Assurance Activity Report

ViaSat, Inc.
Viasat Secure VPN v1.1.7

VID 11405

UL1341589-AAR Rev 1.2
December 19, 2023

Evaluated by:

UL Verification Services Inc.
709 Fiero Lane, Suite 25
San Luis Obispo, CA 93401

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

TOE Evaluation Sponsor and Developer

Viasat, Inc.
6155 El Camino Real
Carlsbad, CA 92009
USA

ST Author:
UL Verification Services Inc.
709 Fiero Lane, Suite 25
San Luis Obispo, CA 93401

Evaluation Personnel:
Oleg Andrianov
Brad Mitchell
Dylan Lyman

Applicable Common Criteria Version
CC Version 3.1 R5, April 2017

Common Evaluation Methodology Version
CEM Version 3.1 R5, April 2017

Applicable Common Criteria Protection Profiles
collaborative Protection Profile for Network Devices
Version 2.2e, March 23, 2020

PP-Module for Virtual Private Network (VPN) Gateways
Version 1.2, March 31, 2022

Table of Contents

1 Overview

This document presents evaluation results of the ViaSat Secure VPN v1.1.7 against the collaborative Protection Profile for Network Devices, Version 2.2e, March 23, 2020 [PP] and PP-Module for Virtual Private Network (VPN) Gateways Version 1.2, March 31, 2022 [MOD]. This document contains a description of the assurance activities and associated results as performed by UL, an accredited Common Criteria Testing Laboratory. This Evaluation was conducted with the oversight and guidance provided by the National Information Assurance Partnership and its contributors.

1.1 Test Equivalency

The [ST] claims only one platform; therefore, testing only occurred on the one claimed platform and no equivalency argument was necessary.

1.2 Test Environment

The test environment used by the CCTL during the course of testing is briefly summarized below and conforms to the expected use-case of the TOE (Network Device, VPN Gateway).

The evaluation team performed the independent testing activities to confirm the TOE operates to the TOE security functional requirements as specified in the [ST] for a product claiming conformance to [PP]. The evaluation team devised a Test Plan based on the Testing Assurance Activities specified in [PP]. The Test Plan described how each test activity was to be performed. The evaluation team executed the tests specified in the Test Plan and documented the results in the Evaluation Technical Report. The evaluation team consisted of Oleg Andrianov, Dylan Lyman, and Brad Mitchell from the CCTL.

The test laboratory was configured by UL and physically located at the UL San Luis Obispo facility in an access-controlled environment. Testing was conducted between March 24, 2022 and September 22, 2023.

[ST] defines only one TOE model, “Viasat Secure VPN v1.1.7”. This is the model that was tested.

The TOE was tested on the platform that is describe in the [ST]:

- Hardware Platform
  - Dell XPS 8940 Server Hardware
    - Processor: 11th Gen Intel Core i5-1140 @ 2.6Ghz
    - RAM: 16Gb Memory
    - Hard drive: 1TB mechanical (spinning) SATA drive
- Software Platform
    - Windows 10 Pro 22H2
    - Microsoft (MS) Hyper-V Type 1 hypervisor Virtual Machine Manager (VMM)

1.1 Test network overview

The CC test bed consists physically of:

QTY	Device	Purpose
1	HP ProLiant DL360 Gen 9 Blade Server	ESXi 6.7 Server w/8 LAN ports
1	Aruba 2920-24G 24-port layer 3 aware Ethernet switch	Infrastructure Routing Switch
1	Dell PowerEdge 840 Desktop Workstation	Management & Control console
1	Dell XPS 8940 • Processor: 11th Gen Intel Core i5-1140 @ 2.6Ghz • RAM: 16Gb Memory • Hard drive: 1TB mechanical (spinning) SATA drive	Hyper-V Virtualization system hosting the TOE, and other VMs, acting as a CRL server and remote audit server.

	Windows 10 with Microsoft Hyper-V	

The test lab uses several separate test environments (referred to as strings) via the extensive use of virtualization and network access control. Access to the environments is through a dedicated network and VPN.

The management console provides the access point for the validator or tester either through physical access in the lab or via SSH or remote desktop. These provide the ability to subsequently connect to the various devices in each string without exposing the entire environment to the external network. Each workstation has one network interface on the UL test network and a second interface connected to the Aruba 2920 switch.

Figure 1 shows the physical layout of test environment.

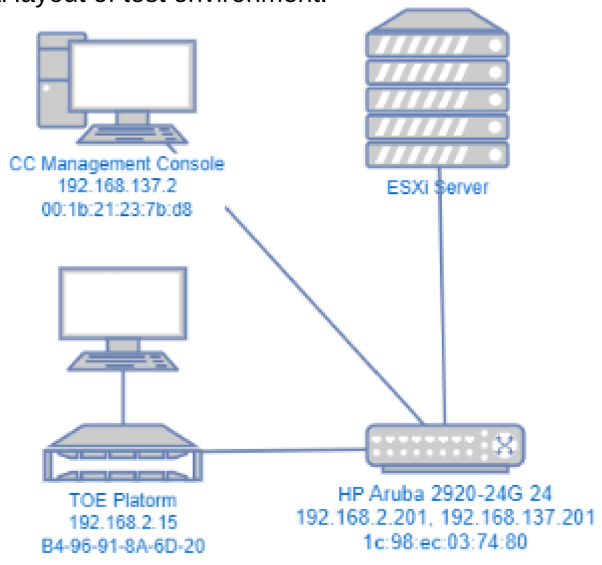

Figure 1 - Physical Setup

Logical layout differs for general testing and testing on NAT support.

1.2 Test layout for general testing

1.2.1 Overall layout

The test string used for this evaluation consists of the following virtual and physical networks:

Network name	Description	Addressing scheme
Management	Network used by evaluator to access all test machines, except the TOE. No test traffic on this network.	192.168.2.0/24
Red	Network representing private segment protected by the TOE, connected to the TOE RED (Plaintext) interface.	10.1.2.0/24
TOE MGMN	Network segment connected to the TOE management network interface.	10.3.2.0/24
Public	Network representing WAN, connected to the TOE BLACK (ciphertext) interface.	10.2.2.0/24
Target	Network segment located behind VPN Gateway. Acts as a destination for VPN-	172.16.2.0/24

Network name	Description	Addressing scheme
	encrypted connections from Red Network.	

TOE platform Hyper-V Hypervisor Virtual Switch was configured to contain the following Virtual Switches:

1. PT(RED) Virtual Switch (private switch)
2. TOE MGMNT Virtual Switch (private switch)
3. CT(BLACK) Virtual Switch (external network with monopolistic usage)
4. Management Virtual Switch (external network wish shared usage)

Figure 2 – Test Network Configuration shows the logical configuration used for the ATE.

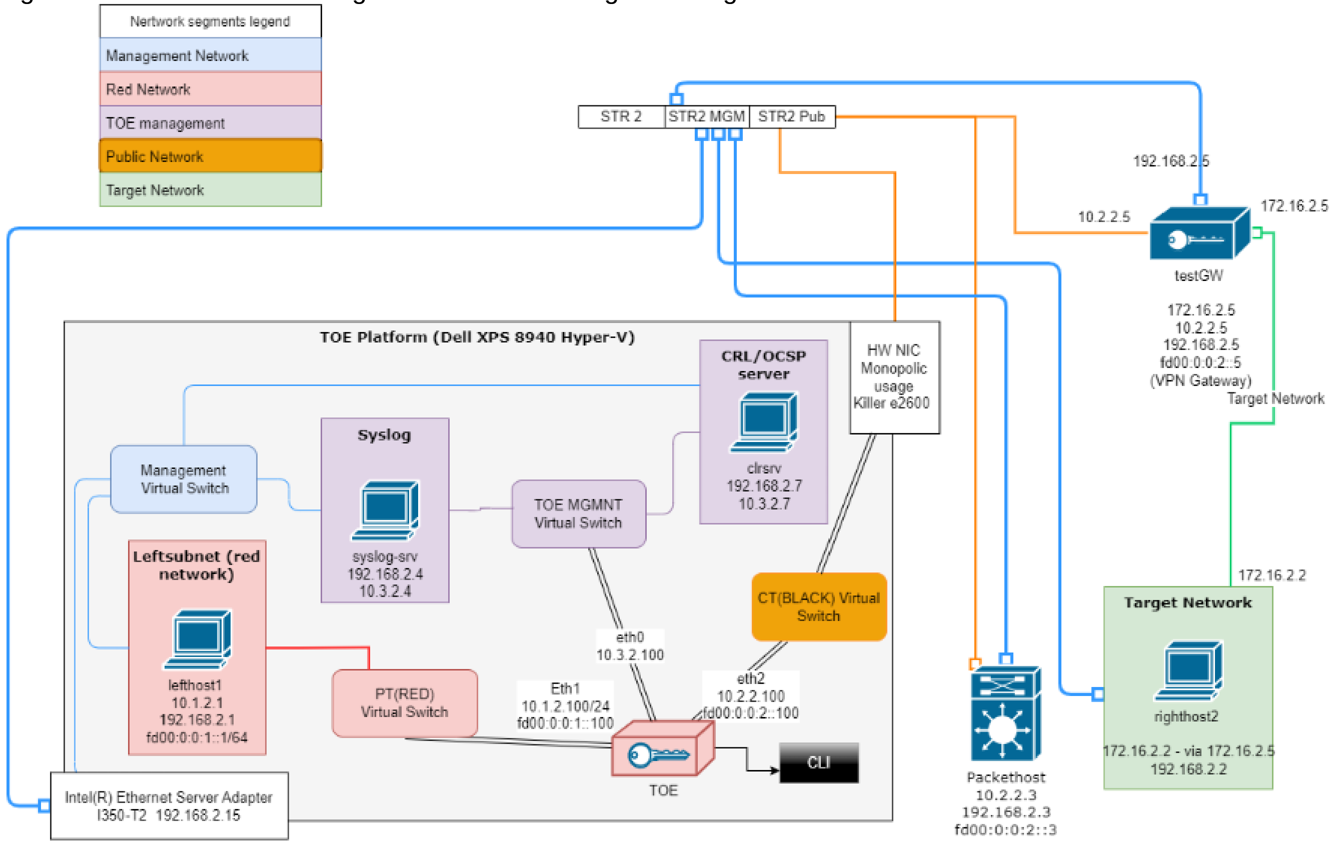

Figure 2 – Test Network Configuration

Test environment contains the following test machines and devices:

