TOE prior to its operation, the evaluator shall check the guidance documentation to ensure that instructions for this configuration are contained within that guidance. (Ref. [6]).

Chapter 11, 10, and 9 respectively of the guidance documents (Ref. [18,19,20]) details instructions for how the administrator can configure the TOE for using both IKEv1 and IKEv2.

FCS_IPSEC_EXT.1.6

The evaluator ensures that the guidance documentation describes the configuration of all selected algorithms in the requirement. (Ref. [6]).

Chapter 11, 10, and 9 respectively of the guidance documents (Ref. [18,19,20]) describes what algorithms are used to encrypt IKEv1 and IKEv2 within Table 10.

FCS_IPSEC_EXT.1.7

The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g. configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours.Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (Ref. [6]).

(TD0800 is applied)

Chapter 4 and 17 of the VPN IPsec guidance (Ref. [23]) details instructions on how to configure SA lifetime, configurable up to 24 hours. The SA lifetime in kilobytes is also configurable as per the VPN IPsec guidance (Ref.[23]).

FCS_IPSEC_EXT.1.8

The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g. configure a time value of 7h 45min to ensure the actual rekey is performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (Ref. [6]).

(TD0800 is applied)

Chapter 4 and 17 of the VPN IPsec guidance (Ref. [23]) details instructions on how to configure SA lifetime, configurable up to 24 hours. The SA lifetime in kilobytes is also configurable as per the VPN IPsec guidance (Ref.[23]).

FCS_IPSEC_EXT.1.11

The evaluator ensures that the guidance documentation describes the configuration of all algorithms selected in the requirement. (Ref. [6]).

Chapter 11,10 and 9 of the guidance documents (Ref. [18,19,20]), describes how to configure the selected DH groups.

FCS_IPSEC_EXT.1.13

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

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

The evaluator will ensure that the guidance documentation describes how to configure the TOE to connect to a trusted CA and ensure a valid certificate for that CA is loaded into the TOE and marked “trusted”. (Ref. [6]).

Chapters 3 and 4 of the VPN IPsec guidance (Ref. [23]) describes how the TOE can be configured to use certificates with ECDSA and RSA respectively.

Chapter 11,10 and 9 of the guidance documents (Ref. [18,19,20]), describe how to configure an IPsec VPN with pre-shared keys and also describes how they are established.

In chapter 11,10 and 9 of the guidance documents (Ref. [18,19,20]), the link to ‘Loading CA and Local Certificates Manually.’ is a reference to how to configure the TOE to connected to CAs and verify their validity.

FCS_IPSEC_EXT.1.14

The evaluator shall ensure that the operational guidance describes all supported identifiers, explicitly states whether the TOE supports the SAN extension or not and includes detailed instructions on how to configure the reference identifier(s) used to check the identity of peer(s). If the identifier scheme implemented by the TOE does not guarantee unique identifiers, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use. (Ref. [6]).

Chapter 5 of the VPN IPsec guidance (Ref.[23]), describes the supported identifiers along with the mentioning of the SAN extension.

Tests

FCS_IPSEC_EXT.1.1

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

Test 2: The evaluator shall devise several tests that cover a variety of scenarios for packet processing. As with Test 1, the evaluator ensures both positive and negative test cases are constructed. These scenarios must exercise the range of possibilities for SPD entries and processing modes as outlined in the TSS and guidance documentation. Potential areas to cover include rules with overlapping ranges and conflicting entries, inbound and outbound packets, and packets that establish SAs as well as packets that belong to established SAs. The evaluator shall verify, via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

Test 1: Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Set up firewall rules on the TOE to log and drop traffic, log and allow plaintext traffic, and log and encrypt traffic. For each rule, send traffic through the TOE that matches the rule, as well as traffic that does not match the rule. Verify that the TOE handles each type of matching traffic as specified by the rule, and silently drops traffic that does not match any rules. For devices that support High Availability, configure the TOE and a second node with a HA IPsec connection that uses Pre-Shared Key authentication. Send traffic from an external entity destined for the second node, a similar subnet, and another external entity through the TOE. Modify the configuration on the TOE with a new login message and commit the changes. Verify that no traffic sourced from Alice is detected on the HA Control link between the TOE and second node. Also verify that the login message configured is not detected in any plaintext packets on the HA Control link, and that plaintext log traffic is detected.

Test 2: Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Set up firewall rules on the TOE to log and encrypt specific traffic, with an overlapping rule below to reject traffic. Send both plaintext and encrypted traffic through the TOE and verify that the encrypted traffic is permitted to pass through. Modify the order of the rules so that the overlapping reject rule is listed first and then the specific rule to permit encrypted traffic. Repeat sending the plaintext and encrypted traffic through the TOE and verify that all traffic is blocked. Set up new conflicting firewall rules on the TOE where encrypted traffic is rejected before being permitted, and plaintext traffic is permitted before being rejected. Send both plaintext and encrypted traffic through the TOE and verify that the plaintext traffic is permitted while the encrypted traffic is rejected. Modify the order of the rules so that the plaintext traffic is rejected before being permitted and the encrypted traffic is permitted before being rejected. Send both plaintext and encrypted traffic through the TOE and verify that the plaintext traffic is rejected while the encrypted traffic is permitted.

The script saves all audit records and validatedf the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.2

The assurance activity for this element is performed in conjunction with the activities for FCS_IPSEC_EXT.1.1 (Ref. [6]).

N/A

FCS_IPSEC_EXT.1.3

Test 1: If tunnel mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in tunnel mode and also configures a VPN peer to operate in tunnel mode. The evaluator configures the TOE and the VPN peer to use any of the allowable cryptographic algorithms, authentication methods, etc. to ensure an allowable SA can be negotiated. The evaluator shall then initiate a connection from the TOE to connect to the VPN peer. The evaluator observes (for example, in the audit trail and the captured packets) that a successful connection was established using the tunnel mode. Test 2: Does not apply as transport mode is not selected for the TOE. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

Test 1: Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Bring up the IPsec connection. Verify that both the TOE and external entity report using the Tunnel mode for the connection. For High Availability, configure the TOE and a second node with a HA IPsec connection that uses Pre-Shared Key authentication. Ensure the IPsec connection is active. Verify that ESP traffic is detected on the HA Control link between the TOE and the second node, and that the TOE reports using the Tunnel mode for the connection.

The script saves all audit records and validates the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.4

The evaluator shall configure the TOE as indicated in the guidance documentation configuring the TOE to use each of the supported algorithms, attempt to establish a connection using ESP, and verify that the attempt succeeds. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Ensure the configuration uses IKEv1 with the first IKE algorithm, first ESP algorithm, first HMAC algorithm and first DH group supported. Bring up the connection and verify that it is successful. Repeat this process with each remaining IKE algorithm, ESP algorithm and DH group supported. Repeat the entire process, this time using IKEv2. For High Availability, configure the TOE and second node with a HA IPsec connection that uses Pre-Shared Key authentication. Ensure the configuration uses IKEv2 with the first IKE algorithm, first ESP algorithm, first HMAC algorithm and first DH group supported. Bring up the connection and verify that it is successful. Repeat this process with each remaining IKE algorithm, ESP algorithm and DH group supported.

The script saves all audit records and validates the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.5

Test 1: If IKEv1 is selected, the evaluator shall configure the TOE as indicated in the guidance documentation and attempt to establish a connection using an IKEv1 Phase 1 connection in aggressive mode. This attempt should fail. The evaluator should then show that main mode exchanges are supported. Test 2: Does not apply as NAT traversal is not selected. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Ensure this configuration uses IKEv1 Main Mode. Initiate the IPsec connection and verify that is successfully established. Bring down the IPsec connection and modify the external entity to request IKEv1 Aggressive Mode. Initiate the IPsec connection and verify that it does not successfully establish.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.6

The evaluator shall configure the TOE to use the ciphersuite under test to encrypt the IKEv1 and/or IKEv2 payload and establish a connection with a peer device, which is configured to only accept the payload encrypted using the indicated ciphersuite. The evaluator will confirm the algorithm was that used in the negotiation. (Ref. [6]).

The testing for this is subsumed by the tests for FCS_IPSEC_EXT.1.4.

FCS_IPSEC_EXT.1.7

Test 1: Does not apply as 'number of bytes' is not selected Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the Phase 1 SA lifetime on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that a new Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation. (Ref. [6]). (TD0800 is applied)

The evaluator used an automation script to carry out these tests.

Test 2: Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Ensure the connection uses IKEv1. Configure the TOE to have an IKE SA lifetime of 24 hours and the external entity a lifetime of 96 hours. Initiate the IPsec connection from the TOE, wait for 24 hours, then verify that the TOE deletes the IKE SA after the 24 hours and uses a new SA for any subsequent traffic. Repeat the process, this time using IKEv2.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.8

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

Test 2: If ‘length of time’ is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once a new Phase 2 SA is negotiated when or before 8 hours has lapsed. The evaluator shall verify that the TOE initiates a Phase 2 negotiation. (Ref. [6]).

(TD0800 is applied)

The evaluator used an automation script to carry out these tests.

Test 1: Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Ensure the connection uses IKEv1. Configure the TOE to have an IPsec SA lifetime of 64 kilobytes and the external entity a lifetime of unlimited bytes. Initiate the IPsec connection, send 64 kilobytes of traffic from the external entity to the TOE, then verify that the TOE has deleted the IPsec SA and rekeyed a new one for subsequent traffic. Repeat the process, this time using IKEv2.

Test 2: Configure the TOE and an external entity with an IPsec connection that uses Pre-Shared Key authentication. Ensure the connection uses IKEv1. Configure the TOE to have an IPsec SA lifetime of 8 hours and the external entity a lifetime of 24 hours. Initiate the IPsec connection from the TOE, wait for 8 hours, then verify that the TOE deletes the IPsec SA after the 8 hours and uses a new SA for any subsequent traffic. Repeat the process, this time using IKEv2. For High Availability, configure the TOE and second node with a HA IPsec connection that uses Pre-Shared Key authentication. Ensure the connection uses IKEv2. Configure the TOE to have a HA IPsec SA lifetime of 8 hours and the second node a lifetime of 24 hours. Initiate the HA IPsec connection from the TOE, wait for 8 hours, then verify that the TOE deletes the HA IPsec SA after the 8 hours and uses a new SA for any subsequent traffic.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.11

For each supported DH group, the evaluator shall test to ensure that all supported IKE protocols can be successfully completed using that particular DH group. (Ref. [6]).

The testing for this is subsumed by the tests for FCS_IPSEC_EXT.1.4.

FCS_IPSEC_EXT.1.12

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

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

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

d) Test 4: This test shall be performed for each version of IKE supported. The evaluator shall attempt to establish a SA for ESP (assumes the proper parameters where used to establish the IKE SA) that selects an encryption algorithm that is not identified in FCS_IPSEC_EXT.1.4. Such an attempt should fail. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

Test 1: Not applicable as this test is subsumed elsewhere.

Test 2: Configure the TOE and an external entity with an IKEv2 IPsec connection that uses PKI authentication. Ensure that the algorithm strengths for the IKE algorithm and ESP algorithm are both set to 128 bits. Verify the IKE authentication succeeds. Modify the ESP algorithm to use AES-192-CBC and verify that the configuration is rejected due to a higher ESP algorith strength. Repeat the process for each remaining ESP algorithm that provides greater than 128 bits of security. Configure the TOE to use IKEv1, then repeat the process for all ESP algorithms that provide greater than 128 bits of security. For devices that support High Availability, configure both nodes of the HA link to use an ESP algorithm with higher strength than the IKE algorithm and verify that the configuration is rejected.

Test 3: Configure the TOE and an external entity with an IKEv2 IPsec connection that uses PKI authentication. Verify the IKE authentication succeeds. Configure the external entity to offer an unsupported IKE algorithm and verify the connection fails. Attempt to configure a non-supported IKE algorithm on the TOE and verify that the configuration is rejected. Repeat the process using IKEv1. For High Availability, configure both nodes of the HA link to use an IKE algorithm that is not supported and verify that the configuration is rejected.

Test 4: Configure the TOE and an external entity with an IKEv2 IPsec connection that uses PKI authentication. Verify the IKE authentication succeeds. Configure the external entity to offer an unsupported ESP algorithm and verify the connection fails. Attempt to configure a non-supported ESP algorithm on the TOE and verify that the configuration is rejected. Repeat the process using IKEv1. For High Availability, configure both nodes of the HA link to use an ESP algorithm that is not supported and verify that the configuration is rejected.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.13

For efficiency sake, the testing is combined with the testing for FIA_X509_EXT.1, FIA_X509_EXT.2 (for IPsec connections), and FCS_IPSEC_EXT.1.1. (Ref. [6]).

NOTE: The tests for FCS_IPSEC_EXT.1.4 cover both IKEv1 and IKEv2 using PSK. The tests for FIA_X509_EXT.1.1 cover IKEv2 using RSA and ECDSA.

The remaining tests here cover IKEv1 using RSA and ECDSA.

The evaluator used an automation script to carry out these tests.

Configure the TOE and an external entity with an IKEv1 IPsec connection that uses PKI authentication. Configure both sides to use an RSA algorithm. Verify the IKE authentication succeeds. Reconfigure the TOE and external entity to use an ECDSA algorithm again for IKEv1. Verify the IKE authentication succeeds.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

FCS_IPSEC_EXT.1.14

Test 1: Not applicable as no CN identifiers exist.

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

Test 3: Not applicable as no CN identifiers exist.

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

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

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

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

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

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

b) Append '\0' to a non-CN field of an otherwise authorized DN. (Ref. [6]).

Test 2: Configure the TOE and an external entity with an IPsec connection that uses PKI authentication. Configure the TOE to require a SAN:IP identifier and create a signed certificate

for the external entity that matches the SAN:IP identifier. Verify that the IKE authentication succeeds with this identifier. Repeat the process using SAN:FDQN and SAN: User FDQN identifiers.

Test 4: Configure the TOE and an external entity with an IPsec connection that uses PKI authentication. Configure the TOE to require a SAN:IP identifier and create a signed certificate for the external entity that does not match the SAN:IP identifier, but with a CN value that is a string representation of the identifier. Verify that the IKE authentication does not succeed with this identifier. Repeat the process using SAN:FDQN and SAN: User FDQN identifiers.

Test 5: Configure the TOE and an external entity with an IPsec connection that uses PKI authentication. Configure the TOE to require a DN identifier and create a signed certificate for the external entity that matches the DN identifier. Verify that the IKE authentication succeeds with this identifier.

Test 6: Configure the TOE and an external entity with an IPsec connection that uses PKI authentication. Configure the TOE to require a DN identifier and create a signed certificate for the external entity that does matches the DN identifier, except that the CN field is duplicated. Verify that the IKE authentication does not succeed with this identifier. Repeat the process, this time appending '\\0' to a non-CN field of the DN instead of duplicating the DN field.

5.3.12 **FCS_NTP_EXT.1 NTP Protocol**

TSS

FCS_NTP_EXT.1.1

The evaluator shall examine the TSS to ensure it identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained.

The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp. (Ref. [6]).

Section 7.1.5 of the ST (Ref. [13]) states that “Junos OS supports NTPv3 and NTPv4, authenticating the NTP server that it synchronizes to using a SHA1 or SHA2 message digest algorithms. The TOE ignores NTP timestamps that are broadcasted/multicasted. The TOE multiple NTP servers to be configured. At least one is required for time synchronization to occur, but more than 3 NTP servers can be specified.”

Guidance Documentation

FCS_NTP_EXT.1.1

The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the version of NTP supported, how to configure multiple NTP servers for the TOE’s time source and how to configure the TOE to use the method(s) that are selected in the ST. (Ref. [6]).

Chapter 4 in the guidance documents (Ref. [18,19]) details configuration instructions for the versions of NTP supported, NTP time servers and configuring the TOE to use selected methods.

FCS_NTP_EXT.1.2

For each of the secondary selections made in the ST, the evaluator shall examine the guidance document to ensure it instructs the Security Administrator how to configure the TOE to use the algorithms that support the authenticity of the timestamp and/or how to configure the TOE to use the protocols that ensure the integrity of the timestamp. (Ref. [6]).

Chapter 4 in the guidance documents (Ref. [18,19]), instructs how time synchronisation can be authenticated to ensure the integrity of time services, coming from only known sources. Configuration instructions for this are provided.

FCS_NTP_EXT.1.3

The evaluator shall examine the guidance documentation to ensure it provides the Security Administrator instructions as how to configure the TOE to not accept broadcast and multicast NTP packets that would result in the timestamp being updated. (Ref. [6]).

Chapter 4 in the guidance documents (Ref. [18,19]), provides instructions on how to configure NTP as a single time source, using an authentication method to ensure that the TOE is prevented from synchronising from a broadcast or multicast NTP packet.