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
ESXI6	Hosting for test VMs listed below.	Management VLAN: 192.168.137.200 (28:80:23:b4:34:e3) + Virtual VMs as shown below	ANY	N/A	Esxi 6.7	N/A	NTP from time.nist.gov
ARUBA	Network switch	24 Networking ports, transparent for test networks.	ANY	N/A	Aruba OS 16.10	N/A	NTP from cclabslo01
cclabslo01	Lab access via SSH tunnel, NTP server	Management VLAN: 192.168.137.2/24, MAC: 00:1b:21:23:7b:d8	SSH, NTP	Evaluator	Ubuntu 18.04.6	OpenSSH_7.6p1, chrony 3.2	NTP from time.nist.gov
Evaluator machine	Lab access, evidence capture and extraction	Not directly connected to the lab network	SSH, TLS	Evaluator	Windows 10	See Section 1.6.2	UL Domain
Crlsrv	VM on a Hyper-V. CRL distribution point	eth0: • link/ether 00:15:5d:09:70:03 brd ff:ff:ff:ff:ff:ff • inet 192.168.2.7/24 brd 192.168.2.255 scope global eth0 • inet6 fe80::215:5dff:fe09:7003/64 scope link eth1: • link/ether 00:15:5d:09:70:04 brd ff:ff:ff:ff:ff:ff • inet 10.3.2.7/24 brd	HTTP, SSH	ulvs	Ubuntu 20.04.4 LTS	See Section 1.6	NTP sync with cclabslo01

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
		10.3.2.255 scope global eth1 • inet6 fe80::215:5dff:fe09:7004/64 scope link					
Syslog-srv	VM on a Hyper-V. Remote audit server	eth0: • link/ether 00:15:5d:09:70:07 brd ff:ff:ff:ff:ff:ff • inet 192.168.2.4/24 brd 192.168.2.255 • inet6 fe80::215:5dff:fe09:7007/64 scope link eth1: • link/ether 00:15:5d:09:70:08 brd ff:ff:ff:ff:ff:ff • inet 10.3.2.4/24 brd 10.3.2.255 scope global eth1 • inet6 fe80::215:5dff:fe09:7008/64 scope link	TLS, SSH	ulvs	Ubuntu 20.04.4 LTS	See Section 1.6	NTP sync with cclabslo01
Lefthost1	VM on a Hyper-V. VPN Tunnel user, machine on the left side of the VPN tunnel	eth0: • link/ether 00:15:5d:09:70:05 brd ff:ff:ff:ff:ff:ff	SSH, TCP, UDP, all	ulvs	Ubuntu 20.04.4 LTS	See Section 1.6	NTP sync with cclabslo01

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
		• inet 192.168.2.1/24 brd 192.168.2.255 scope global eth0 • inet6 fe80::215:5dff:fe09:7005/64 scope link eth1: • link/ether 00:15:5d:09:70:0f brd ff:ff:ff:ff:ff:ff • inet 10.1.2.1/24 brd 10.1.2.255 scope global eth1 • inet6 fd00:0:0:1::1/64 scope global • inet6 fe80::215:5dff:fe09:7006/64 scope link					
Righthost2	VM on an ESXi. VPN Tunnel user, machine on the other side of the VPN tunnel	ens224: • link/ether 00:0c:29:b6:12:de brd ff:ff:ff:ff:ff:ff • inet 172.16.2.2/24 brd 172.16.2.255 scope global ens224 • inet6 fe80::20c:29ff:feb6:12de	TCP,UDP, SSH, all	ulvs	Ubuntu 20.04.4 LTS	See Section 1.6	NTP sync with cclabslo01

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
		/64 scope link ens256: • link/ether 00:0c:29:b6:12:e8 brd ff:ff:ff:ff:ff:ff • inet 192.168.2.2/24 brd 192.168.2.255 scope global ens256 • inet6 fe80::20c:29ff:feb6:12e8 /64 scope link					
testGW	VM on a ESXi. VPN Gateway peer	ens192: • link/ether 00:0c:29:49:0b:93 brd ff:ff:ff:ff:ff:ff • inet 10.2.2.5/24 brd 10.2.2.255 scope global ens192 • inet6 fd00:0:0:3::5/7 scope global ens224: • link/ether 00:0c:29:49:0b:9d brd ff:ff:ff:ff:ff:ff • inet 172.16.2.5/24 brd 172.16.2.255 scope global ens224 • inet6 fd00:0:0:4::5/64	SSH, IPSEC	ulvs	Ubuntu 20.04.4 LTS	See Section 1.6	NTP sync with cclabslo01

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
		scope global ens256: • link/ether 00:0c:29:49:0b:a7 brd ff:ff:ff:ff:ff:ff • inet 192.168.2.5/24 brd 192.168.2.255 scope global ens256					
Packethost	VM on a ESXi. Packet capture for Ipsec traffic.	Ens192: • link/ether 00:0c:29:ae:8c:8e brd ff:ff:ff:ff:ff:ff • inet 10.2.2.3/24 brd 10.2.2.255 scope global ens192 • inet6 fd00:0:0:2::3/64 scope global ens256: • link/ether 00:0c:29:ae:8c:a2 brd ff:ff:ff:ff:ff:ff • inet 192.168.2.3/24 brd 192.168.2.255 scope global ens256	all	ulvs	Ubuntu 20.04.4 LTS	See Section 1.6	NTP sync with cclabslo01
TOE (instance 1)	TOE under evaluation	Public segment: • MAC 00:15:5d:09:70:06 • Ipv4 10.2.2.100/24	ALL, TLS, Ipsec	admin	TOE v1.0.2	-	Manual, Hyper-V

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
		TOE management: • 00:15:5d:09:70:04 • Ipv4 10.3.2.100/24 Plaintext Segment: • MAC: 00:15:5d:09:70:05 • Ipv6 ff02::1:ff00:100 • Ipv4 10.1.2.100/24					
TOE (instance 2)	TOE under evaluation.	Public segment: • MAC: 00:15:5d:09:70:0C • Ipv4 10.2.2.101/24 TOE management: • MAC: 00:15:5d:09:70:0A • Ipv4 10.3.2.101/24 Plaintext Segment: • MAC: 00:15:5d:09:70:0B • Ipv4 10.1.2.101/24	ALL, TLS, Ipsec	admin	TOE v1.0.2	-	Manual, Hyper-V

1.3 Test Layout for FPT_ITC.1.3 Test 4

This Layout is similar to the general testing layout, but connection to syslog server was reconfigured to be routed through Physical network interfaces, connected with a physical cord that can be removed to perform a network interruption.

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
Syslog-	VM on a Hyper-V.	eth0:	TLS, SSH	ulvs	Ubuntu	See Section 1.6	NTP sync with

Device Name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
srv6	Remote audit server	• link/ether 00:15:5d:09:70:10 brd ff:ff:ff:ff:ff:ff • inet 192.168.2.6/24 brd 192.168.2.255 eth1: • link/ether 00:15:5d:09:70:11 brd ff:ff:ff:ff:ff:ff • inet 10.3.2.6/24 brd 10.3.2.255 scope global eth1			20.04.4 LTS		cclabslo01

1.4 Test Layout for FCS_IPSEC_EXT.1.5 Test 2.

This Layout is similar to the general testing layout, but connection to TestGW was reconfigured to create a NAT environment.

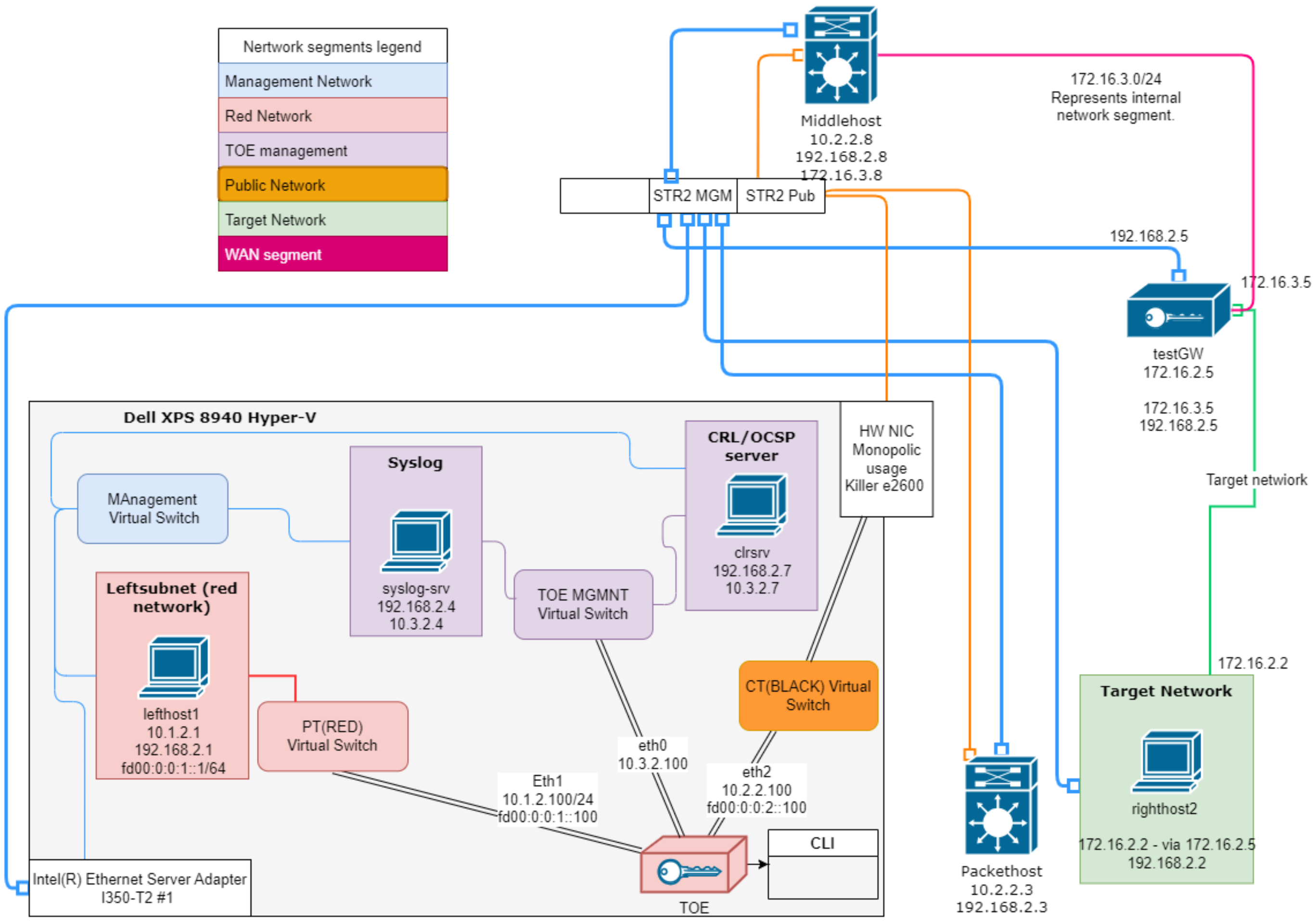

Figure 3 - Test Layout (NAT)

Device name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
Middlehost8	VM on a ESXi.	Ens192:	IPv4, SSH	ulvs	Ubuntu	See Section 1.6	NTP sync

Device name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
	NAT server	• MAC: 00:0c:29:7f:fb:7a • IPv4: 10.2.2.8 • IPv6: fc00:0:0:2::8 Ens193: • MAC: 00:0c:29:7f:fb:98 • IPv4: 10.3.2.8 Ens256: • MAC: 00:0c:29:7f:fb:8e • IPv4: 192.168.2.8 Ens224 • MAC 00:0c:29:7f:fb:84 • IPv4 172.16.3.8/24 • IPv6 fc00:0:0:3::8/64			20.04.4 LTS		with cclabslo01
testGW	VM on a ESXi. VPN Gateway peer	ens192: • link/ether 00:0c:29:49:0b:93 • inet 10.2.2.5/24 brd 10.2.2.255 scope global ens192 • inet6 fd00:0:0:3::5/7 scope global ens224: • link/ether 00:0c:29:49:0b:9d • inet 172.16.2.5/24 brd 172.16.2.255 scope global ens224	SSH, IPSEC	ulvs	Ubuntu 20.04.4 LTS	See Section 1.6	NTP sync with cclabslo01

Device name	Functions/ test role	Network, IP address, Subnet, MAC	Protocols used	Account (if specific)	OS	Test tools, versions	Time stamp source
		• inet6 fd00:0:0:4::5/64 scope global ens256: • link/ether 00:0c:29:49:0b:a7 • inet 192.168.2.5/24 brd 192.168.2.255 scope global ens256 ens193: • MAC: 00:0c:29:49:0b:b1 • IPv4: 172.16.3.5/24 • IPv6: fd00:0:0:3::5/7					

1.5 Connections and Interfaces

The Evaluator used Postman to interact with the TOE using RestAPI interface. Port forwarding was set up so that https calls to 127.0.0.1:7443 were routed to the TOE management interface on 10.3.2.100 port 443, so throughout this test report some screenshots show RestAPI calls to the TOE as placed to 127.0.0.1:7443.

The Hyper-V local console for the TOE was used as the TOE local console as per [AGD] Section 4.1.

1.6 Test Tools and Equipment

1.6.1 Test Environment Tools:

1. tcpdump version 4.9.3
2. libpcap version 1.9.1 (with TPACKET_V3)
3. OpenSSL 1.1.1f 31 Mar 2020
4. Linux strongSwan U5.8.2/K5.4.0-109-generic
5. OpenBSD netcat (Debian patchlevel 1.206-1ubuntu1)
6. rsyslogd 8.2001.0 (aka 2020.01)
7. Nmap 7.80 (https://nmap.org)
8. In-house tools:

1) UL-IPsec test tools (commit 8a3682e2483f1cc4025b801381c877733a064faa)
2) TLS test tool util v.1 (based on s2n) (commit e5bd15cd12b472db2de82e7ff4268e1053779e56)
3) TLS PytestTool (commit e5bd15cd12b472db2de82e7ff4268e1053779e56)
4) Traffic generation scripts py3Scripts (commit 92205041ac40cd11c5983c36f6e24e5f6c08acb9)
5) OCSP/CRL server (commit

9. Python 3.8
10. chrony version 4.0

1.6.2 Evaluator Machine

1. OpenSSL Toolkit (for Windows 1.1.1e 2020-03-08 (Certificate parsing and creation)
2. XCA portable 2.4.0 (For Certificate parsing and creation)
3. Postman v10.6.0 (For TOE management using Rest API)
4. Bitvise SSH Client 9.26 (For SSH connection to the lab machines and to the TOE)
5. HxD Hex Editor (Version 2.5.0.0 (x86-64))

2 SFR Assurance Activities and Results

2.1 FAU_GEN.1 Audit Data Generation

The main reasons for collecting audit information are to detect and identify error conditions, security violations, etc. and to provide sufficient information to the Security Administrator to resolve the issue. The audit information to be collected according to FAU_GEN.1, and the failure conditions identified in tables 2, 4, and 5 need to enable the Security Administrator at least to detect and identify the problem and provide at least basic information to resolve the issue. Also for this level of detail, the other FAU requirements apply, in particular the need for local and remote storage of audit information according to FAU_STG_EXT.1.

The level of detail that needs to be provided to the Security Administrator to actually resolve an issue usually depends on the complexity of the underlying use case. It is expected that a product provides additional levels of auditing to support resolution of error conditions, security violations, etc. beyond the level required by FAU_GEN.1, but it should also be clear that a high level of granularity cannot be maintained on most systems by default due to the high number of audit events that would be generated in such a configuration. It is expected that the TOE will be capable of auditing sufficient information to meet the requirements of FAU_GEN.1. This may include audits that are generated only when configured if the TOE configuration can be modified without taking the TOE out of the evaluated configuration.

The issue described above explicitly refers to the use of X.509 certificates. In case a certificate-based authentication fails, an error message telling the Security Administrator that ‘something is wrong with the certificate’ shall not be considered as sufficient information about the ‘reason for failure’ as a basic information to resolve the issue. The log message will inform the Security Administrator of at least the following:

- ‘Trust issue’ with the certificate, e.g. due to failed path validation
- Use of an ‘expired certificate’
- Absence of basicConstraints extension
- CA flag not set for a certificate presented as a CA
- Signature validation failure for any certificate in the certificate path; failure to establish revocation status; revoked certificate

As such for audit information related to the use of X.509 certificates that it uniquely identifies the certificate that could not be successfully verified. For example, identification of a certificate could include Key Subject and Key ID, where key subject is an identifier contained in the CN or SAN and where Key ID is a certificate's serial number and issuer name or subject key identifier (SKI) and authority key identifier (AKI).

In general, when using open source libraries like OpenSSL, passing on error messages from such libraries to the Security Administrator is regarded as good practice.

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.

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

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 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.

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.

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

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

2.1.1 TSS

The TOE is not a distributed TOE.

[ST] Section 7.1.1 describes that for key creation the TOE logs the service name for which the key is created and for key deletion, the TOE logs service name and key file type (‘key’, ‘csr’ or ‘cert’). It states that only one private key/csr is allowed for each service, so this level of detail is sufficient to identify the keys on which the operation is performed.