Tests

FCS_ NTP _EXT.1.1

The version of NTP selected in element 1.1 and specified in the ST shall be verified by observing establishment of a connection to an external NTP server known to be using the specified version(s) of NTP. This may be combined with tests of other aspects of FCS_NTP_EXT.1 as described below. (Ref. [6]).

The evaluator configures the TOE to connect to an NTP server using NTP version 3. They observe network traffic and status outputs indicating version 3 is used to communicate between the TOE and NTP server. They then configure the TOE to use NTP version 4 and again observe that the TOE and NTP server communicate using NTP version 4.

FCS_ NTP _EXT.1.2

The cryptographic algorithms selected in element 1.2 and specified in the ST will have been specified in an FCS_COP SFR and tested in the accompanying Evaluation Activity for that SFR. Likewise, the cryptographic protocol selected in in element 1.2 and specified in the ST will have been specified in an FCS SFR and tested in the accompanying Evaluation Activity for that SFR. [Conditional] If the message digest algorithm is claimed in element 1.2, the evaluator will change the message digest algorithm used by the NTP server in such a way that the new value does not match the configuration on the TOE and confirms that the TOE does not synchronize to this time source. The evaluator shall use a packet sniffer to capture the network traffic between the TOE and the NTP server. The evaluator uses the captured network traffic, to verify the NTP version, to observe time change of the TOE and uses the TOE's audit log to determine that the TOE accepted the NTP server's timestamp update.

The captured traffic is also used to verify that the appropriate message digest algorithm was used to authenticate the time source and/or the appropriate protocol was used to ensure integrity of the timestamp that was transmitted in the NTP packets. (Ref. [6]).

The evaluators configure the TOE to attempt to use SHA256 for a valid SHA1 NTP authentication key and observe that the authentication attempt fails. They then configure the TOE to attempt to use SHA1 for a valid SHA256 NTP authentication key and observe that the authentication attempt again fails.

FCS_ NTP _EXT.1.3

The evaluator shall configure NTP server(s) to support periodic time updates to broadcast and multicast addresses. The evaluator shall confirm the TOE is configured to not accept broadcast and multicast NTP packets that would result in the timestamp being updated. The evaluator shall check that the time stamp is not updated after receipt of the broadcast and multicast packets. (Ref. [6]).

The evaluators configure an NTP server to provide broadcast and multicast time updates. After a sufficient period of time, they verify that the TOE does not accept the time updates.

FCS_ NTP _EXT.1.4

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

b) Test 2: (The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers). The evaluator shall confirm that the TOE would not synchronize to other, not explicitly configured time sources by sending an otherwise valid but unsolicited NTP Server responses indicating different time from the TOE's current system time. This rogue time source needs to be configured in a way (e.g. degrade or disable valid and configured NTP servers) that could plausibly result in unsolicited updates becoming a preferred time source if they are not discarded by the TOE. The TOE is not mandated to respond in a detectable way or audit the occurrence of such unsolicited updates. The intent of this test is to ensure that the TOE would only accept NTP updates from configured NTP Servers. It is up to to the evaluator to craft and transmit unsolicited updates in a way that would be consistent with the behaviour of a correctly-functioning NTP server.

TD0528 is applied to this test and hence also removes FCS_NTP_EXT.1.5.

a) The evaluators configure the TOE to synchronise with three NTP servers and verify that the TOE successfully updates its time. They also configure the NTP servers to individually and solely provide alternate times and verify that the server can synchronise with each one.

b) The TOE is configured to synchronise time with the NTP server. A script is then run on the NTP server that responds to NTP requests with a valid response except for a rogue IP address. The evaluator verifies that the TOE does not accept these time updates nor update the TOE clock.

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 7.2 of the ST (Ref. [13]) 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.

It is always possible for another administrator to “unlock” the account of administrator whose account has been locked for a period of time following failed authentication attempts. In this way the Security Administrator is not permanently blocked from being able to authenticate as the maximum timeout period is 24 hours.

Even when an account is blocked for remote access to the TOE, an administrator is always able to login locally through the serial console and the administrator can attempt authentication via remote access after the maximum timeout period of 24 hours.

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. [18,19,20]) to ensure that instruction for configuring the number of successive unsuccessful authentication attempts for authentication are provided. The “Limiting the Number of User Login Attempts for SSH Sessions” section in Chapter 5, 5 and 4 of the ECG (Ref. [18,19,20]) respectively, 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); and back-off factor (the factor by which the delay increases after each unsuccessful attempt);

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]).

Chapter 5,5 and 4 of the guidance documents (Ref. [18,19,20]) respectively, explain “The device prevents the locked users to perform activities that require authentication, until a security administrator clears the lock or the defined time period for the device to remain locked has elapsed. However, the existing locks are ignored when the user attempts to log in from the local console”.

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 configures the TOE to lock-out access for a user after 3 failed attempts at authentication via SSH. The evaluator also configures the number of minutes for a lockout period to be 5 minutes. This configuration is then validated by the evaluator by attempting a login via SSH with incorrect password credentials 3 times. Having been locked out, the evaluator waits 5 minutes and tries again with the correct credentials.

The evaluator shall ensure that a user with system administrator privileges is locked out from logging in via SSH. This is carried out by repeating the first two steps of the previous test. The evaluator shall then access the TOE 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 shall be able to still access the system via the serial console. Once authenticated, the evaluator shall verify that the account is locked out and clear the lockout state for the locked-out user. Once

cleared, the evaluator shall verify the action by logging in to the TOE via SSH using the credentials of the previously locked-out user.

5.4.2 **FIA_PMG_EXT.1 Password Management**

TSS

The evaluator shall check that the TSS lists the supported special character(s) for the composition of administrator passwords

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

The evaluator shall check that the TSS lists the range of values supported for the minimum_password_length parameter. The listed range shall include the value of 15.(Ref. [6]).

(TD0792 is applied)

Section 7.2 of the ST (Ref. [13]) 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 3 of the Evaluated Configuration Guide (Ref. [18,19,20]) identifies the characters that may be used in the password as “both alphanumeric and punctuation characters, composed of any combination of upper and lowercase letters, numbers, and special characters such as, “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”.” Furthermore, it states that “there should be at least a change in one case, one or more digits, and one or more punctuation marks”.

The guide 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 2
```

In addition to this, the guide states that hashing algorithm for user passwords shall be configured as SHA256 and provides the following command to set it:

```
set system login password format sha256
```

A note had been added that states “When you change the password algorithm to SHA256, change even the user password. Until then, the old hash algorithm is used.”.

Tests

The evaluator shall perform the following tests.

a) Test 1: The evaluator shall compose passwords that meet the requirements in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, and a minimum length listed in the requirement are supported and justify the subset of those characters chosen for testing. (Ref. [6]).

b) Test 2: The evaluator shall compose passwords that do not meet the requirements in some way. For each password, the evaluator shall verify that the TOE does not support the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that the TOE enforces the allowed characters and the minimum length listed in the requirement and justify the subset of those characters chosen for testing. (Ref. [6]).

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 tested different sets of passwords that were expected to pass and fail the password requirements enforced by the TOE.

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 7.2 of the ST (Ref. [6]) describes the logon process for each logon method allowed (local console and SSH). The Authentication process and library are login process 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 process 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 7.2 of the ST(Ref. [6]) 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(Ref. [6]) provides:

- Guidance on configuring administrative credentials and privileges (Chapter 3); and
- Guidance on configuring SSH and Console Connections (Chapter 5,5 and 4 respectively).

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 ensures that all methods of authentication to the TOE are configured: console, password-based SSH login and public-key based SSH login. The evaluator carries out a successful login using each one of these methods. The evaluator also carries out an unsuccessful attempt at login using each one of these methods. On all instances, the evaluator verifies the success or failure of these methods through verifying entries in syslog.

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 carries out a Nmap scan on the IP address of the TOE for all available protocols.

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]).

Section 7.2 of the ST (Ref. [13]) states that: “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.”

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.

5.4.6 **FIA_PSK_EXT.1 Pre-Shared Key Composition**

TSS

The evaluator shall examine the TSS to ensure that it identifies all protocols that allow both text-based and bit-based pre-shared keys, and states that text-based pre-shared keys of 22 characters are supported. For each protocol identified by the requirement, the evaluator shall confirm that the TSS states the conditioning that takes place to transform the text-based pre- shared key from the key sequence entered by the user (e.g., ASCII representation) to the bit string used by the protocol, and that this conditioning is consistent with the last selection in the FIA_PSK_EXT.1.3 requirement (Ref. [6]).

Section 7.1.4 of the ST (Ref. [13]) explains that “The TOE uses pre-shared keys for IPsec. The TOE accepts ASCII pre-shared or bit-based keys of 1 to 255 characters (and their binary equivalent) that may contain upper and lower case letters, numbers, and special characters (that include: “!”, “@”, “#”, “$”, “%”, “^”, “&”, “*”, “(“, and “)”.” “The TOE accepts pre-shared text keys and converts the text string into an authentication value as per RFC 2409 for IKEv1 or RFC 4306 for IKEv2, using the PRF that is configured as the hash algorithm for the IKE exchanges.”

Guidance Documentation

The evaluator shall examine the operational guidance to determine that it provides guidance to administrators on the composition of strong text-based pre-shared keys, and (if the selection indicates keys of various lengths can be entered) that it provides information on the merits of shorter or longer pre- shared keys. The guidance must specify the allowable characters for pre-shared keys, and that list must be a super-set of the list contained in FIA_PSK_EXT.1.2.

The evaluator shall confirm the operational guidance contains instructions for either entering bit-based pre-shared keys for each protocol identified in the requirement, or generating a bit-based pre-shared key (or both). The evaluator shall also examine the TSS to ensure it describes the process by which the bit- based pre-shared keys are generated (if the TOE supports this functionality), and confirm that this process uses the RBG specified in FCS_RBG_EXT.1 in the Base-PP (Ref. [6]).

Chapter 10, 11 and 9 respectively of the Evaluated Configuration Guides (Ref. [18,19,20]) provides the following requirements of the PSK:

“A device running Junos OS uses certificate-based authentication or preshared keys for IPsec. TOE accepts ASCII preshared or bit-based keys up to 255 characters (and their binary equivalents) that contain uppercase and lowercase letters, numbers, and special characters such as !, @, #, $, %, ^, &, *, (, and). The device accepts the preshared text keys and converts the text string into an authentication value as per RFC 2409 for IKEv1 or RFC 4306 for IKEv2, using the PRF that is configured as the hash algorithm for the IKE exchanges. The Junos OS does not impose minimum complexity requirements for preshared keys. Hence, users are advised to carefully choose long preshared keys of sufficient complexity.”

Tests

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

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

Test 2 [conditional]: If the TOE supports pre-shared keys of multiple lengths, the evaluator shall repeat Test 1 using the minimum length; the maximum length; and an invalid length. The minimum and maximum length tests should be successful, and the invalid length must be rejected by the TOE.

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

Test 4 [conditional]: If the TOE does generate bit-based pre-shared keys, the evaluator shall generate a bit-based pre-shared key of the appropriate length and use it according to the instructions in the operational guidance. The evaluator shall then demonstrate that a successful protocol negotiation can be performed with the key (Ref. [6]).

The evaluator used an automation script to carry out these tests. The script composes a pre-shared key of 22 characters that contains a combination of the allowed characters in accordance with the operational guidance and then establishes and sends and receives traffic over a PSK-based VPN tunnel with the 22-character password.

The script repeats the above procedure with a password that is below the minimum length and then again with a password that is above the minimum length and verifies that both attempts fail.

The script then repeats the same procedure with a bit-based pre-shared key of the appropriate length.

The TOE does not generate bit-based pre-shared keys.

5.4.7 **FIA_X509_EXT.1/Rev X.509 Certificate Validation**

TSS

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

The TSS shall describe when revocation checking is performed and on what certificates. If the revocation checking during authentication is handled differently depending on whether a full certificate chain or only a leaf certificate is being presented, any differences must be summarized in the TSS section and explained in the Guidance. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) describes the process of certificate validation:

“To validate certificates, the TOE extracts the subject, issuer, subjects public key, signature, basicConstraints and validity period fields. If any of those fields is not present, the validation fails. The issuer is looked up in the PKI database. If the issuer is not present, or if the issuer certificate does not have the CA:true flag in the basicConstraints section, the validation fails. The TOE verifies the validity of the signature. If the signature is not valid, the validation fails. It then confirms that the current date and time is within the valid time period specified in the certificate. The TOE also extracts the extendedKeyUsage field and verifies the value represents that for the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3).

If the TOE has been configured to perform a revocation check using CRL (as specified in RFC 5280 Section 6.3). If the CRL fails to download, the certificate is considered to have failed validation, unless the option to skip CRL checking on download failure has been enabled.

The TOE validates a certificate path by building a chain of (at least 3) certificates based upon issuer and subject linkage, validating each according the certificate validation procedure described above. If any certificate in the chain fails validation, the validation fails as a whole. A self-signed certificate is not required to be at the root of the certificate chain.

The TOE determines if a certificate is a CA certificate by requiring the CA:true flag to be present in the basicConstraints section.”

Guidance Documentation

The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate. (Ref. [6]).

Chapter 3 of the IPsec VPN User Guide(Ref.[23]), PKI in Junos OS, describes the details of certificate validation, with any rules of fields along with how certificate revocation checks are performed.

Tests

The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self- testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

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

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

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

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

d) Test 4: If OCSP is selected, the evaluator shall configure the OCSP server or use a man-in-the-middle tool to present a certificate that does not have the OCSP signing purpose and verify that validation of the OCSP response fails. If CRL is selected, the evaluator shall configure the CA to sign a CRL with a certificate that does not have the cRLsign key usage bit set and verify that validation of the CRL fails.

e) Test 5: The evaluator shall modify any byte in the first eight bytes of the certificate and demonstrate that the certificate fails to validate. (The certificate will fail to parse correctly.)

f) Test 6: The evaluator shall modify any byte in the certificate signatureValue field (see RFC5280 Sec. 4.1.1.3), which is normally the last field in the certificate, and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

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

(Ref. [6]).

The evaluator used an automation script to carry out these tests.

a) The script creates valid and broken certificate chains, initiates the IPsec connection with the valid chain, remove the intermediate CA from Alice (The Client) then initiate an IPsec connection with the invalid chain.
b) The script initiates an IPsec connection with a valid certificate that is set to expire in 2 months, then sets the TOE time to three months in the futures. The script then attempts to initiate an IPsec connection and observes that the attempt fails.
c) The script attempts to initiate IPsec connections from Alice with certificates that have been revoked. First revoking Alices certificate by the root CA, then revoking the inter CA certificate.
d) The script attempts to initiate an IPsec connection with a CA certificate without cRLsign key bit set and observes the connection fails.
e) The script attempts to initiate an IPsec connection with a certificate after modifying the first eight bytes of the certificate and observes that the connection fails.
f) The script attempts to initiate an IPsec connection with a certificate after modifying a byte the certificate signature field and observes that the connection fails.
g) The script attempts to initiate an IPsec connection with a certificate after modifying a byte of the certificate public key and observes that the connection fails.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

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

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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

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

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

The evaluator shall repeat these tests for each distinct use of certificates. Thus, for example, use of certificates for TLS connection is distinct from use of certificates for trusted updates so both of these uses would be tested. But there is no need to repeat the tests for each separate TLS channel in FTP_ITC.1 and FTP_TRP.1/Admin (unless the channels use separate implementations of TLS). (Ref. [6]).

The evaluator used an automation script to carry out these tests.

a) The script sets algorithm and key sizes, clears existing TOE and PKI settings, configuring the TOE for x509 certificates, it then verifies that there at least one CA does not have the basicConstraints extension, and ensures that the TOE rejects the certificate at both root and leaf points.
b) The script generates a similar setup to part(a), configuring that at least one CA has a basicConstraints extension and the CA flag is set to false and hence confirming that the TOE rejects this certificate in both (i) and (ii) instances.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

5.4.8 **FIA_X509_EXT.2 X.509 Certificate Authentication**

TSS

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

The evaluator shall examine the TSS to confirm that it describes the behaviour of the TOE when a connection cannot be established during the validity check of a certificate used in establishing a trusted channel. The evaluator shall verify that any distinctions between trusted channels are described. If the requirement that the administrator is able to specify the default action, then the evaluator shall ensure that the guidance documentation contains instructions on how this configuration action is performed. (Ref. [6]).

Section 7.1.4 of the ST(Ref. [13]) describes how the TOE chooses which certificates to use, namely for IKE, the TOE validates the certificate presented by the peer. The section on `Configuring VPNs' in the administrative guidance describes in detail how to configure certificate support.

Guidance Documentation

The evaluator shall also ensure that the guidance documentation describes the configuration required in the operating environment so the TOE can use the certificates. The guidance documentation shall also include any required configuration on the TOE to use the certificates. The guidance document shall also describe the steps for the Security Administrator to follow if the connection cannot be established during the validity check of a certificate used in establishing a trusted channel. (Ref. [6]).

As mentioned above, Chapter 10, 11, 9 respectively in the administrative guidance documents(Ref[18,19,20]), Configuring VPNs, details the configurations required for the operating environment and the TOE. Furthermore, in the VPN IPsec guidance (Ref[]), Chapter 17, there is detailed an Online Certificate Status Protocol (OCSP), which explains actions for a Security Administrator to take if there is a connection failure.

Tests

The evaluator shall perform the following test for each trusted channel:

The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate and observe that the action selected in FIA_X509_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

The script generates a root CA and configures the TOE to install it. It then sets up the TOE with a certificate signed by the root CA and signs a leaf certificate for an external entity. The script then configures an IPsec connection between the external entity and the TOE that uses PKI, ensuring the TOE requires the use of a CRL and set up an external server with the CRL. It then initiates the connection where the external entity provides the leaf certificate. The script verifies that the connection succeeds. Stopping the CRL server from running, the script re-attempts the IPsec connection, verifying that this time it fails. Then, modifying the TOE configuration to disable the CRL check on download failure, the script re-attempts the IPsec connection and verifies that it succeeds. Revocation of the external entity certificate is performed and then the script loads the CRL directly on the TOE. Re-attempt the IPsec connection and verify that it fails. Modify the TOE configuration to disable CRL checking entirely. Re-attempt the IPsec connection and verify that it succeeds.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

5.4.9 **FIA_X509_EXT.3 Extended: X509 Certificate Requests**

TSS

If the ST author selects "device-specific information", the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests. (Ref. [6]).

Section 7.1.4 of the ST(Ref.[13]) describes the device specific fields (email and IP address) used in certificate requests.

Guidance Documentation

The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the Certification Request. (Ref. [6]).

The guidance documents (Ref. [18,19,20]) provides links on “Configuring IPsec VPN with RSA Signature as IKE Authentication on the Initiator or Responder”. This section refers to further links on configuring the PKI. The link “Example: Configuring PKI” in this section provides a complete set of examples on how to:

- Set-up a PKI Basic Configuration
- Configuring a CA Profile
- Generating a Public-Private Key Pair
- Enrolling a Local Certificate
- Loading CA and Local Certificates
- Configuring the IPsec VPN with the Certificates

Tests

The evaluator shall perform the following tests:

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

b) Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate response message and demonstrate that the function succeeds. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

a) The script generates a root CA but does not load it on the TOE. Setting up the TOE with a certificate signed by the root CA, it verifies the CSR that the TOE generates is valid and contains all necessary information.

b) The script verifies the TOE certificate on the TOE from Test 1 and confirms that it is not considered valid. The script then installs the root CA from Test 1 onto the TOE, then re-verifies the TOE certificate and confirms that it is now considered valid.

The script saves all audit records and validated the TOE behaves as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

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 Chapter 2 of the Evaluated Configuration guidance (Ref. [18,19,20]) the system software can be updated via the following commands:

```
request system software add /<image path>/<junos package> no-copy no-validate reboot
```

Further information on this procedure may be obtained in the Installation and Upgrade guide that is linked to (referenced) in the Evaluated Configuration Guides (Ref. [18][19][20]) on page 23.

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]).

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. As expected, this 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]).

Section 7.6 of the ST (Ref. [13]) states that “The Security Administrator has the capability to:

- Manage Functions:
  o 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 (89etconf over SSH) and the trusted path for remote Administrative sessions (SSH)
  o Configuring the packet filtering rules of the
    - o Handling of audit data, including setting limits of log file size”

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 documentation (Ref. [18][19]) provides instructions on how to start and stop each of the management services listed in the ST (Ref. [13]).

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 evaluator 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 these attempts 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 user that has security administrator privileges and attempt to run the commands to execute the FIPS self-tests and carry out a reboot. The evaluator confirmed that these attempts succeeded.

5.5.3 FMT_MOF.1/Functions Management of security functions behaviour

TSS

None (Ref. [6]).

Section 7.6 of the ST (Ref. [13]) states that: “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)
  - Configuring the packet filtering rules of the
    - Handling of audit data, including setting limits of log file size”

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 documentation (Ref. [18][19][20]) “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 operation can be carried out without prior authentication in Junos. Therefore, this aspect of the test aims implicitly passes.

In Junos, modifying security relevant parameters requires the ability to enter the configuration mode or the ability to execute the set command. To satisfy the aims of this test, the evaluator logged in as a user without Security Administrator privileges via the console. Once at the CLI, the evaluator attempted to enter the configuration mode. This was not possible for a user without Security Administrator privileges.

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]).

Via a user with security-administrator privileges, the evaluator changed the SSH settings on the TOE to disallow for use of passwords. The evaluator then attempted to log in to the TOE with a username that does not allow for public-key authentication. This resulted in the failure of the attempted login.

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 operation can be carried out without prior authentication in Junos. Therefore, this aspect of the test aims implicitly passes.

The evaluator logged in to the device as a user without security administrator privileges. From the CLI prompt, the evaluator attempted to enter the configuration mode in order to: deactivate all the files that are currently receiving logging information, change the size of the logfiles before they’re archived and the log-rotation frequency. This attempt failed.

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 connected to the TOE via the serial console using the credentials of a user with security administrator credentials. From the configuration mode, the evaluator successfully set the log-rotation frequency, the size of the logfiles before archival, and made them world readable.

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]).

As per Section 7.6 of the ST (Ref.[13]), no functionality is provided prior to login (with the exception of ICMP response).

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 support handling of X509v3 certificates and implements a trust store, which allows administrators to add certificates to the TOE. The ability to manage the TOE’s trust store is restricted using standard Junos permissions settings.

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 other SFRs.

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. [18,19,20]) 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.

When the Evaluated Configuration Guide (Ref. [18,19,20]) is followed, the TOE implements a single role, that of the authorised administrator in order to access these functions. 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]).

The documentation (Ref. [18,19,20]) 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.

When the Evaluated Configuration Guide (Ref. [18,19,20]) is followed, the TOE implements a single role, that of the authorised administrator in order to access these functions. As such, no configuration is required to restrict access to TOE functions and TSF data.

5.5.5 **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]).

Section 7.6 of the ST (Ref.[13]) states that: “The Security Administrator has the capability to:

- Manage crypto keys (***FMT_MTD.1/CryptoKeys***):
  - 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 that in chapters 5, 5 and 4 respectively in the guidance documents (Ref. [18][19][20]) 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]).

No operation can be carried out without prior authentication in Junos. Therefore, this aspect of the test aim implicitly passed.

The evaluator shall login to the device as user without security administrator privileges. From the CLI prompt, the user shall attempt to enter the configuration mode in order to load a public key for a user on the TOE.

The evaluator attempted to generate an RSA key-pair for the creation of a certificate request. This attempt was expected to fail. Accordingly, it did. When the evaluator attempted to configure the IKE and IPsec settings on the TOE, it was expected that the evaluator was not allowed to enter the configuration mode in order to accomplish this. This was the resulting case.

The evaluator generated valid keypairs of each supported client public-key algorithms for SSH. The TOE is configured with each corresponding public key and SSH connections were successfully authenticated.

5.5.6 **FMT_SMF.1/NDcPP 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 7.6 of the ST (Ref. [13]) specifies the security management functions available via the serial port on the device or remotely over SSH. The same CLI functions can be accessed both

locally or remotely. These functions correspond to those described in the guidance documentation as well as those observed by the evaluators during exercising of the TOE.

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 (Ref. [13]), 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_SMF.1/IPS Specification of Management Functions (IPS)**

TSS

The evaluator shall verify that the TSS describes how the IPS data analysis and reactions can be configured. This may be performed in conjunction with the evaluation of IPS_ABD_EXT.1, IPS_IPB_EXT.1, and IPS_SBD_EXT.1. (Ref. [6]).

Section 5.4.3 of the ST (Ref. [13]) details that the TSF is capable of performing the following management functions with regard to IPS:

- Modify these parameters that define the network traffic to be collected and analyzed:
  - Source IP addresses (host address and network address)
  - Destination IP addresses (host address and network address)
  - Source port (TCP and UDP)
  - Destination port (TCP and UDP)
  - Protocol (Ipv4 and Ipv6)
  - ICMP type and code
- Modify thresholds that trigger IPS reactions

Guidance Documentation

The evaluator shall verify that the operational guidance describes the instructions for each function defined in the SFR, describes how to configure the IPS data analysis and reactions, including how to set any configurable defaults and how to configure each of the applicable analysis pattern matching methods and reaction modes. (Ref. [6]).

The evaluator has examined the operational guidance (Ref. [18,19,20]) to determine that information regarding the management and set-up of the IDP Engine may be found in the " Managing the IPS Signature Database (CLI)" section of the Intrusion Detection and Prevention User Guide (Ref. [21]) that is linked in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18,19,20]).

Tests

The evaluator shall perform the following tests:

a) Test 1: The evaluator shall use the operational guidance to create a signature and enable it on an interface. The evaluator shall then generate traffic that would be successfully triggered by the signature. The evaluator should observe the TOE applying the corresponding reaction in the signature.

b) Test 2: The evaluator shall then disable the signature and attempt to regenerate the same traffic and ensure that the TOE allows the traffic to pass with no reaction.

c) Test 3: The evaluator shall use the operational guidance to import signatures and repeat the test conducted in Test 1.

Other testing for this SFR is performed in conjunction with the EAs for IPS_ABD_EXT.1 and IPS_SBD_EXT.1. (Ref. [6]).

The evaluator used an automation script to carry out these tests.

a) The script configured the TOE with a new custom IPS signature. It then sent network traffic through the TOE that matches the IPS signature and verifies that the TOE detects and blocks the traffic as configured.

b) The script then disabled the previously configured IPS signature on the TOE. Sending the same network traffic through the TOE, it verified that the TOE now allows the traffic to pass through.

c) The script imported a custom IPS signature onto the TOE. It then sent network traffic through the TOE that matches the IPS signature and verified that the TOE detects and blocks the traffic as configured.

The script saved all audit records and validated the TOE behaved as expected. The evaluators reviewed the saved results and audit data and verified that the script demonstrated correct operation.

5.5.8 FMT_SMF.1/VPN Specification of Management Functions (VPN)

TSS

The evaluator shall examine the TSS to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the TSS identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface. This activity is addressed with the TSS assurance activities for FPF_RUL_EXT.1. (Ref. [6]).

Section 7.6 of the ST (Ref. [13]) states the TOE provides the ability to manage:

- Configuring the packet filtering rules of the TOE
- Ability to enable, disable, determine and modify behavior, and configure all other VPN-associated security functions of the TOE identified in [MOD_VPNGW]
- Manage the packet filtering rules and VPN configuration.

Guidance Documentation

The evaluator shall examine the operational guidance to confirm that all management functions specified in FMT_SMF.1/VPN are provided by the TOE. As with FMT_SMF.1 in the Base-PP, the evaluator shall ensure that the operational guidance identifies what logical interfaces are used to perform these functions and that this includes a description of the local administrative interface. (Ref. [6]).

The evaluator found that the IPsec VPN user guidance (Ref. [23]) in combination with chapters 10,11 and 9 respectively of the operational guidance (Ref. [18,19,20]) details the management functions specified and are provided by the TOE. Further, these guidance documents also describe what interfaces are used to perform these actions.

Tests

The evaluator tests management functions as part of performing other test EAs. No separate testing for FMT_SMF.1/VPN is required unless one of the management functions in FMT_SMF.1.1/VPN has not already been exercised under any other SFR. (Ref. [6]).

Testing of this SFR is assessed in the test cases associated with FPF_RUL_EXT.1.

5.5.9 FMT_SMF.1/FFW Specification of Management Functions (FFW)

TSS

The evaluation activities specified for FMT_SMF.1 in the Supporting Document for the Base-PP shall be applied in the same way to the newly added management functions defined in FMT_SMF.1/FFW in the FW Module. (Ref. [10]).

Section 7.6 of the ST (Ref. [13]) states the TOE provides the ability to:

Configuring the packet filtering rules of the TOE

- Perform management functions

Guidance Documentation

The evaluation activities specified for FMT_SMF.1 in the Supporting Document for the Base-PP shall be applied in the same way to the newly added management functions defined in FMT_SMF.1/FFW in the FW Module. (Ref. [10]).

Table 4 of the guidance documentation (Ref. [18,19,20]) includes all auditable events for FMT_SMF.1/FFW.

Tests

The evaluation activities specified for FMT_SMF.1 in the Supporting Document for the Base-PP shall be applied in the same way to the newly added management functions defined in FMT_SMF.1/FFW in the FW Module. (Ref. [10]).

The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. (Ref. [6])

The evaluator performed management functions on the TOE, including creating a firewall rule, modifying a firewall rule and deleting a firewall rule.

5.5.10 **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]).

Section 7.6 of the ST (Ref. [13]) states that: “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 documents (Ref. [18,19,20]) 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 7.1.1 of the ST (Ref. [13]), 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 permission39. Encrypted or obfuscated passwords can be viewed by Security Administrators using the CLI command ‘request system decrypt password’.

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])

Section 7.2 of the ST (Ref. [13]) describes that;

“Locally stored authentication credentials are protected (FPT_APW_EXT.1):

- The passwords are stored in obfuscated form using HMAC-sha1.
- Authentication data for public key-based authentication methods are stored in a directory owned by the user (and typically with the same name as the user). This directory contains the files ‘.ssh/authorized_keys’ and ‘.ssh/authorized_keys2’ which are used for SSH public key authentication.”

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 7.3 of the ST (Ref. [13]) 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 16, 17 and 15 respectively of the guidance documents (Ref. [18,19,20]) describes in detail the self-tests that are executed and provides guidance to the Administrator on what actions to take, namely to “contact the Juniper Networks Technical Assistance Center (JTAC)”.

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 accessed the TOE as a security administrator via the console; verified that the system is in FIPS mode and rebooted the TOE. On reboot of the TOE, the evaluator verified that the FIPS self-tests are performed.

5.6.4 **FPT_TST_EXT.3 Self-Test with Defined Methods**

TSS

The evaluator verifies that the TSS describes the method used to perform self-testing on the TSF executable code, and that this method is consistent with what is described in the SFR. (Ref. [6]).

Section 7.3 of the ST (Ref. [13]) explains how Junos OS uses digital signatures to verify the integrity of Junos OS firmware binaries.

Guidance Documentation

There are no guidance Evaluation Activities for this SFR.

Tests

There are no test Evaluation Activities for this SFR.

5.6.5 **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. 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]).

Section 7.4 of the ST (Ref. [13]) indicates that security administrators are able to query the current version of the TOE firmware using the CLI command “show version”.

Also in Section 7.4 of the ST, 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. If verification fails, the TOE uses the last known verified image.

If the options ‘support automatic checking for updates’ or ‘support automatic updates’ are chosen from the selection in FPT_TUD_EXT.1.2, the evaluator shall verify that the TSS explains what actions are involved in automatic checking or automatic updating by the TOE, respectively. (Ref. [6]).

Section 7.4 of the ST (Ref. [13]) indicates that there is no automatic updating of the Junos OS.

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]).

As per the guidance documentation (Ref. [18,19,20]) 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 the “Junos Overview” section of the Installation and Upgrade Guide that is referenced on Page 22 on each of the guidance documents (Ref. [18,19,20]):

“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]).

This process involved applying the same version of the TOE as an update to the currently installed version of the TOE. The evaluator copied the upgrade image to a USB flash drive. The flash drive was mounted on the TOE in a temporary location. The evaluator checked the currently installed

version of the TOE to verify that the version under evaluation is installed. The evaluator carried out an update of the TOE from the USB flash drive. The evaluator observed the output on the screen to verify that the upgrade image was applied. Having completed the process, the evaluator checked the version of the TOE to ensure that it was the version under evaluation.

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.

5.6.6 **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 7.5 of the ST (Ref. [13]) 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]).

Chapter 4 of the guidance documents (Ref. [18][19][20]), Network Time Protocol, details how configuration and communication occurs for the NTP of the TOE.

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]).

a) The evaluator manually set the time on the TOE using the operational guidance. The evaluator waited for a period of 3 minutes and confirmed that the time was in agreement with respect to the value that was originally set.
b) This test is covered by FCS_NTP_EXT.1.1.

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

TSS

The evaluator shall ensure the TSS describes how the TOE ensures a shutdown upon a self-test failure, a failed integrity check of the TSF executable image, or a failed health test of the noise source. If there are instances when a shut-down does not occur, (e.g., a failure is deemed non-security relevant), the evaluator shall ensure that those cases are identified and a rationale is provided that supports the classification and justifies why the TOE's ability to enforce its security policies is not affected in any such instance.

Section 7.3 of the ST (Ref. [13]) 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 device must be power cycled to attempt to return to operation.”

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

Guidance Documentation

The evaluator shall verify that the operational guidance provides information on the self-test failures that can cause the TOE to shut down and how to diagnose the specific failure that has occurred, including possible remediation steps if available.

Chapter 16, 17 and 15 respectively of the guidance documents (Ref. [18,19,20]) describes in detail the self-tests that are executed and provides guidance to the Administrator on what actions to take, namely to “contact the Juniper Networks Technical Assistance Center (JTAC)”.

Tests

There are no test Evaluation Activities for this SFR.

N/A.

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]).

Section 7.2 of the ST (Ref. [13]) states that 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]).

As per the CLI guide (Ref. [24]), the timeout period for local (serial) connections can be set via the set cli idle-timeout <minutes> command.

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]).

Via the console, the evaluator set the idle timeout for a user with admin privileges to a 1-minute time-out period. With the credentials of this user, the evaluator accessed the TOE via the serial console and not perform any operations for the specified period. The evaluator ensured that the connection is closed at the end of the specified period. The evaluator re-ran this test for time-out periods of 5, 10 and 30 minutes.

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]).

Section 7.2 of the ST (Ref. [13]) states that: “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]).

As per the CLI guide (Ref. [24]), the timeout period for local (serial) connections can be set via the set cli idle-timeout <minutes> command. This method terminates the session with the user having to log back in.

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]).

Via the console, the evaluator set the idle timeout for a user with admin privileges to a 1-minute time-out period. The evaluator accessed the TOE via SSH and did not perform any operations for the specified period. The evaluator ensured that the connection is closed at the end of the specified period. The evaluator re-ran this test for time-out periods of 5, 10 and 30 minutes.

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]).

Section 7.2 of the ST (Ref. [13]) explains that user sessions (local and remote) can be terminated by users (FTA_SSL.4). The administrative user can logout of the existing session by typing exit to exit the CLI admin session and the Junos OS makes the current contents unreadable after the admin initiates the termination.

Guidance Documentation

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

As per the CLI guide (Ref. [24]), any active CLI session can be closed from the CLI via the exit command.

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 logged in to the TOE via the console and entered the command to terminate the connection. The evaluator ensured that the connection was terminated and no further access to the TOE was permitted.

The evaluator accessed the TOE via an SSH connection and terminated the connection using the ‘exit’ command. The evaluator ensured that the connection was closed and verified that no further access to the TOE was permitted.

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]).

Section 7.6 of the ST (Ref. [13]) describes that the TOE provides user access either through the system console or remotely over the Trusted Path using the SSHv2 protocol.

Furthermore, section 7.2 mentioned that Junos enables Security Administrators to configure an access banner provided with the authentication prompt. The banner can provide warnings against unauthorized access to the TOE 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 Guidance Documentation (Ref. [18,19,20]) 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 connected to the TOE using SSH and set the login message. The changes were committed, and the evaluator terminated the SSH session. The evaluator connected to the TOE again using SSH and checks to ensure that the newly set-up banner message was displayed. The evaluator terminated the SSH session and connected to the TOE using the serial console and again verifies that the newly set-up banner message is displayed.

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 7.1.3 of the ST (Ref. [13]), the TOE provides an SSH server to support Trusted Channels using SSHv2 protocol to protect communications 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.

The TOE also implements IPsec as a trusted channel for VPN communications and can act as both initiator and responder.

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. External logging may also be carried out using IKE/IPsec.

The Evaluated Configuration Guide (Ref. [18,19,20]) provides instructions for configuring SSH and the transfer of logs using NETCONF via SSH and/or IPsec/IKE. 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]).

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]).

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 logged in to the TOE via SSH as a user with security administrator privileges. The evaluator generated traffic by viewing the log file that's stored locally on the TOE. The traffic that was generated between the TOE and the remote SSH client is examined by the evaluator to ensure that no data was sent in plaintext. The evaluator then physically disconnected the connection between the TOE and the remote server for a period less than the product of the ClientAliveInterval and ClientAliveCountMax values configured at the TOE. The traffic between the TOE and the client was examined to ensure that no data packets were transmitted during the period of disconnection. The evaluator reconnected the physical link between the TOE and the client and ensured that connectivity is re-established. Having ensured that the connectivity is re-established, the evaluator disconnected the physical link again, this time waiting for a period that is greater than the product of the ClientAliveInterval and ClientAliveCountMax values configured at the TOE. The physical link was reconnected, and the developer ensured that the connection had been terminated and no further communication was possible.

During the course of the evaluation, the evaluator established an IPsec connection with the TOE using a client running Strongswan. The evaluator verified that the TOE was able to function in both in initiator and responder modes. The evaluator was able to verify through Wireshark that packets were not sent in plaintext. The evaluator physically disconnected the link between the TOE and the client. The traffic between the TOE and the client was examined to ensure that no data packets were transmitted during the period of disconnection. The evaluator reconnected the physical link between the TOE and the client and ensured that connectivity is re-established and packets were not being sent in plaintext.

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 7.1.3 of the ST (Ref. [13]) explains that the TOE supports Trusted Paths using SSHv2 protocol which ensures the confidentiality and integrity of remote administration sessions. If desired, an additional layer of protection can be afforded to the trusted path by using IPsec to encapsulate the SSH connection. This is consistent with the statement of security requirements 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. [18,19,20]) and the CLI Guide (Ref. [24]).

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]).

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

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

5.9 Packet Filtering (FPF)

5.9.1 **Rules for Packet Filtering (FPF_RUL_EXT.1)**

FPF_RUL_EXT.1.1

TSS

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

The evaluator shall verify that the TSS also includes a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describes the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets. (Ref. [6]).

Section 7.8 of the ST (Ref. [13]) describes the boot sequence of the TOE which ensures that the packet filtering functionality cannot be bypassed during initialisation. Network interfaces are

brought up only after successful loading of kernel and Information Flow subsystems, and these interfaces cannot send or receive packets unless previously configured by an Administrator.

Section 7.8 of the ST (Ref. [13]) explains that the Information Flow subsystem is responsible for processing network packet and describes the different subsystem modules that packets go through, depending on administrator configured policy. According to the TSS, “If a failure occurs in the “flow” daemon (flowd) causing it to halt, no packet processing will occur and no packets will be forwarded. A failure in another daemon will not prevent the flow daemon from enforcing the policy rule set.”

Guidance Documentation

The operational guidance associated with this requirement is assessed in the subsequent test Evaluation Activities. (Ref. [6]).

N/A

Tests

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

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

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test Evaluation Activities. (Ref. [6]).

This test is subsumed by FFW_RUL_EXT.1.1 and therefore is not duplicated here.

FPF_RUL_EXT.1.2

There are no Evaluation Activities specified for this element. Definition of Packet Filtering policy, association of operations with Packet Filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4. (Ref. 8)

FPF_RUL_EXT.1.3

There are no Evaluation Activities specified for this element. Definition of Packet Filtering policy, association of operations with Packet Filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4. (Ref. 8)

FPF_RUL_EXT.1.4

TSS

The evaluator shall verify that the TSS describes a Packet Filtering policy that can use the following fields for each identified protocol, and that the RFCs identified for each protocol are supported:

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

The evaluator shall verify that the TSS describes how conformance with the identified RFCs has been determined by the TOE developer (e.g., third party interoperability testing, protocol compliance testing).

The evaluator shall verify that each rule can identify the following actions: permit, discard, and log.

The evaluator shall verify that the TSS identifies all interface types subject to the Packet Filtering policy and explains how rules are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used), they can be treated collectively as a distinct network interface. (Ref. [6]).

Section 7.8, Table 18 of the ST (Ref. [13]) explains that the TOE performs stateful network traffic filtering on network packets using the following network traffic protocols and network fields conforming to the described RFCs:

- Internet Protocol (Ipv4)
  - Source address, Destination Address, Transport Layer Protocol
- Internet Protocol version 6 (Ipv6)
  - Source address, Destination Address, Transport Layer Protocol
- Transmission Control Protocol (TCP)
  - Source port, Destination port
- User Datagram Protocol (UDP)

- Source port, Destination port

Section 7.8 of the ST (Ref. [13]) confirms that the required RFCs are supported by the TOE.

According to Section 7.8 of the ST (Ref. [13]), conformance is demonstrated by protocol compliance testing by the product QA team.

Section 7.8 of the ST (Ref. [13]) indicates that the TOE supports permit, deny, and log operations to be associated with rules and these rules can be assigned to distinct network interfaces.

Section 7.9 of the ST (Ref. [13]) indicates that the TOE can inspect all traffic passing through the TOE's Ethernet interfaces (inline mode). Ethernet interfaces can be assigned to Zones on which firewall and IDP policies are predicated.

Guidance Documentation

The evaluators shall verify that the operational guidance identifies the following protocols as being supported and the following attributes as being configurable within Packet filtering rules for the associated protocols:

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

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

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

The guidance may describe the other protocols contained within the ST (e.g., IPsec, IKE, potentially HTTPS, SSH, and TLS) that are processed by the TOE. The evaluator shall ensure that it is made clear what protocols were not considered as part of the TOE evaluation. (Ref. [6]).

Chapter 12, 13 and 11 respectively of the Evaluated Configuration Guides (Ref. [18,19,20]) identifies the protocols and the attributes listed above as being configurable within stateful traffic filtering. It provides a table (Table 13) that displays the protocols and the fields that is able to be configured.

According to 12, 13 and 11 respectively of the Evaluated Configuration Guides (Ref. [18,19,20]) the TOE supports the following protocols and attributes:

- ICMPv4 - RFC 792, Internet Control Message Protocol version 4
  - Type
  - Code
- ICMPv6 - RFC 4443, Internet Control Message Protocol version 6
  - Source address
  - Destination address
  - Transport Layer Protocol
- IPv4 - RFC 791, Internet Protocol
  - Source address
  - Destination address
  - Transport Layer Protocol
- IPv6 - RFC 2460, Internet Protocol
  - Source port
  - Destination port
- TCP - RFC 793, Transmission Control Protocol
  - Source port
  - Destination port
- UDP - RFC 768, User Datagram Protocol
  - Source port
  - Destination port

Chapter 11, 12 and 10 respectively of the Evaluated Configuration Guides (Ref. [18,19,20]) defines the following modes to define how a device directs traffic:

- Bypass: the Permit option directs the traffic traversing the device through the stateful firewall inspection, but not through the IPsec VPN tunnel.
- Discard: the Deny option inspects and drops all packets that do not match any Permit policies.
- Protect: The traffic is routed through an IPsec tunnel based on the combination of route lookup and Permit policy inspection.
- Log: This option logs traffic and session information for all the modes mentioned above.

According to the Evaluated Configuration Guide (Ref. [18,19,20]):

- Interfaces are assigned security zones
- A screen may be assigned to a security zone
- Security zones are assigned to policies

Also, the guidance makes clear that the following “protocols are also supported on devices running Junos OS and are a part of this evaluation”:

- IPsec
- IKE
- SSH

Furthermore, “the following protocols are supported on devices running Junos OS but are not included in the scope of this evaluation”:

- OSPF
- BGP
- RIP

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet filter rules can be created that permit, discard, and log packets for each of the following attributes:

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

Test 2: The evaluator shall repeat Test 1 above for each distinct network interface type supported by the TOE to ensure that Packet filtering rules can be defined for each all supported types.

Note that these test activities should be performed in conjunction with those of FPF_RUL_EXT.1.6 where the effectiveness of the rules is tested; here the evaluator is just ensuring the guidance is sufficient and the TOE supports the administrator creating a ruleset based on the above attributes. The test activities for FPF_RUL_EXT.1.6 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfill the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator. (Ref. [6]).

The requirements for this element are fully tested in FPF_RUL_EXT.1.6

FPF_RUL_EXT.1.5

TSS

The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset. (Ref. [6]).

Section 7.8 of the ST (Ref. [13]) explains that the Information Flow subsystem is responsible for processing network packet and describes the different subsystem modules that packets go through. By default, the TOE behaviour is to deny packets when there is no rule match unless another required condition allows the network traffic.

The Information Flow subsystem consists of the following modules:

- IP Classification Module – classifies packets into several categories, saves classification information in packet processing context, and provides other modules with that information for assisting further processing;
- Attack Detection Module – monitors arriving traffic, performs predefined attack detection services (prevents attacks), and issues actions when an attack is found;
- Session Lookup Module – performs lookups in the session table which is used for all interfaces based on the information in incoming packets. The lookup is based on the exact match of source IP address and port, destination IP address and port, protocol attributes (e.g., SYN, ACK, RST, and FIN), and egress/ingress zone;
- Security Policy Module – applies the administrator defined policy rule set. The Security Policy module will deny packets when there is no policy match unless another policy allows the traffic;
- Session Setup Module – sets up a session for packet that do not match current sessions;
- INETD Module – provides internet services for the TOE; and
- RDP Module – provides the implementations and algorithms for the routing protocols and route calculations.

Guidance Documentation

The evaluator shall verify that the operational guidance describes how the order of Packet filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing. (Ref. [6]).

The link to the “Reordering Security Policies” reference in the Evaluated Configuration Guide (Ref. [18][19][20]) states that: *The concept of policy shadowing refers to the situation where a policy higher in the policy list always takes effect before a subsequent policy. Because the policy lookup always uses the first policy it finds that matches the five-part tuple of the source and destination zone, source and destination address, and application type, if another policy applies to the same tuple (or a subset of the tuple), the policy lookup uses the first policy in the list and never reaches the second one.*

This reference also provides examples on how the ordering of the policies may be changed using the *insert* statement:

Reorder two existing policies by entering the following command:

```
insert security policies from-zone trust to-zone untrust policy permit-
mail before policy permit-all
```

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet filter rules can be created that permit, discard, and log packets for each of the following attributes:

Test 2: The evaluator shall repeat Test 1 above for each distinct network interface type supported by the TOE to ensure that Packet filtering rules can be defined for each all supported types.

Note that these test activities should be performed in conjunction with those of FPF_RUL_EXT.1.6 where the effectiveness of the rules is tested; here the evaluator is just ensuring the guidance is sufficient and the TOE supports the administrator creating a ruleset based on the above attributes. The test activities for FPF_RUL_EXT.1.6 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfill the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator.

Test 1: Configure the TOE with equal stateless firewall rules that either permit or discard traffic with a specific address for both IPv4 and IPv6. Set the permit rules as the first rule. Direct matching network traffic through the TOE. Invert the order of the rules so that the discard rules are ordered before the permit rules. Again, direct matching network traffic through the TOE. Verify that the TOE permits and logs the traffic in the first instance and discards and logs the traffic in the second instance. The expected outcome was observed in both instances.

Test 2: Configure the TOE with stateless IPv4 rules that permit traffic for a specific address and discard traffic for a subnet that contains the specific address. Configure stateless IPv6 rules that permit traffic for a subnet that contains a specific address and discard traffic for the specific address. Set the permit rules as the first rule. Direct matching network traffic through the TOE. Invert the order of the rules so that the discard rules are ordered before the permit rules. Again, direct matching network traffic through the TOE. Verify that the TOE permits and logs the traffic in the first instance and discards and logs the traffic in the second instance. The expected outcome was observed in both instances.

FPF_RUL_EXT.1.6

TSS

The evaluator shall verify that the TSS describes the process for applying Packet Filtering rules and also that the behavior (either by default, or as configured by the administrator) is to discard packets when there is no rule match. (Ref. [6]).

Section 7.8 of the ST (Ref. [13]) describes the information flow subsystem process for applying packet filtering rules. The Security Policy Module applies the firewall policy rule set configured by the administrator. The default policy is to deny packets when there is no policy match unless another policy allows the traffic.

Guidance Documentation

The evaluator shall verify that the operational guidance describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the operational guidance provides the appropriate instructions to configure the behavior to discard packets with no matching rules. (Ref. [6]).

As per Chapter 12, 13 and 11 respectively of the Evaluated Configuration Guides (Ref. [18,19,20]), the default reject-all rule can be implemented via the following command:

```
set security policies default-policy deny-all
```

This rule will be applied to all traffic that does not meet any other configured rule.

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall configure the TOE to permit and log each defined IPv4 Transport Layer Protocol (see table below) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

Test 2: The evaluator shall configure the TOE to permit all traffic except to discard and log each defined IPv4 Transport Layer Protocol (see table below) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 3: The evaluator shall configure the TOE to permit and log each defined IPv4 Transport Layer Protocol (see table below) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each defined IPv4 Transport Layer Protocol (See table below) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 4: The evaluator shall configure the TOE to permit and log each defined IPv6 Transport Layer Protocol (see table below) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source

address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

Test 5: The evaluator shall configure the TOE to permit all traffic except to discard and log each defined IPv6 Transport Layer Protocol (see table below) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 6: The evaluator shall configure the TOE to permit and log each defined IPv6 Transport Layer Protocol (see table below) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each defined IPv6 Transport Layer Protocol (see table below) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that they are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 7: The evaluator shall configure the TOE to permit and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.

Test 8: The evaluator shall configure the TOE to discard and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.

Test 9: The evaluator shall configure the TOE to permit and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Here the evaluator ensures that the UDP port 500 (IKE) is included in the set of tests.

Test 10: The evaluator shall configure the TOE to discard and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Again, the evaluator ensures that UDP port 500

is included in the set of tests. The following table identifies the RFC defined values for the protocol fields for IPv4 and IPv6 to be used in configuring and otherwise testing Packet Filtering rule definition and enforcement:

Table 3: RFC Values for IPv4 and IPv6, page 15 (Ref. [6]).

Test 1: Configure the TOE to permit and log each of the transport layer protocols specified in this SFR based on specific source and destination addresses. Generate IPv4 traffic that matches each of the transport layer protocols specified and source/destination addresses and direct it through the TOE. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE based on combinations of specific source and wildcard destination addresses, wildcard source and specific destination addresses, and wildcard source and destination addresses.

Test 2: Configure the TOE to discard and log each of the transport layer protocols specified in this SFR based on specific source and destination addresses and permit all other traffic. Generate IPv4 traffic that matches each of the transport layer protocols specified and source/destination addresses and direct it through the TOE. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE based on combinations of specific source and wildcard destination addresses, wildcard source and specific destination addresses, and wildcard source and destination addresses.

Test 3: Configure the TOE to permit and log each of the transport layer protocols specified in this SFR based on specific source and destination addresses. Further configure the TOE to discard and log each of the transport layer protocols specified in this SFR based on different (non-overlapping) specific source and destination addresses. Generate IPv4 traffic that matches each of the transport layer protocols specified but does not match any of the source/destination addresses and direct it through the TOE. Verify via audit logs and packet captures that the TOE discards all traffic. Repeat the test, configuring the TOE based on combinations of specific source and wildcard destination addresses, wildcard source and specific destination addresses, and wildcard source and destination addresses.

Test 4: Configure the TOE to permit and log each of the transport layer protocols specified in this SFR based on specific source and destination addresses. Generate IPv6 traffic that matches each of the transport layer protocols specified and source/destination addresses and direct it through the TOE. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE based on combinations of specific source and wildcard destination addresses, wildcard source and specific destination addresses, and wildcard source and destination addresses.

Test 5: Configure the TOE to discard and log each of the transport layer protocols specified in this SFR based on specific source and destination addresses and permit all other traffic. Generate IPv6 traffic that matches each of the transport layer protocols specified and source/destination addresses and direct it through the TOE. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE based on combinations of specific source and wildcard destination addresses, wildcard source and specific destination addresses, and wildcard source and destination addresses.

Test 6: Configure the TOE to permit and log each of the transport layer protocols specified in this SFR based on specific source and destination addresses. Further configure the TOE to discard and log each of the transport layer protocols specified in this SFR based on different (non-overlapping) specific source and destination addresses. Generate IPv6 traffic that matches each

of the transport layer protocols specified but does not match any of the source/destination addresses and direct it through the TOE. Verify via audit logs and packet captures that the TOE discards all traffic. Repeat the test, configuring the TOE based on combinations of specific source and wildcard destination addresses, wildcard source and specific destination addresses, and wildcard source and destination addresses.

Test 7: Configure the TOE to permit and log TCP traffic based on a specific source port. Generate IPv4 and IPv6 traffic that matches the specified port. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE with a specific destination port, and specific source and destination ports.

Test 8: Configure the TOE to discard and log TCP traffic based on a specific source port. Generate IPv4 and IPv6 traffic that matches the specified port. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE with a specific destination port, and specific source and destination ports.

Test 9: Configure the TOE to permit and log UDP traffic based on a specific source port. Generate IPv4 and IPv6 traffic that matches the specified port. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE with a specific destination port, and specific source and destination ports.

Test 10: Configure the TOE to discard and log UDP traffic based on a specific source port. Generate IPv4 and IPv6 traffic that matches the specified port. Verify via audit logs and packet captures that the TOE processes the packets as specified. Repeat the test, configuring the TOE with a specific destination port, and specific source and destination ports.

The evaluator was able to confirm that all tests listed above were able to be completed without failure.

5.10 Firewall (FFW)

5.10.1 **FFW_RUL_EXT.1 Stateful Traffic Filtering**

TSS

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

The evaluator shall verify that the TSS also include a narrative that identifies the components (e.g., active entity such as a process or task) involved in processing the network packets and describe the safeguards that would prevent packets flowing through the TOE without applying the ruleset in the event of a component failure. This could include the failure of a component, such as a process being terminated, or a failure within a component, such as memory buffers full and cannot process packets. The description shall also include a description how the TOE behaves in the situation where the traffic exceeds the amount of traffic the TOE can handle and how it is ensured that also in this condition stateful traffic filtering rules are still applied so that traffic does not pass that shouldn't pass according to the specified rules. (Ref. [6]).

See FPF_RUL_EXT.1.1.

Guidance Documentation

The guidance documentation associated with this requirement is assessed in the subsequent test evaluation activities. (Ref. [6]).

N/A.

Tests

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

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

Note: The remaining testing associated with application of the ruleset is addressed in the subsequent test evaluation activities. (Ref. [6]).

Test 1: Configure the TOE to block network traffic by default. Start directing network traffic through the TOE. Reboot the TOE and wait until it has fully initialised. Verify that the TOE does not let any network traffic pass through it. The evaluator was able to verify that the TOE did not let any network traffic pass through it until it was fully initialised.

Test 2: Configure the TOE to allow network traffic by default. Start directing network traffic through the TOE. Reboot the TOE and wait until it has fully initialised. Verify that the TOE does not let any network traffic pass through it while initialising and only permits traffic through once initialisation is complete. The evaluator was able to verify that the TOE did not let any network traffic pass through it until it was fully initialised. No traffic was allowed through during the initialisation process.

5.10.2 **FFW_RUL_EXT.1.2/FFW_RUL_EXT.1.3/FFW_RUL_EXT.1.4**

TSS

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

- IPv4 (RFC 791)
  - Source address
  - Destination Address
  - Protocol

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

The evaluator shall verify that each rule can identify the following actions: permit or drop with the option to log the operation. The evaluator shall verify that the TSS identifies all interface types subject to the stateful packet filtering policy and explains how rules are associated with distinct network interfaces. (Ref. [6])

See FPF_RUL_EXT.1.5.

Guidance Documentation

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

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

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

The evaluator shall verify that the guidance documentation explains how rules are associated with distinct network interfaces. (Ref. [6]).

Chapter 12, 13 and 11 respectively of the Evaluated Configuration Guide (Ref. [18,19,20]) identifies the protocols and the attributes listed above as being configurable within stateful traffic filtering. It provides a table (Table 13) that displays the protocol and the fields that is able to be configured.

Tests

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

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

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

Note that these test activities should be performed in conjunction with those of FFW_RUL_EXT.1.9 where the effectiveness of the rules is tested. The test activities for FFW_RUL_EXT.1.9 define the protocol/attribute combinations required to be tested. If those combinations are configured manually, that will fulfil the objective of these test activities, but if those combinations are configured otherwise (e.g., using automation), these test activities may be necessary in order to ensure the guidance is correct and the full range of configurations can be achieved by a TOE administrator. (Ref. [6]).

Test 1: Configure the TOE to permit and log traffic for the first firewall attribute listed in this SFR. Generate traffic matching the attribute and direct it through the TOE. Verify that the TOE logs the traffic and handles it as configured. Repeat the test while modifying the TOE configuration to explicitly drop traffic matching the firewall attribute. Repeat again after removing any explicit firewall rules and verify that the TOE drops the traffic by default. Repeat all of these steps for each subsequent firewall attribute listed in this SFR.

Test 2: Not applicable as only one distinct network interface type is supported.

5.10.3 **FFW_RUL_EXT.1.5**

TSS

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

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

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

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

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

The evaluator shall verify that the TSS describes how established stateful sessions are removed. The TSS shall describe how connections are removed for each protocol based on normal completion and/or timeout conditions. The TSS shall also indicate when session removal becomes effective (e.g., before the next packet that might match the session is processed). (Ref. [6])

Section 7.8 of the ST (Ref. [13]) explains that the TOE accepts network packets if it matches an established TCP, UDP or ICMP session using:

- TCP: source and destination addresses, source and destination ports, sequence number, flags
- UDP: source and destination addresses, source and destination ports
- ICMP: source and destination addresses, type, code

The TOE will remove existing traffic flows due to session inactivity timeout, or completion of the session.

Guidance Documentation

The evaluator shall verify that the guidance documentation describes stateful session behaviours. For example, a TOE might not log packets that are permitted as part of an existing session. (Ref. [6]).

The evaluator noted that links have been provided to the “Traffic Processing on SRX Series Devices Overview” online guidance in Chapter “Configuring Security Flow Policies” of the guidance documents (Ref. [18,19,20]). This document describes in stateful session behaviours.

Tests

The following tests shall be run using IPv4 and IPv6.

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

sequence number, flags) one at a time in order to verify that the altered packets are not accepted as part of the established session.

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

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

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

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

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

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

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

Test 1: Configure the TOE to permit and log TCP traffic. Start initiating a TCP session that passes through the TOE, then generate TCP packets with unexpected flags during the initialisation. Verify that the TOE does not accept these unexpected packets as part of the initialisation and the packet is logged. Finish initiating a TCP session, then generate TCP packets with altered session determining attributes. The evaluator verified that for each altered packet, the TOE did not accept them as part of the initial session and the packet is logged.

Test 2: Initialise a valid TCP session that passes through the TOE, then tear it down. Send a TCP packet through the TOE that matches the session determining attributes from the previous session and verify that the TOE does not accept it as part of the initial session and the packet is logged. The evaluator was able to verify this took place.

Test 3: Initialise a valid TCP session that passes through the TOE, then wait until it times out. Send a TCP packet through the TOE that matches the session determining attributes from the previous session and verify that the TOE does not accept it as part of the initial session and the packet is logged. The evaluator was able to verify that the packet was logged.

Test 4: Configure the TOE to permit and log UDP traffic. Generate a UDP packet and direct it through the TOE to set up a UDP session. Then generate UDP packets with altered session determining attributes compared to the first packet. Verify that for each altered packet, the TOE does not accept them as part of the initial session and the packet is logged. The evaluator was able to verify that the altered packet was logged.

Test 5: Generate a UDP packet and direct it through the TOE to set up a UDP session, then wait until it times out. Send a UDP packet through the TOE that matches the session determining attributes from the previous session and verify that the TOE does not accept it as part of the initial session and the packet is logged. The evaluator was able to verify that the TOE did not accept the packet as part of the initial session and the packet was logged.

Test 6: Configure the TOE to permit and log ICMP traffic. Generate an ICMP packet and direct it through the TOE to set up an ICMP session. Then generate ICMP packets with altered session determining attributes compared to the first packet. Verify that for each altered packet, the TOE does not accept them as part of the initial session and the packet is logged. The evaluator was able to verify that the TOE did not accept the packet as part of the initial session and the packet was logged.

Test 7: Generate an ICMP packet and direct it through the TOE to set up an ICMP session, then command the TOE to tear down the session. Send a ICMP packet through the TOE that matches the session determining attributes from the previous session and verify that the TOE does not accept it as part of the initial session and the packet is logged. The evaluator was able to verify that the TOE did not accept the packet as part of the initial session and the packet was logged.

Test 8: Generate an ICMP packet and direct it through the TOE to set up an ICMP session, then wait until it times out. Send a ICMP packet through the TOE that matches the session determining attributes from the previous session and verify that the TOE does not accept it as part of the initial session and the packet is logged. The evaluator was able to verify that the TOE did not accept the packet as part of the initial session and the packet was logged.

5.10.4 **FFW_RUL_EXT.1.6**

TSS

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

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

b) Fragments that cannot be completely re-assembled

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

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

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

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

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

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

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

(Ref. [6])

Section 7.8 of the ST (Ref. [13]) explains that The TOE enforces the following default reject rules with logging on all network traffic:

- invalid fragments;
- fragmented IP packets which cannot be re-assembled completely;
- where the source address is equal to the address of the network interface where the network packet was received;
- where the source address does not belong to the networks associated with the network interface where the network packet was received;
- where the source address is defined as being on a broadcast network;
- where the source address is defined as being on a multicast network;
- where the source address is defined as being a loopback address;
- where the source address is a multicast;
- packets where the source or destination address is a link-local address;
- where the source or destination address is defined as being an address “reserved for future use” as specified in RFC 5735 for Ipv4;
- where the source or destination address is defined as an “unspecified address” or an address “reserved for future definition and use” as specified in RFC 3513 for Ipv6;
- with the IP options: Loose Source Routing, Strict Source Routing, or Record Route specified;
- packets are checked for validity. “Invalid fragments” are those that violate these rules:
    - No overlap
    - The total fragments in one packet should not be more than 62 pieces
    - The total length of merged fragments should not larger than 64k
    - All fragments in one packet should arrive in 2 seconds

- The total queued fragments has limitation, depending on the platform
- The total number of concurrent fragment processing for different packet has limitations depending on platform

Guidance Documentation

The evaluator shall verify that the guidance documentation describes packets that are discarded and potentially logged by default. If applicable protocols are identified, their descriptions need to be consistent with the TSS. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets (Ref. [6]).

Chapter 12, 13 and 11 respectively of the Evaluated Configuration Guides (Ref. [18][19][20]) provides TOE administrators with the configuration steps necessary to set-up the following rules:

- A default "deny-all" rule that drops all traffic not matching any other rule
- Drop fragmented IP packets and invalid fragments;
- Drop packets with source-address spoofing;
- Drop packets with IP options;
- Drop illegal or out-of-sequence TCP packets; and
- Drop unassigned IPv6 packets

All packets meeting the rules listed above are logged by default.

Tests

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

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

Test 2: For each of the cases above, the evaluator shall use any applicable guidance to enable dropped packet logging or counting. In each case above, the evaluator shall ensure that the rejected packet or packet fragment was recorded (either logged or an appropriate counter incremented) (Ref. [6]).

Test 1: Configure the TOE to log and drop any traffic matching the packets specified in this SFR. For each packet specified, generate the relevant network traffic and direct it through the TOE. Verify for each packet that the TOE logs the attempt and drops the traffic. The evaluator was able to ensure that unallowable packet or packet fragment was not passed through the TOE.

Test 2: This test was integrated into the test steps for Test 1.

5.10.5 **FFW_RUL_EXT.1.7**

TSS

The evaluator shall verify that the TSS explains how the following traffic can be dropped and counted or logged: a) Packets where the source address is equal to the address of the network interface where the network packet was received b) Packets where the source or destination address of the network packet is a link-local address c) Packets where the source address does not belong to the networks associated with the network interface where the network packet was received, including a description of how the TOE determines whether a source address belongs to a network associated with a given network interface (Ref. [12]).

Section 7.8 of the ST (Ref. [13]) indicates that the TOE enforces the following default reject rules with logging on all network traffic:

- where the source address is equal to the address of the network interface where the network packet was received;
- where the source address does not belong to the networks associated with the network interface where the network packet was received;
- where the source or destination address is a link-local address;

The TOE is configured to associate network interfaces to IP subnets. Source IP addresses are then associated with the network interface.

Guidance Documentation

The evaluator shall verify that the guidance documentation describes how the TOE can be configured to implement the required rules. If logging is configurable, the evaluator shall verify that applicable instructions are provided to configure auditing of automatically rejected packets (Ref. [6]).

The TOE enforces the following default reject rules with logging on all network traffic:

- where the source address is equal to the address of the network interface where the network packet was received;
- where the source address does not belong to the networks associated with the network interface where the network packet was received;
- where the source or destination address is a link-local address;

As per Chapter 13 of the guidance (Ref. [18][19][20]) this can be configured using the configuration item `set security policies default-policy deny-all`

Logging these dropped packets can be configured using the configuration provided in the "Logging the Dropped Packets Using Default Deny-all Option" section in Chapter 13 of the guidance (Ref. [18][19][20]).

Tests

The following tests shall be run using IPv4 and IPv6.

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

b) Test 2: The evaluator shall configure the TOE to drop and log network traffic where the source IP address of the packet fails to match the network reachability information of the interface to which it is targeted, e.g. if the TOE believes that network 192.168.1.0/24 is reachable through interface 2, network traffic with a source address from the 192.168.1.0/24 network should be generated and sent to an interface other than interface 2. The evaluator shall verify that the network traffic is dropped and a log message generated (Ref. [6]).

a) The evaluator generated network traffic where the source address matches the interface address on the TOE and direct it through the TOE. The TOE logged the attempts and dropped the packets.

b) The evaluator configured the TOE to log and drop spoofed traffic. Network traffic where the source address are from subnets that are not reachable via the TOE interfaces is generated. The TOE logged the attempts and dropped the packets.

5.10.6 **FFW_RUL_EXT.1.8**

TSS

The evaluator shall verify that the TSS describes the algorithm applied to incoming packets, including the processing of default rules, determination of whether a packet is part of an established session, and application of administrator defined and ordered ruleset (Ref. [6])

See FPF_RUL_EXT.1.6

Guidance Documentation

The evaluator shall verify that the guidance documentation describes how the order of stateful traffic filtering rules is determined and provides the necessary instructions so that an administrator can configure the order of rule processing (Ref. [6]).

The “Reordering Security Policies” link on page 146, 170 and 104 respectively of the Evaluated Configuration (Ref. [18,19,20]) states that:

“The concept of policy shadowing refers to the situation where a policy higher in the policy list always takes effect before a subsequent policy. Because the policy lookup always uses the first policy it finds that matches the five-part tuple of the source and destination zone, source and destination address, and application type, if another policy applies to the same tuple (or a subset of the tuple), the policy lookup uses the first policy in the list and never reaches the second one.

This reference also provides examples on how the ordering of the policies may be changed using the *insert* statement:

Reorder two existing policies by entering the following command:

```
insert security policies from-zone trust to-zone untrust policy permit-
mail before policy permit-all
```

Tests

Test 1: The evaluator shall devise two equal stateful traffic filtering rules with alternate operations – permit and drop. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation Test 2: The evaluator shall repeat the procedure above, except that the two rules should be devised where one is a subset of the other (e.g., a specific address vs. a network segment). Again, the evaluator should test both orders to ensure that the first is enforced regardless of the specificity of the rule. (Ref. [6]).

Test 1: Configure the TOE with two equal firewall rules that either permit or drop traffic. Set the permit rule as the first rule. Direct matching network traffic through the TOE. Verify that the TOE permits and logs the traffic. Modify the rule order so that the drop rule is the first rule. Direct matching network traffic through the TOE. The evaluator was able to verify that the TOE dropped and logged the traffic.

Test 2: Configure the TOE with two firewall rules where the permit rule matches a specific address and the deny rule matches a subnet containing the specific address. Set the permit rule as the first rule. Direct matching network traffic through the TOE. Verify that the TOE permits and logs the traffic. Modify the rule order so that the drop rule is the first rule. Direct matching network traffic through the TOE. The evaluator was able to verify that the TOE dropped and logged the traffic.

5.10.7 **FFW_RUL_EXT.1.9**

TSS

The evaluator shall verify that the TSS describes the process for applying stateful traffic filtering rules and also that the behavior (either by default, or as configured by the administrator) is to deny packets when there is no rule match unless another required conditions allows the network traffic (i.e., FFW_RUL_EXT.1.5 or FFW_RUL_EXT.2.1) (Ref. [6]).

See FPF_RUL_EXT.1.7

Guidance Documentation

The evaluator shall verify that the guidance documentation describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the guidance documentation provides the appropriate instructions to configure the behavior to deny packets with no matching rules (Ref. [6]).

As per Chapter 12, 13 and 11 respectively of the Evaluated Configuration Guide (Ref.) the default reject-all rule can be implemented via the following command:

```
set security policies default-policy deny-all
```

Tests

For each attribute in FFW_RUL_EXT.1.2, the evaluator shall construct a test to demonstrate that the TOE can correctly compare the attribute from the packet header to the ruleset, and shall demonstrate both the permit and deny for each case. It shall also be verified that a packet is dropped if no matching rule can be identified for the packet. The evaluator shall check the log in each case to confirm that the relevant rule was applied. The evaluator shall record a packet capture for each test to demonstrate the correct TOE behaviour. (Ref. [6]).

This test is subsumed in its entirety by the tests for FFW_RUL_EXT.1.2. No tests are implemented here.

5.10.8 **FFW_RUL_EXT.1.10**

TSS

The evaluator shall verify that the TSS describes how the TOE tracks and maintains information relating to the number of half-open TCP connections. The TSS should identify how the TOE behaves when the administratively defined limit is reached and should describe under what circumstances stale half-open connections are removed (e.g. after a timer expires) (Ref. [6]).

Section 7.8 of the ST (Ref. [13]) explains that the TOE can be configured to drop connection attempts after a defined number of half-open TCP connections using the Junos screen ‘tcp syn-flood’, which provides both source and destination thresholds on the number of uncompleted TCP connections, as well as a timeout period.

The source threshold option allows administrators to specify the number of SYN segments received per second from a single source IP address—regardless of the destination IP address—before Junos OS begins dropping connection requests from that source.

Similarly, the destination threshold option allows administrators to specify the number of SYN segments received per second for a single destination IP address before Junos OS begins dropping connection requests to that destination. The timeout option allows administrators to set the maximum length of time before an uncompleted connection is dropped from the queue.

Guidance Documentation

The evaluator shall verify that the guidance documentation describes the behaviour of imposing TCP half-open connection limits and its default state if unconfigured. The evaluator shall verify that the guidance clearly indicates the conditions under which new connections will be dropped e.g. per-destination or per-client (Ref. [6]).

The evaluator noted that links have been provided to the “Traffic Processing on SRX Series Devices Overview” online guidance in Chapter “Configuring Security Flow Policies” of the CC guidance documents (Ref. [18,19,20]). This document describes in detail how to impose TCP half-open connection limits and the default state.

Tests

The following tests shall be run using IPv4 and IPv6.

Test 1: The evaluator shall define a TCP half-open connection limit on the TOE. The evaluator shall generate TCP SYN requests to pass through the TOE to the target system using a randomised source IP address and common destination IP address. The number of SYN requests should exceed the TCP half-open threshold defined on the TOE. TCP SYN-ACK messages should not be acknowledged. The evaluator shall verify through packet capture that once the defined TCP half-open threshold has been reached, subsequent TCP SYN packets are not transmitted to the target system. The evaluator shall verify that when the configured threshold is reached that, depending upon the selection, either a log entry is generated or a counter is incremented. (Ref. [6]).

This test was subsumed in its entirety by the tests for FFW_RUL_EXT.1.2. No tests were implemented here.

5.10.9 **FFW_RUL_EXT.2.1**

TSS

The evaluator shall verify that the TSS identifies the protocols that can cause the automatic creation of dynamic packet filtering rules. In some cases rather than creating dynamic rules, the TOE might establish stateful sessions to support some identified protocol behaviors.

The evaluator shall verify that the TSS explains the dynamic nature of session establishment and removal. The TSS also shall explain any logging ramifications.

The evaluator shall verify that for each of the protocols selected, the TSS explains the dynamic nature of session establishment and removal specific to the protocol. (Ref. [6]).

Section 7.8 of the ST (Ref. [13]) indicates that the TOE supports FTP (RFC 959) to dynamically establish sessions allowing network traffic according to Administrator rules. Session events will be

logged in accordance with ‘log’ operations defined in the rules. Source and destination addresses, source and destination ports, transport layer protocol, and TOE Interface are recorded in each log record.

Junos implements what is referred to as an Application Layer gateway (ALG) that inspects FTP traffic to determine the port number used for data sessions. The ALG permits data traffic for the duration of the session, closing the port when the session ends. In this context, “session” refers to the TCP data transfer connection, not the duration of the FTP control session. Junos implements ALGs for a number of protocols.

Guidance Documentation

The evaluator shall verify that the guidance documentation describes dynamic session establishment capabilities. The evaluator shall verify that the guidance documentation describes the logging of dynamic sessions consistent with the TSS. (Ref. [6]).

As per the “Configuring Traffic Filter Rules” section in Chapters 12, 13 and 11 respectively of the Evaluated Configuration Guide (Ref. [18,19,20]) the TOE may be configured to permit or deny dynamic FTP sessions. These sessions are handled in an identical manner to all other protocols supported by the TOE.

Session may be logged, permitted or denied and limited may be put in place to prevent sessions above the threshold.

As per the Evaluated Configuration Guide (Ref. [18,19,20]) session initialization and closure are logged per the applicable traffic policy that the dynamic session is associated with.

Tests

Test 1: The evaluator shall define stateful traffic filtering rules to permit and log traffic for each of the supported protocols and drop and log TCP and UDP ports above 1024. Subsequently, the evaluator shall establish a connection for each of the selected protocols in order to ensure that it succeeds. The evaluator shall examine the generated logs to verify they are consistent with the guidance documentation. Test 2: Continuing from Test 1, the evaluator shall determine (e.g., using a packet sniffer) which port above 1024 opened by the control protocol, terminate the connection session, and then verify that TCP or UDP (depending on the protocol selection) packets cannot be sent through the TOE using the same source and destination addresses and ports Test 3: For each additionally supported protocol, the evaluator shall repeat the procedure above for the protocol. In each case the evaluator must use the applicable RFC or standard in order to determine what range of ports to block in order to ensure the dynamic rules are created and effective. (Ref. [6]).

Test 1: Configure the TOE to permit and log FTP traffic, and drop and log all TCP/UDP ports above port 1024. The evaluator was able to establish an FTP session that passed through the TOE and verified that it was successful.

Test 2: Terminate the FTP session that was established in Test 1. Generate a TCP packet that matches the addresses and TCP ports above 1024 that were opened by the FTP session and attempt to direct it through the TOE. The evaluator was able to verify that the TOE dropped and logged the generated traffic.

Test 3: Not applicable as only one network protocol is supported.

5.11 User Data Protection (FDP)

5.11.1 **FDP_RIP.2 Full Residual Information Protection**

TSS

"Resources" in the context of this requirement are network packets being sent through (as opposed to "to", as is the case when a security administrator connects to the TOE) the TOE. The concern is that once a network packet is sent, the buffer or memory area used by the packet still contains data from that packet, and that if that buffer is re-used, those data might remain and make their way into a new packet. The evaluator shall check to ensure that the TSS describes packet processing to the extent that they can determine that no data will be reused when processing network packets. The evaluator shall ensure that this description at a minimum describes how the previous data are zeroized/overwritten, and at what point in the buffer processing this occurs (Ref. [12]).

As per Section 7.7 of the ST (Ref. [13]), "The only resource made available to information flowing through a TOE is the temporary storage of packet information when access is requested and when information is being routed. User data is not persistent when resources are released by one user/process and allocated to another user/process. Temporary storage (memory) used to build network packets is erased when the resource is called into use by the next user/process. Junos knows, and keeps track of, the length of the packet. This means that when memory allocated from a previous user/process arrives to build the next network packet, Junos is aware of when the end of the packet is reached and pads a short packet with zeros accordingly. Therefore, no residual information from packets in a previous information stream can traverse through the TOE."

5.12 Trusted path/channels (FTP)

5.12.1 **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 7.1.3 of the ST (Ref. [13]), the TOE provides an SSH server to support Trusted Channels using SSHv2 protocol to protect communications 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.

The TOE also implements IPsec as a trusted channel for VPN communications and can act as both initiator and responder.

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. External logging may also be carried out using IKE/IPsec.

The Evaluated Configuration Guide (Ref. [18,19,20]) provides instructions for configuring SSH and the transfer of logs using NETCONF via SSH and/or IPsec/IKE. 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]).

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]).

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 logged in to the TOE via SSH as a user with security administrator privileges. The evaluator then generated traffic by viewing the log file that was stored locally on the TOE. The traffic that's generated between the TOE and the remote SSH client was examined by the evaluator to ensure that no data was sent in plain-text. The evaluator then physically disconnected the connection between the TOE and the remote server for a period less than the product of the ClientAliveInterval and ClientAliveCountMax values configured at the TOE. The traffic between the TOE and the client was examined to ensure that no data packets were transmitted during the period of disconnection. The evaluator reconnected the physical link between the TOE and the client and ensures that connectivity was re-established. Having ensured that the connectivity was re-established, the evaluator disconnected the physical link again, this time waiting for a period that was greater than the product of the ClientAliveInterval and ClientAliveCountMax values configured at the TOE. The physical link was reconnected, and the developer ensured that the connection was terminated and no further communication was possible.

5.13 Intrusion Prevention (IPS)

5.13.1 **IPS_NTA_EXT.1 Network Traffic Analysis**

IPS_NTA_EXT.1.1

TSS

The evaluator shall verify that the TSS explains the TOE's capability of analyzing IP traffic in terms of the TOE's policy hierarchy (precedence). The TSS should identify if the TOE's policy hierarchy order is configurable by the administrator for IPS policy elements (known-good lists, known-bad lists, signature-based rules, and anomaly-based rules).

Regardless of whether the precedence is configurable, the evaluator shall verify that the TSS describes the default precedence as well as the IP analyzing functions supported by the TOE (Ref. [12]).

Section 7.9 of the ST (Ref. [13]) explains that an Intrusion Detection and Prevention (IDP) policy is made up of rule bases, and each rule base contains a set of rules that specify rule parameters, such as traffic match conditions, action, and logging requirements. IDP policies can then be associated to firewall policies. IDP can be invoked on a firewall rule by rule basis for maximum granularity. Only firewall policies marked for IDP will be processed by IDP engine, all other rules will only be processed by the firewall.

Firewall Policies match Source Zone, Destination Zone, Source IP, Destination IP, Source Port, Destination Port, and Protocol. Interface and VLAN matching can be achieved through the use of zones. Rules are organized into a firewall policy rulebase. Within IPS Policies, further matching for specific attacks is done on Source Zone, Destination Zone, Source IP, Destination IP, Source Port, Destination Port, and Protocol. Interface matching can be achieved through the use of zones. Attack Actions are configurable on a rule by rule basis. Rules within policies are processed in an Administrator-defined order when network traffic flows through the TOE network interfaces.

Guidance Documentation

The evaluator shall verify that the guidance describes the default precedence. If the precedence is configurable. The evaluator shall verify that the guidance explains how to configure the precedence (Ref. [12]).

The link to the “Reordering Security Policies” reference in the Evaluated Configuration (Ref. [18,19,20]) states that: *The concept of policy shadowing refers to the situation where a policy higher in the policy list always takes effect before a subsequent policy. Because the policy lookup always uses the first policy it finds that matches the five-part tuple of the source and destination zone, source and destination address, and application type, if another policy applies to the same tuple (or a subset of the tuple), the policy lookup uses the first policy in the list and never reaches the second one.*

This reference also provides examples on how the ordering of the policies may be changed using the *insert* statement:

Reorder two existing policies by entering the following command:

```
insert security policies from-zone trust to-zone untrust policy permit-
mail before policy permit-all
```

Tests

There are no test EAs for this element (Ref. [12]).

N/A

IPS_NTA_EXT.1.2

TSS

The evaluator shall verify that the TSS indicates that the following protocols are supported: • Ipv4 • Ipv6 • ICMPv4 • ICMPv6 • TCP • UDP

The evaluator shall verify that the TSS describes how conformance with the identified protocols has been determined by the TOE developer. (e.g., third party interoperability testing, protocol compliance testing) (Ref. [12]).

Section 7.9 of the ST (Ref. [13]) confirms support for the following network traffic protocols:

- RFC 792 (ICMPv4)
- RFC 4443 (ICMPv6)
- RFC 791 (Ipv4)
- RFC 2460 (Ipv6)
- RFC 793 (TCP)
- RFC 768 (UDP)

Conformance to these RFCs is demonstrated by the protocol compliance testing by the product QA team.

Guidance Documentation

There are no guidance EAs for this element (Ref. [12]).

N/A

Tests

There are no test EAs for this element (Ref. [12]).

N/A

IPS_NTA_EXT.1.3

TSS

The evaluator shall verify that the TSS identifies all interface types capable of being deployed in the modes of promiscuous, and or inline mode as well as the interfaces necessary to facilitate each deployment mode (at a minimum, the interfaces need to support inline mode). The TSS should also provide descriptions how the management interface is distinct from sensor interfaces (Ref. [6]).

Section 7.9 of the ST (Ref. [13]) indicates that the TOE is capable of inspecting all traffic passing through the TOE's Ethernet interfaces (inline mode). Ethernet interfaces can be assigned to Zones on which firewall and IDP policies are predicated. The TOE supports management through the console port, as well as through a dedicated Ethernet management port whose traffic is never processed for routing. Remote management of the TOE can also be performed via SSH.

Guidance Documentation

The evaluator shall verify that the operational guidance provides instructions on how to deploy each of the deployment methods outlined in the TSS. The evaluator shall also verify that the operational guidance provides instructions of applying IPS policies to interfaces for each

deployment mode. If the management interface is configurable the evaluator shall verify operational guidance explains how to configure the interface into a management interface.

The evaluator shall verify that the operational guidance explains how the TOE sends commands to remote traffic filtering devices. (Ref. [6]).

The TOE's primary interfaces are deployed in inline mode by default, and therefore, do not require any additional configuration. No interfaces can operate in promiscuous mode. All ethernet interfaces present on the device may be configured into management interfaces by permitting the SSH service on the interface.

IDP rules are associated with zones, and not a single interface. The IDP engine listens on all interfaces.

The TOE does not utilise remote filtering devices.

Tests

Testing for this element is performed in conjunction with testing where promiscuous and inline interfaces are tested (Ref. [12]).

N/A

5.14 IP Blocking (IPS_IPB_EXT)

5.14.1 **IPS_IPB_EXT.1 IP Blocking**

TSS

The evaluator shall verify how good/bad lists affect the way in which traffic is analyzed with respect to processing packets. The evaluator shall also very that the TSS provides details for the attributes that create a known good list, a known bad list, and their associated rules, including how to define the source or destination IP address (e.g. a single IP address or a range of IP addresses).

If the TSF uses address types other than a single IP or a range of IP addresses (e.g. MAC addresses), the evaluator shall check that the TSS explains what configurations would cause non-IP lists of known-good and known-bad addresses to take precedence over IP-based address lists.

The evaluator shall also verify that the TSS identifies all the roles and level of access for each of those roles that have been specified in the requirement. (Ref. [6]).

Section 7.9 of the ST (Ref. [13]) indicates that the TOE supports the definition by administrators of known-good and known-bad lists of source and/or destination addresses at the firewall rule level as described in Section 7.8 of the ST. Address ranges can be defined by creating address book entries and attaching them to firewall policies.

Guidance Documentation

The evaluator shall verify that the administrative guidance provides instructions with how each role specified in the requirement can create, modify and delete the attributes of a known good and known bad lists.

If the TSF uses address types other than a single IP or a range of IP addresses (e.g. MAC addresses), the evaluator shall check that the operational guidance includes instructions for any configurations that would cause non-IP lists of known-good and known-bad addresses to take precedence over IP-based address lists. (Ref. [6]).

The Evaluated Configuration Guides (Ref. [18,19,20]) provides a link on Page 118 which provides further information on configuring known good and bad lists.

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the instructions in the operational guidance to create a known-bad address list. Using a single IP address, a list of addresses or a range of addresses from that list, the evaluator shall attempt to send traffic through the TOE that would otherwise be allowed by the TOE and observe the TOE automatically drops that traffic.

Test 2: The evaluator shall use the instructions in the operational guidance to create a known-good address list. Using a single IP address, a list of addresses or a range of addresses from that list, the evaluator shall attempt to send traffic that would otherwise be denied by the TOE and observe the TOE automatically allowing traffic.

Test 3: The evaluator shall add conflicting IP addresses to each list and ensure that the TOE handles conflicting traffic in a manner consistent with the precedence in IPS_NTA_EXT.1.1. (Ref. [6]).

Test 1: Configure the TOE to allow traffic by default and send network traffic through it. Then set up a known-bad address list on the TOE and configure the TOE to block any traffic from that list. Send traffic matching the known-bad address list through the TOE and verify that the traffic is dropped. The evaluator was able to verify that the traffic was dropped.

Test 2: Delete any existing address books on the TOE and configure it to block traffic by default. Send network traffic through to verify this behaviour. Then set up a known-good address list on the TOE and configure the TOE to allow any traffic from that list. Send traffic matching the known-good address list through the TOE and verify that the traffic is allowed to pass. The evaluator was able to verify that the traffic was able to pass.

Test 3: Re-add the known-bad address list and modify the TOE configuration with a rule to block traffic from the known-bad list before allowing traffic from the known-good list. Send traffic through the TOE that matches both lists and verify that the traffic is dropped. The evaluator was able to verify that the traffic was dropped.

Modify the precedence of the policies on the TOE to allow known-good traffic before blocking known-bad traffic. Send traffic through the TOE that matches both lists and verify that the traffic is now permitted to pass. The evaluator was able to verify that the traffic was permitted to pass.

5.15 Signature-Based IPS Functionality (IPS_SBD_EXT)

5.15.1 **IPS_SBD_EXT.1 Signature-Based IPS Functionality**

IPS_SBD_EXT.1.1

TSS

The evaluator shall verify that the TSS describes what is comprised within a signature rule.

The evaluator shall verify that each signature can be associated with a reaction specified in IPS_SBD_EXT.1.5.

The evaluator shall verify that the TSS identifies all interface types capable of applying signatures and explains how rules are associated with distinct network interfaces.

Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used) they can be treated collectively as a distinct network interface. (Ref. [6]).

Section 7.9 of the ST (Ref. [13]) explains that the TOE uses Attack Objects to define signature rules. Attack Objects use context-based matching to match regular expressions in specific locations where they occur. Attack Objects can be in turn composed of multiple signatures and protocol anomalies, including logical expressions between signatures for compound matching.

Section 7.9 of the ST (Ref. [13]) lists protocol header-field values that can be used to define signatures using the command “set security idp custom-attack”, along with the actions (allow/block) using the command “set security idp idp-policy”, that the TOE will perform when a match is found in the processed packets.

Section 7.9 of the ST (Ref. [13]) indicates that the TOE is capable of inspecting all traffic passing through the TOE’s Ethernet interfaces (inline mode). Ethernet interfaces can be assigned to Zones on which firewall and IDP policies are predicated.

Guidance Documentation

The evaluator shall verify that the operational guidance provides instructions with how to create and/or configure rules using the following protocols and header inspection fields:

- IPv4: Version; Header Length; Packet Length; ID; IP Flags; Fragment Offset; Time to Live (TTL); Protocol; Header Checksum; Source Address; Destination Address; and IP Options.

- IPv6: Version; traffic class; flow label; payload length; next header; hop limit; source address; destination address; routing header; home address options.

- ICMP: Type; Code; Header Checksum; and Rest of Header (varies based on the ICMP type and code).

- ICMPv6: Type; Code; and Header Checksum.

- TCP: Source port; destination port; sequence number; acknowledgement number; offset; reserved; TCP flags; window; checksum; urgent pointer; and TCP options.

- UDP: Source port; destination port; length; and UDP checksum.

The evaluator shall verify that the operational guidance provides instructions with how to select and/or configure reactions specified in IPS_SBD_EXT.1.5 in the signature rules.

(Ref. [6]).

The evaluator examined the Intrusion Detection and Prevention User Guide (Ref. [21]) that is referenced in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18,19,20].

It provides administrators with configuration example on how to create a "custom attack" to detect each of the protocols and fields specified above.

The evaluator examined the Intrusion Detection and Prevention User Guide (Ref. [21]) that is referenced in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18,19,20].

The guide provides administrators with example commands and configuration data on how to configure an IDP rule to perform "allow" or "drop" operations.

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet header signatures can be created and/or configured with the selected and/or configured reactions specified in IPS_SBD_EXT.1.5 for each of the attributes listed below. Each attribute shall be individually assigned to its own unique signature:

• IPv4: Version; Header Length; Packet Length; ID; IP Flags; Fragment Offset; Time to Live (TTL); Protocol; Header Checksum; Source Address; Destination Address; and IP Options.

• IPv6: Version; traffic class; flow label; payload length; next header; hop limit; source address; destination address; routing header; home address options.

• ICMP: Type; Code; Header Checksum; and Rest of Header (varies based on the ICMP type and code).

• ICMPv6: Type; Code; and Header Checksum.

• TCP: Source port; destination port; sequence number; acknowledgement number; offset; reserved; TCP flags; window; checksum; urgent pointer; and TCP options.

• UDP: Source port; destination port; length; and UDP checksum.

The evaluator shall generate traffic to trigger a signature and shall then use a packet sniffer to capture traffic that ensures the reactions of each rule are performed as expected. Test 2: The evaluator shall repeat the test above to ensure that signature-based IPS policies can be defined for each distinct network interface type capable of applying signatures as supported by the TOE. (Ref. [12])

Test 1: Configure the TOE to allow traffic by default and send network traffic through it. Then set up a known-bad address list on the TOE and configure the TOE to block any traffic from that list. Send traffic matching the known-bad address list through the TOE and verify that the traffic is dropped. The evaluator was able to verify that the traffic was dropped.

Test 2: Delete any existing address books on the TOE and configure it to block traffic by default. Send network traffic through to verify this behaviour. Then set up a known-good address list on the TOE and configure the TOE to allow any traffic from that list. Send traffic matching the known-good address list through the TOE and verify that the traffic is allowed to pass. The evaluator was able to verify that the traffic was allowed to pass.

Test 3: Re-add the known-bad address list and modify the TOE configuration with a rule to block traffic from the known-bad list before allowing traffic from the known-good list. Send traffic through the TOE that matches both lists and verify that the traffic is dropped. The evaluator was able to verify that the traffic was dropped.

Modify the precedence of the policies on the TOE to allow known-good traffic before blocking known-bad traffic. Send traffic through the TOE that matches both lists and verify that the traffic is now permitted to pass. The evaluator was able to verify that the traffic was permitted to pass.

IPS_SBD_EXT.1.2

TSS

The evaluator shall verify that the TSS describes what is comprised within a string-based detection signature.

The evaluator shall verify that each packet payload string-based detection signature can be associated with a reaction specified in IPS_SBD_EXT.1.5. (Ref. [6]).

As per Section 7.9 of the ST (Ref. [13]), for TCP payload inspection, Junos OS provides pre-defined attack signatures to detect FTP commands, HTTP commands and content, and STMP states. Alternative, administrators can define custom-attack signatures for these application layer protocols using the command “set security idp custom-attack”.

Guidance Documentation

The evaluator shall verify that the operational guidance provides instructions with how to configure rules using the packet payload string-based detection fields defined in IPS_SBD_EXT.1.2.

The evaluator shall verify that the operational guidance provides instructions with how to configure reactions specified in IPS_SBD_EXT.1.5 for each string-based detection signature.

The evaluator shall verify that the operational guidance provides instructions with how rules are associated with distinct network interfaces that are capable of being associated with signatures. (Ref. [6]).

The evaluator examined the Intrusion Detection and Prevention User Guide (Ref. [21]) that is referenced in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18,19,20].

The guide provides configuration example and guidance on how to develop and implement custom rules using each of the payload string-based detection fields defined in IP_SBD_EXT.1.2.

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet payload string-based detection rules can be assigned to the reactions specified in IPS_SBD_EXT.1.5 using the attributes specified in IPS_SBD_EXT.1.2. However it is not required (nor is it feasible) to test all possible strings of protocol data, the evaluator shall ensure that a selection of strings in the requirement is selected to be tested. At a minimum at least one string using each of the following attributes from IPS_SBD_EXT.1.2 should be tested for each protocol. The evaluator shall generate packets that match the string in the rule and observe the corresponding reaction is as configured.

- Test at least one string of characters for ICMPv4 data: beyond the first 4 bytes of the ICMP header.
- Test at least one string of characters for ICMPv6 data: beyond the first 4 bytes of the ICMP header.
- TCP data (characters beyond the 20 byte TCP header):
    - Test at least one FTP (file transfer) command: help, noop, stat, syst, user, abort, acct, allo, appe, cdup, cwd, dele, list, mkd, mode, nlst, pass, pasv, port, pass, quit, rein, rest, retr, rmd, rnfr, rnto, site, smnt, stor, stou, stru, and type.
    - HTTP (web) commands and content:
        - Test both GET and POST commands
        - Test at least one administrator-defined strings to match URLs/URIs, and web page content.
    - Test at least one SMTP (email) state: start state, SMTP commands state, mail header state mail body state, abort state.
    - Test at least one string in any additional attribute type defined within the “other types of TCP payload inspection” assignment, if any other types are specified.

Test 2: The evaluator shall repeat Test 1 above to ensure that signature-based IPS policies can be defined for each distinct network interface type capable of applying signatures as supported by the TOE.(Ref. [12])

Test 1: Configure the TOE with a custom IPS signature to detect the first attribute specified in this SFR. Generate network traffic that contains the attribute specified. The evaluator was able to verify that the TOE detected that the traffic that contained the attribute, logs the event and blocks the traffic as configured. The evaluator proceeded to clear the existing configuration and repeated the process for every remaining attribute specified.

Test 2: Not applicable as only one distinct network interface type is supported.

IPS_SBD_EXT.1.3

TSS

The evaluator shall verify that the TSS describes how the attacks defined in IPS_SBD_EXT.1.3 are processed by the TOE and what reaction is triggered when these attacks are identified. (Ref. [6]).

Section 7.9 of the ST (Ref. [13]) explains how administrators can configure the TOE to detect the signature attacks defined in IPS_SBD_EXT.1.3, mostly by using Junos predefined screen options.

The default action for the screens is to drop the packets. To allow the packets through, the “alarm-without-drop” action can be defined using the command “set security screen ids-option”.

Guidance Documentation

The evaluator shall verify that the operational guidance provides instructions with configuring rules to identify the attacks defined in IPS_SBD_EXT.1.3 as well as the reactions to these attacks as specified in IPS_SBD_EXT.1.5. (Ref. [6]).

This is addressed in the item IPS_SBD_EXT.1-2.

Tests

The evaluator shall create and/or configure rules for each attack signature in IPS_SBD_EXT.1.3. For each attack, the TOE should apply its corresponding signature and enable it to each distinct network interface type capable of applying the signatures. The evaluator shall use packet captures to ensure that the attack traffic is detected by the TOE and a reaction specified in IPS_SBD_EXT.1.5 is triggered and stops the attack. Each attack should be performed one after another so as to ensure that its corresponding signature successfully identified and appropriately reacted to a particular attack. (Ref. [12])

The evaluator configured the TOE with a custom IPS rule to detect the first signature specified in this SFR. Generated network traffic that contains the signature specified. Then verified that the TOE detected the traffic that contains the signature, logs the event and blocks the traffic as configured. The evaluator then cleared the existing configuration and repeat the process for every remaining signature specified.

IPS_SBD_EXT.1.4

TSS

The evaluator shall verify that the TSS describes how the attacks defined in IPS_SBD_EXT.1.4 are processed by the TOE and what reaction is triggered when these attacks are identified. (Ref. [6]).

Section 7.9 of the ST (Ref. [13]) explains how administrators can configure the TOE to detect the signature attacks defined in IPS_SBD_EXT.1.4, mostly by using Junos predefined screen options.

The default action for the above screens is to drop the packets. To allow the packets through, the "alarm-without-drop" action can be defined using the command "set security screen ids-option".

Guidance Documentation

The evaluator shall verify that the operational guidance provides instructions with configuring rules to identify the attacks defined in IPS_SBD_EXT.1.4 as well as the reactions to these attacks as specified in IPS_SBD_EXT.1.5. (Ref. [6]).

The evaluator examined the Intrusion Detection and Prevention User Guide (Ref. [21]) that is referenced in the "IDP Extended Package Configuration Overview" section of the Evaluated Configuration Guide (Ref. [18,19,20].

The guide provides configuration examples and guidance on rules to identify the attacks defined in IPS_SBD_EXT.1.4 as well as reactions to these attacks as specified in IPS_SBD_EXT.1.5.

Tests

The evaluator shall configure individual signatures for each attack in IPS_SBD_EXT.1.4. For each attack, the TOE should apply its corresponding signature and enable it to each distinct network interface type capable of applying signatures. The evaluator shall use packet captures to ensure that the attack traffic is detected by the TOE and a reaction specified in IPS_SBD_EXT.1.5 is triggered and stops the attack. Each attack should be performed one after another so as to ensure that its corresponding signature successfully identified and appropriately reacted to a particular attack. (Ref. [12])

The evaluator configured the TOE with a custom IPS rule to detect the first signature specified in this SFR. Generated network traffic that contains the signature specified. Verified that the TOE detects the traffic that contains the signature, logs the event and blocks the traffic as configured. The evaluator then cleared the existing configuration and repeat the process for every remaining signature specified.

IPS_SBD_EXT.1.5

TSS

There are no TSS EAs for this element. (Ref. [6]).

N/A

Guidance Documentation

The guidance EAs for this element are performed in conjunction with IPS_SBD_EXT.1.1, IPS_SBD_EXT.1.3, and IPS_SBD_EXT.1.4. (Ref. [6]).

N/A

Tests

The test EAs for this element are performed in conjunction with those for IPS_SBD_EXT.1.1, IPS_SBD_EXT.1.2, IPS_SBD_EXT.1.3, and IPS_SBD_EXT.1.4. (Ref. [12])

N/A

IPS_SBD_EXT.1.6

TSS

There are no TSS EAs for this element. (Ref. [[6]).

N/A

Guidance Documentation

The evaluator shall verify that the operational guidance provides configuration instructions, if needed, to detect payload across multiple packets. (Ref. [6]).

The evaluator examined the Intrusion Detection and Prevention User Guide that is referenced in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [2][3]).

The guide provides configuration examples and guidance on rules to identify the attacks defined in IPS_SBD_EXT.1.4 as well as reactions to these attacks as specified in IPS_SBD_EXT.1.5.

Tests

The evaluator shall repeat one of the tests in IPS_SBD_EXT.1.2 Test 1 but generate multiple non-fragmented packets that contain the string in the rule defined. The evaluator shall verify that the malicious traffic is still detected when split across multiple non-fragmented packets. (Ref. [12])

The evaluator configured the TOE with a custom IPS signature that works with data streams and detects two discrete text values. Initiate a TCP session through the TOE and sent the two discrete text values in two separate packets. Verified that the TOE detects the stream reassembly of the combined text values, logs the event and permits the traffic to pass as configured.

5.16 Anomaly-Based IPS Functionality (IPS_ABD_EXT)

5.16.1 **IPS_ABD_EXT.1 Anomaly-Based IPS Functionality**

TSS

The evaluator shall verify that the TSS describes the composition, construction, and application of baselines or anomaly-based attributes specified in IPS_ABD_EXT.1.1. The evaluator shall verify that the TSS provides a description of how baselines are defined and implemented by the TOE, or a description of how anomaly-based rules are defined and configured by the administrator. If 'frequency' is selected in IPS_ABD_EXT.1.1, the TSS shall include an explanation of how frequencies can be defined on the TOE. If 'thresholds' is selected in IPS_ABD_EXT.1.1, the TSS shall include an explanation of how the thresholds can be defined on the TOE. The evaluator shall verify that each baseline or anomaly-based rule can be associated with a reaction specified in IPS_ABD_EXT.1.3. The evaluator shall verify that the TSS identifies all interface types capable of applying baseline or anomaly-based rules and explains how they are associated with distinct network interfaces. Where interfaces can be grouped into a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used) they can be treated collectively as a distinct network interface. (Ref. [6]).

Section 7.9 of the ST (Ref. [13]) explains that administrators can define signatures for anomalous traffic in terms of throughput (bits per second), time of the day for defined source/destination address and source/destination port, frequency of traffic patterns and thresholds of traffic patterns.

Anomaly signatures based on time of day characteristics are implemented by configuring schedulers using the Junos command 'set schedulers' and attaching them to firewall policies, which in turn specify the target traffic in terms of IP addresses and port numbers as well as the action to be perform on signature triggering (allow or block/drop traffic).

Anomaly signatures based on throughput characteristics are implemented by configuring policers with a bandwidth limit and the desired signature action (discard or forward), using the Junos command 'set firewall policer', and attaching it to any interface with the Junos command 'set interfaces'. Traffic exceeding the specified throughput limit is dropped when the policer is configured to discard traffic. A policer can be applied to specific inbound or outbound IP packets in a Layer 3 traffic flow at a logical interface by using a stateless firewall filter. If an input firewall filter is configured on the same logical interface as a policer, the policer is executed first. If an output firewall filter is configured on the same logical interface as a policer, the firewall filter is executed first.

The TOE is capable of inspecting all traffic passing through the TOE's Ethernet interfaces (inline mode). Ethernet interfaces can be assigned to Zones on which firewall and IDP policies are predicated.

Guidance Documentation

The evaluator shall verify that the operational guidance provides instructions to manually create baselines or anomaly-based rules according to the selections made in IPS_ABD_EXT.1.1. Note that dynamic “profiling” of a network to establish a baseline is outside the scope of this PP.

The evaluator shall verify that the operational guidance provides instructions to associate reactions specified in IPS_ABD_EXT.1.3 with baselines or anomaly-based rules. The evaluator shall verify that the operational guidance provides instructions to associate the different policies with distinct network interfaces.

(Ref. [6]).

The evaluator examined the Intrusion Detection and Prevention User Guide (Ref. [21]) that is referenced in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18,19,20].

The Security Target (Ref. [13]) has carried out the following selections for IPS_ABD_EXT.1.1 requirement:

- throughput ([bits per second]);
- time of day;
- frequency;
- thresholds;
- [no other methods]

The Intrusion Detection and Prevention User Guide provides information on how to create baselines or anomaly-based rules according to frequency using frequency and thresholds. As per the guide, “Count or threshold value specifies the number of times that the attack object must detect an attack within the specified scope before the device considers the attack object to match the attack. If you bind the attack object to multiple ports and the attack object detects that attack on different ports, each attack on each port is counted as a separate occurrence.”. Also, as per the guide, the time bindings may be used to configure time attributes for the binding custom attack object. According to the guide “Time attributes control how the attack object identifies attacks that repeat for a certain number of times. By configuring the scope and count of an attack, you can detect a sequence of the same attacks over a period of time across sessions.”.

The Evaluated Configuration Guide (Ref. [18,19,20]) has links to “Policer Implementation Overview” and on “Scheduling Security Policies”. These references link to the Juniper website and in-turn addresses how all the devices implement the throughput and time of day selections. These links may be found in the “Configuring a Security Flow Policy in IPsec Protect Mode” section of the Evaluated Configuration Guide (Ref. [18,19,20]).

The evaluator examined the Intrusion Detection and Prevention User Guide (Ref. [21]) that is referenced in the “IDP Extended Package Configuration Overview” section of the Evaluated Configuration Guide (Ref. [18,19,20].

As per the User Guide, rules are configured with reactions based on the following configuration settings:

```
then {
    action {
        class-of-service {
            dscp-code-point number;
```

```
            forwarding-class forwarding-class;
          }
        (close-client | close-client-and-server | close-server |drop-
connection | drop-packet | ignore-connection |mark-diffserv value | no-
action | recommended);
       }
}
```

This allows for the TOE administrator to configure IDP reactions in line with reactions specified in the Security Target.

For bandwidth policers, the default reaction is to drop traffic. No further configuration is required for this.

Tests

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the instructions in the operational guidance to configure baselines or anomaly-based rules for each attributes specified in IPS_ABD_EXT.1.1. The evaluator shall send traffic that does not match the baseline or matches the anomaly- based rule and verify the TOE applies the configured reaction. This shall be performed for each attribute in IPS_ABD_EXT.1.1.

Test 2: The evaluator shall repeat the test above to ensure that baselines or anomaly- based rules can be defined for each distinct network interface type supported by the TOE. (Ref. [12])

Test 1: Configure the TOE to detect anomalies based on the following attributes: traffic throughput (bits per second), time of the day for defined source/destination address and source/destination port, frequency of traffic patterns and thresholds of traffic patterns. For each of these configurations, send network traffic to trigger the defined signatures. The evaluator verified that the TOE detects the traffic and reacts as configured.

Test 2: Not applicable as only one distinct network interface type is supported.

6 Evaluation Activities for SARs

This Section covers the evaluation activities for the Security Assurance Requirements (SARs) included in NDcPP v2.2e.

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 the NDcPP 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. [18][19][20]) and ST (Ref. [13]), 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 audit server interface, which communicates over SSH; and

the HA Control link, which allows two instances of the TOE configured in Cluster Mode to communicate with one another.

The purpose of these interfaces is clear from the ST and guidance documentation. Furthermore, the guidance documentation contains detailed instructions on how to administer the TOE via the CLI, configure and export data to the audit/syslog server, and set up the Control link to enable Cluster Mode.

The guidance documentation (Ref. [18][19][20]) provide a thorough description of how to use the CLI, the audit server interface and the HA Control link, and all the commands that are available, including a description of all the parameters.

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 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. [18][19][20]) 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]).

The Evaluated Configuration Guide (Ref. [18][19][20]) requires the TOE to operate in FIPS mode and covers all Operational Environment that the product supports.

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. [2][3]) 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. [18][19][20]) 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) The “Downloading Software Packages from Juniper Networks (FIPS Mode)” section of the of the Evaluated Configuration Guide (Ref. [18][19][20]) provides instructions on how an update can be obtained. The “Installing Junos Software Packages (FIPS Mode)” section outlines the steps on how the update can be installed.
c) The Evaluated Configuration Guide (Ref. [18][19][20]) 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 Evaluated Configuration Guide (Ref. [18][19][20]) address this in detail.

- OE.NO_GENERAL_PURPOSE is met implicitly as the TOE is a dedicated firewall platform and does not permit the installation of any general-purpose software.
- OE.TRUSTED_ADMIN is met if the administrator configures the TOE according to the Evaluated Configuration Guide (Ref. [18][19][20]). 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 is dependent on the administrator behaviour. The Evaluated Configuration Guide (Ref. [18][19][20]) provide details on how the administrator can provide updates to the TOE.
- OE.RESIDUAL_INFORMATION is depended on the administrator behaviour. The Evaluated Configuration Guide (Ref. [18][19][20]) provides details on how the administrator can zeroize the TOE.
- OE.ADMIN_CREDENTIALS_SECURE is dependent on the configuration of the TOE according to the Evaluated Configuration Guide, the behaviour of the administrator and the overall security of the environment.
- OE.CONNECTIONS is met by the collective procedures that are provided in the guidance documentation (Ref. [18][19][20]).

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. [18][19][20]) 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. [18][19][20]) allowed for the security of the TSF to be managed as a component of the operational environment.

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.

a) The “Configuring Administrative Credentials and Privileges’ section of the Evaluated Configuration Guide (Ref. [18][19][20]) provides instructions on configuring protected administrative capability.

b) The TOE does not have default values for passwords. However, as per the applicable Evaluated Configuration Guide (Ref. [18][19][20]) the TOE requires the root password to be configured before any changes are made to the configuration.

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. In all cases, the output indicates that the TOE version on all platforms are running Junos 22.2R1. This output is consistent with the expected version.

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. [13]) to determine whether it includes the TOE and its associated evaluation evidence.

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

[ECG1] Junos OS Common Criteria Guide for SRX320, SRX340, SRX345, SRX345-Dual-AC and SRX380 Devices, Release 22.2R1, 27 August 2022

[ECG2] Junos OS Common Criteria Guide for SRX1500, SRX4100, SRX4200, and SRX4600 Devices, Release 22.2R1, 27 August 2022

[ECG3] Junos OS Common Criteria Guide for SRX5400, SRX5600, and SRX5800 Devices, Release 22.2R1, 27 August 2022

The evaluators examined the Configuration List provided in Section 1.6.2 of the Security Target (Ref. [1]) to determine whether each configuration item is uniquely identified.

Each item in the configuration list is assigned a unique name, and publication date. This allows for unique identification of each document and the TOE itself.

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 evaluator confirms that the test configuration as described in the Test Reports (Refs. [14][15][16][17]) is consistent with the configuration under evaluation as per the ST (Ref. [13]), and CC guidance documentation (Ref. [18][19][20]). All tests in the Test report start with the TOE configured as per CC guidance, which is in turn consistent with the ST.

The test plan executed by the evaluators (Refs. [7][8][9][10]) requires the evaluators to configure the TOE as per the CC guidance (Ref.[14][15][16]). The evaluators have followed these instructions as per the test plan and were able to confirm that the instructions leave the TOE configured in the expected state.

The evaluators developed a test plan with all the tests specified in the Supporting Documents for NDcPP v2.2E (Ref. [2]), FW_MOD_SD (Ref. [4]), MOD_VPNGW (Ref. [12]) and MOD_IPS (Ref. [13]) updated with all the technical decisions listed in the ST (Ref. [11]). This test plan was executed in its entirety and documented in the Test Reports (Refs. [7][8][9][10]).

The Test Reports (Refs. [7][8][9][10]) document the testing performed by the evaluators and provides sufficient detail on the test environment, step-by-step test cases, expected and observed results, such that the test plan can be easily repeatable.

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 evaluation activities for this work unit are specified in the following references:

- Appendix A of the NDcPP SD (Ref.[6]),

- Section 6 of the MOD_VPNGW SD (Ref. [8]), and
- Appendix A of the MOD_FWcPP SD (Ref. [10]).
- Section 7 of the MOD_IPS SD (Ref. [15]).

The evaluation activities follow the flaw hypothesis methodology. Accordingly, four types of flaw hypotheses have been considered.

Type 1 Hypotheses – Public-Vulnerability-Based

The evaluators performed a search on the sources listed in Section A.4 of the NDcPP SD (Ref. [6]) to determine a list of potential flaw hypotheses that are more recent than the publication date of the cPP, i.e. December 2019, and those that are specific to the TOE and its components.

The search terms used were as follows:

- Terms related to the device type of the TOE:
  - “firewall”
  - “router”
- Protocols required to be checked:
  - “TCP”
  - “UDP”
  - “IPv4”
  - “IPv6”
- Other protocols supported by the TOE
  - “SSH”
  - “IPsec”
  - “IKE”
- TOE’s name and components:
  - Juniper SRX
  - “Junos 22.2R1”
  - “SRX300”
  - “SRX320”
  - “SRX340”
  - “SRX345”
  - “SRX345-DUAL-AC”
  - “SRX1500”
  - “SRX4100”
  - “SRX4200”
  - “SRX4600”
  - “SRX5000 Series”
  - “SRX5400”
  - “SRX5600”
  - “SRX5800”
  - “OpenSSL 1.0.2zd”
  - “OpenSSH 7.5”
  - "FreeBSD 6”

- "FreeBSD 12"

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 and MOD_FWcPP 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.

Type 4 Hypotheses – Tool-Generated

As per NDcPP SD, MOD_VPNGW SD and MOD_FWcPP SD, 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.

In total, the following vulnerability tests were created to address the above vulnerabilities:

- **VUL-NDcPP-001**, The aim of this test is to determine whether the TOE is affected by undefined protocol codes in packets. This test focuses on the consumer ports. PP-Requirement
- **VUL-NDcPP-002**, The aim of this test is to determine whether the TOE is affected by malformed IPv4, IPv6, ICMPv4 and ICMPv6 packets. This test focuses on the consumer ports. PP-Requirement
- **VUL-NDcPP-003**, The aim of this test is to determine whether the TOE is affected by undefined protocol codes in packets sent to the HA control ports.
  NOTE: This test is only applicable to High Availability configurations. PP-Requirement
- **VUL-NDcPP-004**, The aim of this test is to determine whether the TOE is affected by undefined protocol codes in packets sent to the HA fabric ports.
  NOTE: This test is only applicable to High Availability configurations. PP-Requirement
- **VUL-NDcPP-005**, The aim of this test is to determine whether the TOE's control port is affected by malformed IPv4, IPv6, ICMPv4 and ICMPv6 packets.
  NOTE: This test is only applicable to High Availability configurations. PP-Requirement
- **VUL-NDcPP-006**, The aim of this test is to determine whether the TOE's fabric port is affected by malformed IPv4, IPv6, ICMPv4 and ICMPv6 packets.
  NOTE: This test is only applicable to High Availability configurations. PP-Requirement

- **VUL-NDcPP-007**, This test aims to using Nessus to identify all potential vulnerabilities that may affect the TOE. If Nessus finds a potential issue, further manual analysis is required. Evaluation-Team-Generated
- **VUL-NDcPP-008**, This test aims to ensuring that all CSP data types communicated between the two cluster nodes is sent protected over the control link IPSEC connection.
  NOTE: This test is only applicable to High Availability configurations. Evaluation-Team-Generated
- **VUL-NDcPP-009**, This test aims to check if console sessions established on the two HA nodes are independent. It means that an established session on one node is not shared with the other node.
  NOTE: This test is only applicable to High Availability configurations. Evaluation-Team-Generated
- **VUL-NDcPP-010**, This test aims to check if failover in a cluster may cause any security issues.
  NOTE: This test is only applicable to High Availability configurations. Evaluation-Team-Generated
- **VUL-NDcPP-011**, 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. Evaluation-Team-Generated
- **VUL-NDcPP-012**, This test aims to check if the TOE is affected by the issues described at CVE-2020-7451.
  The evaluator shall establish a half-open TCP connection from Alice to the port 22 on the TOE, and force the TOE to re-transmit its [SYN, ACK] packet. Finally, Alice will send a reset packet. The evaluator shall confirm that in every IPv6 packet from the TOE, the traffic class value is set to zero. Public-Vulnerability-Based
- **VUL-NDcPP-013**, This test aims to check if the TOE is affected by the issues described at CVE-2020-7469.
  The evaluator shall reply the TOE'e ICMPv6 EchoRequest frames by sending all possible ICMPv6 error messages to the TOE. The evaluator confirms that those error messages do not make any process crash on the TOE.Public-Vulnerability-Based

The methodology, set up requirements, conditions, expected results, actual results and testing instructions have been recorded in the test reports for this evaluation (Ref. [15][16][14][17]).

The penetration testing conducted discovered no exploitable 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