2.1.2 Guidance Documentation

FAU_GEN.1.1		FAU_GEN.1.2					
SFR	**Auditable Events**	**Date/Time of Event** *("Time") or ("Start Time" and "End Time") or ("Timestamp") field*	**Type of Event** *("Command" and "subcommand") or ("Event") or ("Service") fields*	**Subject Identity** *("User") or ("Hostname") field*	**Outcome (Success or Failure)**	**Additional Audit Record Contents**	**Section Reference**
FAU_GEN.1	Start-up and shut-down of the audit functions;	X	X	X	X	None.	[AGD] Table 5-2 #24-25
FAU_GEN.1	Administrative login and logout (name of user account shall be logged if individual user accounts are required for Administrators).	X	X	X	X	None.	[AGD] Table 5-2 #2
FAU_GEN.1	Changes to TSF data related to configuration changes (in addition to the information that a change occurred it shall be logged what has been changed).	X	X	X	X	Addition to the information that a change occurred it shall be logged what has been changed.	[AGD] Table 5-2 #5
FAU_GEN.1	Generating/import of, changing, or deleting of cryptographic keys (in addition to the action itself a unique key name or key reference shall be logged).	X	X	X	X	None.	[AGD] Table 5-2 #15
FAU_GEN.1	Resetting passwords (name of related user account shall be logged).					None.	N/A as Not supported by the TOE
FCS_HTTPS_EXT.1 (selection-based)	Failure to establish a HTTPS session	✓	✓	✓	✓	Reason for failure	[AGD] Table 5.2 #17
FCS_IPSEC_EXT.1 (selection	Failure to establish an IPsec SA.	✓	✓	✓	✓	Reason for failure	[AGD] Table 5.2 #18

FAU_GEN.1.1		FAU_GEN.1.2					
SFR	**Auditable Events**	**Date/Time of Event** *(“Time”) or (“Start Time” and “End Time”) or (“Timestamp”) field*	**Type of Event** *(“Command” and “subcommand”) or (“Event”) or (“Service”) fields*	**Subject Identity** *(“User”) or (“Hostname”) field*	**Outcome (Success or Failure)**	**Additional Audit Record Contents**	**Section Reference**
-based)							
FCS_TLSC_EXT.1 (selection-based)	Failure to establish a TLS session	✓	✓	✓	✓	Reason for failure	[AGD] Table 5.2 #19
FCS_TLSS_EXT.1 (selection-based)	Failure to establish a TLS session	✓	✓	✓	✓	Reason for failure	[AGD] Table 5.2 #16
FCS_TLSS_EXT.2 (selection-based)	Failure to authenticate the client	✓	✓	✓	✓	Reason for failure	[AGD] Table 5.2 #16
FIA_AFL.1	Unsuccessful login attempts limit is met or exceeded	✓	✓	✓	✓	Origin of the attempt (e.g., IP address):	[AGD] Table 5-2 #1
FIA_UIA_EXT.1	All use of identification and authentication mechanism	✓	✓	✓	✓	Origin of the attempt (e.g., IP address):	[AGD] Table 5-2 #2
FIA_UAU_EXT.2	All use of identification and authentication mechanism	✓	✓	✓	✓	Origin of the attempt (e.g., IP address):	[AGD] Table 5-2 #2
FIA_X509_EXT.1/Rev (selection-based)	Unsuccessful attempt to validate a certificate Any addition, replacement or removal of trust anchors in the TOE's trust store	✓	✓	✓	✓	Reason for failure of certificate validation Identification of certificates added, replaced, or removed as a trust anchor in the TOE's trust store.	[AGD] Table 5-2 #15
FMT_MOF.1/Functions	None.					None.	

FMT_MOF.1/ManualUpdate	Any attempt to initiate a manual update	✓	✓	✓	✓	None	[AGD] Table 5-2 #4
FMT_MTD.1/CoreData	None.					None	
FMT_MTD.1/CryptoKeys	None.					None	
FMT_SMF.1	All management activities of TSF data.	✓	✓	✓	✓	None	[AGD] Table 5-2 #5
FPT_ITT.1 (optional)	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions.	✓	✓	✓	✓	Identification of the initiator and target of failed trusted channels establishment attempt.	[AGD] Table 5-2 #12
FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure)	✓	✓	✓	✓	None	[AGD] Table 5-2 #6
FPT_TUD_EXT.2 (selection-based)	Failure of update	✓	✓	✓	✓	Reason for failure (including identifier of invalid certificate)	[AGD] Table 5-2 #8
FPT_STM_EXT.1	Discontinuous changes to time - either Administrator actuated or changed via an automated process. (Note that no continuous changes to time need to be logged. See also application note on FPT_STM_EXT.1)	✓	✓	✓	✓	For discontinuous changes to time: the old and new values for the time. Origin of the attempt to change time for success and failure (e.g., IP address)	[AGD] Table 5-2 #7
FTA_SSL_EXT.1 (if "terminate the session" is selected	The termination of a local session by the session locking mechanism	✓	✓	✓	✓	None.	[AGD] Table 5-2 #9

FTA_SSL.3	The termination of a remote session by the session locking mechanism	✓	✓	✓	✓	None	[AGD] Table 5-2 #10
FTA_SSL.4	The termination of an interactive session	✓	✓	✓	✓	None	[AGD] Table 5-2 #11
FTP_ITC.1	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions	✓	✓	✓	✓	Identification of the initiator and target of failed trusted channels establishment attempt.	[AGD] Table 5-2 #12
FTP_TRP.1/Admin	Initiation of the trusted path. Termination of the trusted path. Failure of the trusted path functions.	✓	✓	✓	✓	None	[AGD] Table 5-2 #13

2.1.3 Tests

SFR	Auditable Events	Date/Time Of Event *(“Time”) or (“Start Time” and “End Time”) or (“Timestamp”) field*	Type of Event *(“Command” and “subcommand”) or (“Event”) or (“Service”) fields*	Subject Identity *(“User”) or (“Hostname”) field*	Outcome (Success or Failure)	Additional Audit Record Contents
FAU_GEN.1	Start-up and shut-down of the audit functions;	X	X	X	X	None.
FAU_GEN.1	Administrative login and logout (name of user account shall be logged if individual user accounts are required for Administrators).	X	X	X	X	None.
FAU_GEN.1	Changes to TSF data related to configuration changes (in addition to the information that a change occurred it shall be logged what has been changed).	X	X	X	X	Addition to the information that a change occurred it shall be logged what has been changed.
FAU_GEN.1	Generating/import of, changing, or deleting of cryptographic keys (in	X	X	X	X	None.

SFR	Auditable Events	Date/Time Of Event *(“Time”) or (“Start Time” and “End Time”) or (“Timestamp”) field*	Type of Event *(“Command” and “subcommand”) or (“Event”) or (“Service”) fields*	Subject Identity *(“User”) or (“Hostname”) field*	Outcome (Success or Failure)	Additional Audit Record Contents
	addition to the action itself a unique key name or key reference shall be logged).					
FAU_GEN.1	Resetting passwords (name of related user account shall be logged).					None.
TOE does not support resetting password for user account. The TOE supports only one user account.						
FAU_GEN.1	Events as selected in FAU_GEN.1.1 in the ST: None is selected in the FAU_GEN1.1					None.
FAU_GEN.2	None.					None.
FCS_HTTPS_EXT.1 (selection-based)	Failure to establish a HTTPS Session.	X	X	X	X	Reason for failure.
FCS_IPSEC_EXT.1 (selection-based)	Failure to establish an IPsec SA.	X	X	X	X	Reason for failure.
FCS_IPSEC_EXT.1 (PP Module-based)	Session Establishment with peer	X	X	X	X	Entire packet contents of packets transmitted/ received during session establishment.
FCS_TLSC_EXT.1 (selection-based)	Failure to establish a TLS Session	X	X	X	X	Reason for failure.
FCS_TLSS_EXT.1 (selection-based)	Failure to establish a TLS Session	X	X	X	X	Reason for failure.
FIA_AFL.1	Unsuccessful login attempts limit is met or exceeded.	X	X	X	X	Origin of the attempt (e.g., IP address).
FIA_UIA_EXT.1	All use of identification and authentication mechanism.	X	X	X	X	Origin of the attempt (e.g., IP address).
FIA_UAU_EX	All use of identification and	X	X	X	X	Origin of the

SFR	Auditable Events	Date/Time Of Event (“Time”) or (“Start Time” and “End Time”) or (“Timestamp”) field	Type of Event (“Command” and “subcommand”) or (“Event”) or (“Service”) fields	Subject Identity (“User”) or (“Hostname”) field	Outcome (Success or Failure)	Additional Audit Record Contents
T.2	authentication mechanism.					attempt (e.g., IP address).
FIA_X509_EXT.1/Rev	Unsuccessful attempt to validate a certificate Any addition, replacement or removal of trust anchors in the TOE's trust store	X	X	X	X	Reason for failure of certificate validation. Identification of certificates added, replaced or removed as trust anchor in the TOE's trust store.
FMT_MOF.1/ManualUpdate	Any attempt to initiate a manual update	X	X	X	X	None.
FMT_SMF.1	All management activities of TSF data.	X	X	X	X	None.
FPT_TUD_EXT.1	Initiation of update; result of the update attempt (success or failure)	X	X	X	X	None.
FPT_TUD_EXT.2 (selection-based)	Failure of update	X	X	X	X	Reason for failure (including identifier of invalid certificate).
FPT_STM_EXT.1	Discontinuous changes to time - either Administrator actuated or changed via an automated process. (Note that no continuous changes to time need to be logged. See also application note on FPT_STM_EXT.1)	X	X	X	X	For discontinuous changes to time: The old and new values for the time. Origin of the attempt to change time for success and failure (e.g., IP address).
FTA_SSL_EXT.1 (if “terminate the session” is selected)	The termination of a local session by the session locking mechanism.	X	X	X	X	None.
FTA_SSL.3	The termination of a remote session by the session	X	X	X	X	None.

SFR	**Auditable Events**	**Date/Time Of Event** *(“Time”) or (“Start Time” and “End Time”) or (“Timestamp”) field*	**Type of Event** *(“Command” and “subcommand”) or (“Event”) or (“Service”) fields*	**Subject Identity** *(“User”) or (“Hostname”) field*	**Outcome (Success or Failure)**	**Additional Audit Record Contents**
	locking mechanism.					
FTA_SSL.4	The termination of an interactive session.	X	X	X	X	None.
FTP_ITC.1	Initiation of the trusted channel. Termination of the trusted channel. Failure of the trusted channel functions.	X	X	X	X	Identification of the initiator and target of failed trusted channels establishment attempt.
FTP_TRP.1/Admin	Initiation of the trusted path. Termination of the trusted path. Failure of the trusted path functions.	X	X	X	X	None.

Test Result: Pass

2.2 FAU_GEN.1/VPN Audit Data Generation for VPN Gateways

TSS

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

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

Guidance

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

Tests

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

2.2.1 TSS

[ST] Section 7.1.1 describes the TOE security audit subsystem which covers all required audit functions for the TOE, including logging VPN gateway events. No separate subsystems or behaviors are described.

2.2.2 Guidance Documentation

FAU_GEN.1.1		FAU_GEN.1.2					
SFR	**Auditable Events**	**Date/Time Of Event** *(“Time”) or (“Start Time” and “End Time”) or (“Timestamp”) field*	**Type of Event** *(“Command” and “subcommand”) or (“Event”) or (“Service”) fields*	**Subject Identity** *(“User”) or (“Hostname”) field*	**Outcome (Success or Failure)**	**Additional Audit Record Contents**	**Section Reference**
FAU_GEN.1/VPN	Start-up and shut-down of the audit functions;	✓	✓	✓	✓	No additional information.	[AGD] Table 5-2 #24-25
FAU_GEN.1/VPN	Indication that TSF self-test was completed	✓	✓	✓	✓	No additional information.	[AGD] Table 5-2 #23
FAU_GEN.1/VPN	Failure of self-test	✓	✓	✓	✓	No additional information.	[AGD] Table 5-2 #23
FMT_SMF.1/VPN	All administrative actions	✓	✓	✓	✓	No additional information.	[AGD] Table 5-2 #5
FPF_RUL_EXT.1	Application of rules configured with the 'log' operation	✓	✓	✓	✓	Source and destination addresses. Source and destination ports Transport layer protocol.	[AGD] Table 5-2 #22
FTP_ITC.1/VPN	Initiation of the trusted channel	✓	✓	✓	✓	No additional information.	[AGD] Table 5-2 #12
FTP_ITC.1/VPN	Termination of the trusted channel	✓	✓	✓	✓	No additional information.	[AGD] Table 5-2 #12
FTP_ITC.1/VPN	Failure of the trusted channel functions	✓	✓	✓	✓	Identification of the initiator and target of failed trusted channel establishment attempt.	[AGD] Table 5-2 #12

2.2.3 Tests

SFR	**Auditable Events**	**Date/Time Of Event** *("Time") or ("Start Time" and "End Time") or ("Timestamp") field*	**Type of Event** *("Command" and "subcommand") or ("Event") or ("Service") fields*	**Subject Identity** *("User") or ("Hostname") field*	**Outcome (Success or Failure)**	**Additional Audit Record Contents**
FAU_GEN.1/ VPN	Start-up and shut-down of the audit functions;	X	X	X	X	No additional information.
FAU_GEN.1/ VPN	Indication that TSF self-test was completed	X	X	X	X	No additional information.
FAU_GEN.1/ VPN	Failure of self-test	X	X	X	X	No additional information.
FMT_SMF.1/ VPN	All administrative actions	X	X	X	X	No additional information.
FPF_RUL_EXT.1	Application of rules configured with the 'log' operation	X	X	X	X	Source and destination addresses. Source and destination ports. Transport layer protocol.
FTP_ITC.1/VPN	Initiation of the trusted channel	X	X	X	X	No additional information.
FTP_ITC.1/VPN	Termination of the trusted channel	X	X	X	X	No additional information.
FTP_ITC.1/VPN	Failure of the trusted channel functions	X	X	X	X	Identification of the initiator and target of failed trusted channel establishment attempt.

2.3 *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.

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

2.3.1 TSS

This work unit is implicitly satisfied.

2.3.2 Guidance Documentation

This work unit is implicitly satisfied.

2.3.3 Tests

Test Number	1
Test Objective	This test is accomplished in conjunction with testing FAU_GEN.1.1. This is not a distributed TOE.
Test Steps Performed	Testing accomplished in conjunction with FAU_GEN.1.1
Test Result	Pass

2.4 FAU_STG.1 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.

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

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.

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.

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

2.4.1 TSS

[ST] Section 7.1.2 describes that the TOE allocates 10 MB local storage for audit logs. These logs are protected from unauthorized modification or deletion, as users must be authenticated as a security administrator via the CLI to perform the deletion of audit logs.
The TOE is not a distributed TOE.

2.4.2 Guidance Documentation

[AGD] Section 8.7 describes the process of the authorized deletion of audit records and there is no functionality to edit these records. No configuration required to prevent unauthorized modification or deletion of audit records.

2.4.3 Tests

Test Number	1
Test Objective	Verify users are unable to access the audit trail without prior authentication as a security administrator. The TOE only supports one user role.
Test Steps Performed	Attempt to use local console to access log records. Verify that Log records are inaccessible without prior authentication.
Test Result	Pass; local log records are not accessible without prior authentication.

Test Number	2
Test Objective	Verify a user with administrative privileges is able to delete audit records.
Test Steps Performed	Use local console to authenticate as a security administrator and access audit records. Verify that Audit records are accessible. Attempt to delete audit records. The TOE does not allow selective audit records removal. Verify that Records can be deleted by an authorized administrator.
Test Result	Pass; the evaluator verified that local records are removable by a properly authenticated administrator.

2.5 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.

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

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

The evaluator shall examine the TSS to ensure that it details the 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.

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

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

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

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.), as well as configuration of the TOE needed to communicate with the audit server.

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

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

Tests

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

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

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

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

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

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

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

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

2.5.1 TSS

[ST] Section 7.1.2 describes that the TOE transmits audit records in real time to the remote audit server over a TLS channel.

[ST] Section 6.1.1.6 FAU_STG_EXT.1.3 contains selection "overwrite previous audit records".

[ST] Section 7.1.2 describes that the TOE allocates 10 MB local storage for audit logs. Those are protected from unauthorized modification or deletion, as users must be authenticated as a security administrator via the CLI to perform the deletion of audit logs.

[ST] Section 7.1.2 describes that the TOE stores records in up to 10 files of up to 1 MB each and overwrites the oldest file with a new empty file when file size limit is reached.

The TOE is not a distributed TOE.

2.5.2 Guidance Documentation

[AGD] Section 5.1 describes requirements for the remote audit server (listing the audit protocol, TLS version and ciphersuite required).

[AGD] Section 5.1 lists the configuration steps necessary to enable the use of a remote audit server. The [AGD] contains descriptions for each of the require configuration steps.

[AGD] Section 5.1 describes that audit records are sent to the audit server simultaneously as they are generated. It describes possible case where audit data can be lost if connection to the remote audit server is not established, due to log rotation of local records.

[ST] describes that TOE only performs one action FAU_STG_EXT.1.3: deletion of the oldest audit records, and generation of an audit record indicating that audit data overwriting has occurred. [AGD] Section 5.1 describes this behaviour. No configuration is available for this functionality.

2.5.3 Tests

Test Number	1
Test Objective	Objective 1a: Verify the audit data from the TOE is being sent as encrypted data. Objective 1b: Verify audit logs are sent automatically and without administrator intervention.
Test Steps Performed	Use the TOE with the previously configured and started remote audit service. Start a packet capture between the TOE and syslog server. Reboot the TOE. Inspect the packet capture and verify that the session to the syslog server was started without user intervention; verify that the syslog connection is encrypted.
Test Result	Pass; the TOE initiated the connection without user intervention, and the syslog connection was encrypted.

Test Number	2
Test Objective	Verify that auditable events are stored locally and verify that previous audit logs are overwritten as described by [ST].
Test Steps Performed	Authenticate to the TOE using the local console. Navigate to the audit menu. Verify that the Audit menu is displayed. View the oldest audit log records (#9) and the previous log records (#8). Note the oldest log record(s), in this case Log#8 Wait until the audit log rotation is triggered. Verify that the oldest records were replaced by the previous. Verify that Log records #8 becomes log records #9.
Test Result	Pass; the evaluator verified that the TOE stores audit records locally, and complies with the ST-defined behaviour regarding log rotation.

Test Number	3
Test Objective	Verify the TOE lost audit data count is performed correctly. This test is resolved by testing FAU_STG_EXT.2.1/LocSpace - Test1.
Test Steps Performed	This testing is performed in conjunction with FAU_STG_EXT.2.1/LocSpace – Test 1.
Test Result	Pass

Test Number	4
Test Objective	Repeat tests as specified for distributed TOEs. TOE is not a distributed TOE.

Test Steps Performed	TOE is not distributed.
Test Result	Not Applicable

2.6 FAU_STG_EXT.2/LocSpace Counting Lost Audit Data

This activity should be accomplished in conjunction with the testing of FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3.

TSS

The evaluator shall examine the TSS to ensure that it details the possible options the TOE supports for information about the number of audit records that have been dropped, overwritten, etc. if the local storage for audit data is full.

For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies. Since this SFR is optional, it might only apply to some TOE components but not all. This might lead to the situation where all TOE components store their audit information themselves but FAU_STG_EXT.2/LocSpace is supported only by one of the components.

Guidance Documentation

The evaluator shall also ensure that the guidance documentation describes all possible configuration options and the meaning of the result returned by the TOE for each possible configuration. The description of possible configuration options and explanation of the result shall correspond to those described in the TSS.

The evaluator shall verify that the guidance documentation contains a warning for the administrator about the loss of audit data when clearing the local storage for audit records.

Tests

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.

For distributed TOEs the evaluator shall verify the correct implementation of counting of lost audit data for all TOE components that are supporting this feature according to the description in the TSS.

2.6.1 TSS

[ST] Section 7.1.2 describes that the TOE records the number of audit records present in the oldest audit record file that is overwritten when audit storage is full.
The TOE is not a distributed TOE.

2.6.2 Guidance Documentation

[ST] describes that the TOE only performs one action FAU_STG_EXT.1.3: deletion of the oldest audit records, and generation of an audit record indicating that audit data overwriting has occurred. [AGD] Section 5.1 describes this behaviour. No configuration is available for this functionality.

2.6.3 Tests

Test Number	1
Test Objective	Verify overwritten audit data is counted in accordance with the selection in [ST].
Test Steps Performed	Make note of the audit records in the last audit file in the TOE local storage using the local console. Note the last audit record. (Log#10)

	Wait until the log rotation is performed. Verify that a Log rotation record created. Use the log transferred to syslog server to verify the count of deleted data. Verify that the count matches what is reported in the log records.
Test Result	Pass; the evaluator confirmed that the TOE correctly audits the amount of "lost" audit data.

2.7 FCS_CKM.1 Cryptographic Key Generation

TSS

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

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.

Tests

Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products. Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent *e*, the private prime factors *p* and *q*, the public modulus *n* and the calculation of the private signature exponent *d*.

Key Pair generation specifies 5 ways (or methods) to generate the primes *p* and *q*. These include:

a) Random Primes:
   - Provable primes
   - Probable primes

b) Primes with Conditions:
   - Primes p1, p2, q1, q2, p and q shall all be provable primes
   - Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
   - Primes p1, p2, q1, q2, p and q shall all be probable primes

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y.

The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x:

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

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

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

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

- g != 0,1
- q divides p-1
- g^q mod p = 1
- g^x mod p = y

for each FFC parameter set and key pair.

Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups

Testing for FFC Schemes using Diffie-Hellman group 14 and/or safe-prime groups is done as part of testing in CKM.2.1.

2.7.1 TSS

[ST] Section 7.2.1 describes that the TOE supports key generation using EC scheme.

[ST] Section 7.2.1 describes the TOE as using generation ECC key generation using curve P-384 for TLS and curves P-256 and P-384 for IPsec.

2.7.2 Guidance Documentation

[AGD] Section 6 describes that the TOE performs ECDSA key generation with EC curves P-256 and P-384 with the corresponding key sizes.

[AGD] Section 6.1.1 describes the commands needed to generate keys with a specific curve (length).

2.7.3 Tests

Test Number	1
Test Objective	Verify that the TOE correctly implements crypto function, (ECC schemes using ‘NIST curves’ P-256, P-384).
Test Steps Performed	CAVP Certificate A4427 covers key generation for EC curves P-256, P-384.
Test Result	Pass

2.8 FCS_CKM.1/IKE Cryptographic Key Generation (for IKE Peer Authentication) [MOD]

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.

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.

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.
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.

2.8.1 TSS

[ST] Section 7.2.1 describes that the TOE complies with FIPS 186-4 Appendix B.4 (ECC Key Pair Generation) and B.4.2 for the generation of ECC cryptographic key pairs. It describes that the TOE implements all "shall" and "should" statements and does not implement any '"shall not" " or "should not" statements from these two sections. It also describes one case of how a "should" statement is implemented.

[ST] does not describe any TOE-specific extensions.

2.8.2 Guidance Documentation

[AGD] Section 6 describes that the TOE performs ECDSA key generation with EC curves P-256 and P-384 with the corresponding key sizes.

[AGD] Section 6.1.1 describes the commands needed to generate keys with a specific curve (length). It describes that key generation results in a certificate and private key stored in the TOE cryptographic module.

2.8.3 Tests

Test Number	1
Test Objective	Verify that the IKE key generation is tested alongside with key generation for other services. [ST] claims IKE key generation using: ECDSA using curves P-256, P-384.
Test Steps Performed	CAVP Certificate A4427 covers key generation for EC curves P-256, P-384.
Test Result	Pass

2.9 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.

If Diffie-Hellman group 14 is selected from FCS_CKM.2.1, the TSS shall claim the TOE meets RFC 3526 Section 3.

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
Diffie- Hellman (Group 14)	FCS_SSHC_EXT.1	Backup 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.

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).

Tests

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

RSA-based key establishment

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

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.

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.

2.9.1 TSS

[ST] Section 7.2.1 contains Table 6 describing key establishment schemes. It lists schemes used for FCS_TLSC_EXT.1 (Remote Audit server) and FCS_TLSS_EXT.1 (Remote Administrations) as ECDHE using P-384 curve. and for FCS_IPSEC_EXT.1.9 as Diffie Helman group 19 (P-256) and group 20 (P-384). This corresponds to the key generation schemes, as selected in FCS_CKM.1.

2.9.2 Guidance Documentation

[ST] states that the TOE supports only ECDSA key establishment. [AGD] Section 6.4 states that the TOE does not support a configuration for key establishment schemes.

2.9.3 Tests

Test Number	1
Test Objective	Verify key establishment schemes claimed by the TOE; DH using EC groups for IPsec, ECDHE for TLS.

Test Steps Performed	CAVP Certificate A4427 covers key generation for EC curves P-256, P-384.
Test Result	Pass

2.10 FCS_CKM.4 Cryptographic Key Destruction

TSS

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

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

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

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

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

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

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.

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

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command[2] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

2.10.1 TSS

[ST] Section 7.2.1 contains Table 7 that lists cryptographic keys the TOE is using for TLS and IPsec connections. It lists ephemeral (session) secret keys, as well as persistent private keys (associated with TOE x509 certificates). For each key in the table, the [ST] lists a key storage location (RAM or persistent storage) and key destruction situation as well as a destruction method for each situation. Those situations also include restoring factory default keys. Keys are destroyed by overwrite with zeroes or new value of the key for volatile and non-volatile memory.

The evaluator determined that the table lists all secret/private keys that the TOE uses for TSF.

[ST] Section 7.2.1 describes that the TOE destroys keys in non-volatile memory by overwriting them with all zeroes and file metadata removal. It states that Linux OS system API is used to perform key destruction.

The TSS does not identify keys as stored in non-plaintext mode. The [ST] does not contain selections for “a value that does not contain any CSP”.

2.10.2 Guidance Documentation

[AGD] Section 6.1.3 describes that unexpected power loss or VM shutdown may disrupt proper key destruction on a physical/hypervisor layer. It contains a recommendation to ensure these events will not happen during the key destruction procedure and contains guidance to perform a manual deletion operation on key material.

2.10.3 Tests

None.

2.11 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.

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.

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

[2] Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table).

correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

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

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

```
# Input: PT, IV, Key
for i = 1 to 1000:
    if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
    else:
        CT[i] = AES-CBC-Encrypt(Key, PT)
        PT = CT[i-1]
```

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

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

AES-GCM Test

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

128 bit and 256 bit keys

a) **Two plaintext lengths**. One of the plaintext lengths shall be a non- zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

a) **Three AAD lengths**. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

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

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

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

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

AES-CTR Known Answer Tests

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

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

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

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

KAT-3 To test the encrypt functionality, the evaluator shall supply a set of key values for each selected keysize as described below and obtain the ciphertext values that result from AES encryption of an all zeros plaintext using the given key values. A set of 128 128-bit keys, a set of 192 192-bit keys, and/or a set of 256 256-bit keys. Key_i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1, N].

KAT-4 To test the encrypt functionality, the evaluator shall supply the set of 128 plaintext values described below and obtain the ciphertext values that result from encryption of the given plaintext using each selected keysize with a key value of all zeros (e.g. 256 ciphertext values will be generated if 128 bits and 256 bits are selected and 384 ciphertext values will be generated if all keysizes are selected). Plaintext value i in each set shall have the leftmost bits be ones and the rightmost 128-i bits be zeros, for i in [1, 128].

AES-CTR Multi-Block Message Test

The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 less-than i less-than-or-equal to 10 (test shall be performed using AES-ECB mode). For each i the evaluator shall choose a key and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key using a known good implementation. The evaluator shall perform this test using each selected keysize.

AES-CTR Monte-Carlo Test

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

```
# Input: PT, Key
for i = 1 to 1000:
CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]
```

The ciphertext computed in the 1000th iteration is the result for that trial. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation. The evaluator shall perform this test using each selected keysize.

There is no need to test the decryption engine.

2.11.1 TSS

[ST] Section 7.2.2 lists AES modes (GCM and CBC) and key sizes (256 bits) the TOE supports.

2.11.2 Guidance Documentation

[ST] FCS_COP.1/DataEncryption states that the TOE only uses keys with a size of 256 bits. The TOE data encryption modes and key sizes are not configurable.

2.11.3 Tests

Test Number	1
Test Objective	Verify that the TOE correctly implements crypto function (AES in CBC, GCM and CTR mode).
Test Steps Performed	CAVP Certificate A4427 covers key generation for AES CBC, CTR, GCM mode.
Test Result	Pass

2.12 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.

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.

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.

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.

2.12.1 TSS

[ST] Section 7.2.2 describes that the TOE supports the ECDSA digital signature algorithm using 256-bit and 384-bit curves.

2.12.2 Guidance Documentation

[AGD] Section 6 describes that the TOE performs ECDSA key generation with EC curves P-256 and P-384 with corresponding key sizes. [AGD] Section 6.1.1 describes the commands needed to generate keys with a specific curve (length) for IPsec. It states that only P-384 is available for other services – TLS Client and TLS Server. Trusted update signature verification is not configurable.

2.12.3 Tests

Test Number	1
Test Objective	Verify that the TOE correctly implements crypto function (ECDSA using P-256, P-384 curves).
Test Steps Performed	CAVP Certificate A4427 covers signature generation and verification using ECDSA using P-256, P-384 curves.
Test Result	Pass

2.13 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.

Guidance Documentation

The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

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.

2.13.1 TSS

[ST] Section 7.2.2 describes that the TOE uses SHA-384 and SHA-256 hash functions. This section associates those hash functions with other related cryptographic TSFs: IKEv2 PRF, TLS KDF, Keyed hashing (HMAC-SHA-256 and HMAC-SHA-384), signature generation and verification.

[ST] Sections 7.2.2 and 7.5.1 describes the TOE using SHA-512 to hash administrator passwords and trusted update.

2.13.2 Guidance Documentation

The TOE does not support the configuration of hash sizes used by the TOE.

2.13.3 Tests

Test Number	1

Test Objective	Verify that the TOE correctly implements crypto function (SHA-256, SHA-384, SHA-512).
Test Steps Performed	CAVP Certificates A4427 and A4428 cover SHA-256, SHA-384, SHA-512.
Test Result	Pass

2.14 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.

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.

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.

2.14.1 TSS

[ST] Section 7.2.2 describes that the TOE uses HMAC-SHA-256 (based on SHA-256) and HMAC-SHA-384 (based on SHA-384) for TLS and IPsec. [ST] Section 7.2.2 describes that HMAC-SHA-256 uses a 512-bit block size, a digest size of 256 bits, and a key length of 256 bits, HMAC-SHA-384 uses a 1042-bit block size, a digest size of 384 bits, and a key length of 384 bits.

2.14.2 Guidance Documentation

The TOE does not support the configuration of keyed hash sizes used by the TOE.

2.14.3 Tests

Test Number	1
Test Objective	Verify that the TOE correctly implements crypto function (HMAC-SHA-384).
Test Steps Performed	CAVP Certificate A4427 covers HMAC-SHA-384.
Test Result	Pass

2.15 FCS_HTTPS_EXT.1 HTTPS Protocol

TSS

The evaluator shall examine the TSS and determine that enough detail is provided to explain how the implementation complies with RFC 2818.

Guidance Documentation

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

Tests

This test is now performed as part of FIA_X509_EXT.1/Rev testing.

Tests are performed in conjunction with the TLS evaluation activities.

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

2.15.1 TSS

[ST] Section 7.2.3 describes how the TOE complies with RFC 2818, listing and commenting sections from RFC 2818.

2.15.2 Guidance Documentation

[AGD] Section 4.1 describes that the TOE uses the REST API (HTTPS server) as a remote management interface. It describes that the TOE is shipped with preconfigured TLS certificates. Section 6.1 describes how a user can change this certificate. It also describes how the IP address for the remote management interface can be configured.

2.15.3 Tests

Test Number	1
Test Objective	This test is satisfied by the results in FCS_TLSS_EXT.1 and FIA_X509_EXT.1/Rev.
Test Steps Performed	None
Test Result	Pass

2.16 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.

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

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).

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.

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.

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.

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.1.5.

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.1.5.

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.

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.

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.

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.

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.

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.

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.

If any selection with "Pre-shared Keys" is selected, the evaluator shall check that the operational guidance describes any configuration necessary to enable any selected authentication mechanisms.

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.

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.

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.

FCS_IPSEC_EXT.1.6

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

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, the evaluator ensures that the Administrator is able to configure Phase 1 SA values for 24 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.

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, the evaluator ensures that the Administrator is able to configure Phase 2 SA values for 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.

FCS_IPSEC_EXT.1.11

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

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”.

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.

Tests

FCS_IPSEC_EXT.1.1

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

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

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

FCS_IPSEC_EXT.1.2

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

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

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

FCS_IPSEC_EXT.1.3

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

a) 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.

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

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.

FCS_IPSEC_EXT.1.5

Tests are performed in conjunction with the other IPsec evaluation activities.

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

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

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.

FCS_IPSEC_EXT.1.7

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

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

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

b) Test 2: If ‘length of time’ is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime of 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of 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.

FCS_IPSEC_EXT.1.8

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

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

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

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime of 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a lifetime that exceeds the lifetime of 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.

FCS_IPSEC_EXT.1.10

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

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

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

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.

FCS_IPSEC_EXT.1.12

The evaluator simply follows the guidance to configure the TOE to perform the following tests.

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

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

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

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

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.

FCS_IPSEC_EXT.1.14

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

The evaluator shall perform the following tests:

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

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

- Test 3: [conditional] For each CN/identifier type combination selected, the evaluator shall:

  e) Create a valid certificate with the CN so it contains the valid identifier followed by ‘\0’. If the TOE prioritizes CN checking over SAN (through explicit specification of the field when specifying the reference identifier or prioritization rules) for the same identifier type, the evaluator shall configure the SAN so it matches the reference identifier.

f) Configure the peer's reference identifier on the TOE (per the administrative guidance) to match the CN without the '\0' and verify that IKE authentication fails.

- Test 4: [conditional] For each SAN/identifier type combination selected, the evaluator shall:

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

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

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

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

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

  b) Append '\0' to a non-CN field of an otherwise authorized DN.

2.16.1 TSS

FCS_IPSEC_EXT.1.1
[ST] Section 7.2.4 describes that the TOE uses Linux iptables to enforce SPD packet filtering. SPD entries are enforced in the order the rules are configured. It describes that TSF allowed packet rules to allow packets flow through TOE with no protection (BYPASS), be dropped (Drop), be encrypted (Allow) as defined in RFC 4301. Packets not matching any rules are dropped.

[ST] Section 7.4.2. describes that packet filtering becomes active prior to activation of networking interfaces.

[ST] Section 7.4.2. describes the packet filtering criteria that can be applies to the rules. It describes that the security administrator can configure rules using CLI or Remote management interface. Rules configured through the remote management traffic selectors interface will only be applied by the TOE when IPsec service is running.

[ST] Section 7.4.2 describes that rules required for basic IPsec operation are automatically generated and enforced when IPsec service is started. Those rules are then disabled when service is stopped. Rules entered manually will remain in the packet filtering chain even when service is stopped.

FCS_IPSEC_EXT.1.3
[ST] Section 7.2.4 describes the TOE as only supporting tunnel mode, which is consistent with the selection in FCS_IPSEC_EXT.1.3.

FCS_IPSEC_EXT.1.4
[ST] Section 7.2.4 describes the TOE as using HMAC-SHA-384 for the IKEv2 payload. It also describes HMAC truncated to 192-bits is utilized.

FCS_IPSEC_EXT.1.5
[ST] Section 7.2.4. describes the TOE as supporting IKEv2 only.

FCS_IPSEC_EXT.1.6
[ST] Section 7.2.4. describes the TOE as supporting AES-GCM-256 and AES-CBC-256 to encrypt the IKEv2 payload which is consistent with the selection in FCS_IPSEC_EXT.1.5.

FCS_IPSEC_EXT.1.7
[ST] Section 7.2.4 describes the TOE as supporting an IKEv2 IKE_SA lifetime configuration from 4 to 120 hours which consistent with the selection in FCS_IPSEC_EXT.1.5.

FCS_IPSEC_EXT.1.8
[ST] Section 7.2.4 describes the TOE as supporting an IKEv2 CHILD_SA lifetime configuration from 2 to 48 hours which consistent with the selection in FCS_IPSEC_EXT.1.5.

FCS_IPSEC_EXT.1.9
[ST] Section 7.2.4 describes that the TOE uses DH groups 19 and 20. It describes the TOE as generating the private key according to the negotiated DH group; 256 bits for group 19 and 384 bits for group 20, which is consistent with the requirements to be twice the security strength of the DH group.

FCS_IPSEC_EXT.1.10
[ST] Section 7.2.4 describes the TOE as using DH groups 19 and 20. It describes that nonces used by the TOE for key exchange are generated by the DRBG and are 265 bits long, which is sufficient to satisfy the length requirements for DH groups and the PRF used by the TOE.

FCS_IPSEC_EXT.1.11
[ST] Section 7.2.4 describes the TOE as using DH groups 19 and 20. It describes the groups will be negotiated in the following order: 19 then 20.

FCS_IPSEC_EXT.1.12
[ST] Section 7.2.4 describes that the TOE uses the same key size for CHILD_SA and IKE_SA as the TOE supports only 256-bit keys. If a peer attempts to negotiate a CHILD_SA with a key size that has not been configured on the TSF, the connection will fail.

[ST] Section 7.2.2 describes that IKEv2 payload is using AES-GCM-265 or AES-CBC-256, ESP is using AES-GCM-256 algorithm. It mentions those algorithms are using key size of 256 bits.

FCS_IPSEC_EXT.1.13
[ST] Section 7.2.4 describes that TOE authenticates peers using x509 certificates using ECDSA with P-256 and P-384 curves. This is consistent with the selections in FCS_COP.1/SigGen.

[ST] Section 6.1.2.10 FCS_IPSEC_EXT.1.13 does not contain a selection for pre-shared keys.

FCS_IPSEC_EXT.1.14
[ST] Section 7.2.4. describes the TOE as verifying peer certificates by comparing the presented identifier in the peer certificate with the security administrator configured reference identifier, where reference identifier is a Distinguished Name (DN). It describes that the TSF supports CN, O, OU, L, S, C as a Relative Distinguished Name.

2.16.2 Guidance Documentation

FCS_IPSEC_EXT.1.1
[AGD] Sections 7.2.4 and 7.2.5 describe that the SPD entries are configured in the TOE using the local console or remote management interface. [RAR] and Section 7.2.5 describe how the SPD entries are created using the remote management interface. Those entries can ensure packets are encrypted/decrypted, bypassed or dropped. [AGD] Appendix B and Section 7.2.6 describes how the SPD entries can be created using the local console. Those entries can be used to drop packets or pass them through the TOE unencrypted. [AGD] Sections 7.2.4 and 7.2.5 describe that rules are applied in the order they are configured and that the first matching rule is applied. [AGD] Section 7.2.6 describes that rules generated through the SPD will be added to the end of the firewall chain when the service is restarted.

Pre-shared keys are not selected in [ST].

FCS_IPSEC_EXT.1.3
The TOE supports tunnel mode only. No configuration is necessary. [AGD] describes configuration of the establishment connection in this mode.

FCS_IPSEC_EXT.1.4
ESP algorithms are not configurable in the TOE.

FCS_IPSEC_EXT.1.5
The TOE supports IKEv2 only, which is not configurable.

FCS_IPSEC_EXT.1.6
IKEv2 algorithms are not configurable in the TOE.

FCS_IPSEC_EXT.1.7
[AGD] Section 7.2.2 describes that the administrator can configure SA lifetime limits using the local console. During testing, it was observed that the TOE performs rekey before a configured threshold. It describes that the SA lifetime can be configured in a range of 4 to 120 hours.

FCS_IPSEC_EXT.1.8
[AGD] Section 7.2.2 describes that the administrator can configure SA lifetime limits using the local console. During testing it was observed that the TOE performs a rekey before a configured threshold. It describes that the Child SA lifetime can be configured in a range of 4 to 120 hours.

FCS_IPSEC_EXT.1.11
IKE DH groups are not configurable in the TOE.

FCS_IPSEC_EXT.1.13
[AGD] Section 6 describes how to configure the TOE to use ECDSA certificates and keys. It describes how to configure the trust store and upload trusted CAs for the IPsec service.
The TOE does not support pre-shared keys.

FCS_IPSEC_EXT.1.14
[AGD] Section 7.2.3 describes how the IPsec peer reference identifier (DN) is configured. This is consistent with the selections in [ST] Section FCS_IPSEC_EXT.1.14. [AGD] Section 7.2.3. contains a warning that it must be ensured that the DN is unique.

2.16.3 Tests

FCS_IPSEC_EXT.1.1

Test Number	1
Test Objective	Verify that the TOE is able to apply SPD rule to encrypt packets. Verify that the TOE is able to apply SPD rule to bypass packets. Verify that the TOE is able to apply SPD rule to drop packets. Verify that the TOE applies implicit rule when no rule matches the packet. (This is resolved by section 8.5.1.10 FPF_RUL_EXT.1.6 – Test 3)
Test Steps Performed	Configure the TOE to: • Encrypt all packets from 10.1.2.0/24 to 172.16.2.0/24 network. • Encrypt TCP packets to port 80 in the 172.16.2.0/24 network.

	• Drop TCP packets to port 90 in the 172.16.2.0/24 network. • Drop TCP packets to port 443 in the 10.2.2.0/24 network. • Bypass TCP packets to port 80 in the 10.2.2.0/24 network. • Drop TCP packets to port 2289 to any network. Verify that the TOE is configured. Restart IPSec service on the TOE, verify the effective rules using the local console. Verify that the rules are configured accordingly. Initiate a packet capture on both sides of the TOE and send test packets from lefthost1 to 172.16.2.2 port 80 using nc tool. Verify that Packets to port 80 are encrypted. Initiate a packet capture on both sides of the TOE and send test packets from lefthost1 to 172.16.2.2 port 90 using nc tool. Verify that Packets to port 90 are dropped. Establish a connection to port 443 to 10.2.2.3. Verify that Packets to port 443 are dropped. Establish a connection to port 80 to 10.2.2.3 using nc tool. Verify that Packets to port 80 are bypassed.
Test Result	Pass; The TOE correctly encrypted, dropped, or bypassed traffic according to the defied rules.

Test Number	2
Test Objective	Verify that the TOE is able to apply conflicting or overlapping SPD rules to encrypt packets. (Step 3,4) Verify that the TOE is able to apply conflicting or overlapping SPD rules to bypass packets. (Step 2) Verify that the TOE is able to apply conflicting or overlapping SPD rules to drop packets. (Step 2,4) Verify that the TOE applies implicit rule when no rule matches the packet. (This is resolved by section 8.5.1.10 FPF_RUL_EXT.1.6 – Test 3)
Test Steps Performed	Add additional rules to create conflicting entries using Rest API: • Drop all packets to any network on TCP port 80 with less priority than the rule that allows this traffic to 10.2.2.0 network, but higher priority than the rule that allows this traffic to port 80 to 172.16.2.0 and 10.1.2.1 network. • Create a rule that allows TCP packets on port 2289 to 10.2.2.5 with higher priority than the rule that denies these packets for all networks. Verify that the TOE is configured. Perform a TCP connection on port 2289 to 10.2.2.5 and 10.2.2.3.Packets to 10.2.2.5 are allowed and packets to 10.2.2.3 are dropped. Send TCP packets on port 80 to 10.2.2.5 and 172.16.2.2. Verify that packets to 10.2.2.5 are allowed and packets sent to 172.16.2.2 are dropped. Send TCP packets from righthost (172.16.2.2) port 80 to lefthost (10.1.2.1) port 80 through the established tunnel. Verify that the packet is routed through SA but dropped by the TOE, as the

	tunnel is configured to accept packets from port 80 of righthost, but SPD rules with higher priority should drop the packet.
Test Result	Pass; The TOE correctly encrypted, dropped, or bypassed traffic as expected, in all cases.

FCS_IPSEC_EXT.1.2

Test Number	1
Test Objective	Verify that the packets can be sent through the TOE as plaintext without modification (resolved by testing in FCS_IPSEC_EXT1.1, Test 1, Step 6). Verify that the packets not matching created rules are dropped (this is resolved by testing in FPF_RUL_EXT.1.6 – Test 3).
Test Steps Performed	This test is resolved by testing in FCS_IPSEC_EXT1.1 Test 1, Step 6, and FPF_RUL_EXT.1.6 – Test 3.
Test Result	Pass

FCS_IPSEC_EXT.1.3

Test Number	1
Test Objective	Verify that the connection can successfully be established through the TOE in tunnel mode.
Test Steps Performed	Configure the test gateway as described in section 6 as per [AGD]. Verify that the TOE is configured. Initiate a connection between the TOE and the test gateway. Display the connection status. Verify that a connection is established in tunnel mode and the SA is negotiated.
Test Result	Pass; The TOE correctly negotiated IPsec tunnel mode connection between itself and peers.

Test Number	2
Test Objective	If transport mode is selected, the evaluator uses the guidance documentation to configure the TOE to operate in transport mode and also configures a VPN peer to operate in transport mode.
Test Steps Performed	The TOE uses tunnel mode only.
Test Result	Pass; The TOE does not use transport mode.

FCS_IPSEC_EXT.1.4

Test Number	1
Test Objective	Verify that the TOE implements ESP as defined in the [ST]: AES-GCM-256 (RFC 4106).
Test Steps Performed	Establish an IPsec connection and verify that AES-GCM was used for ESP in the connection on the TestGW. Verify that only AES-GCM-256 is used by the TOE.
Test Result	Pass; the TOE Implements AES-GCM-256 as defied in [ST].

FCS_IPSEC_EXT.1.5

Test Number	1
Test Objective	N/A
Test Steps Performed	The [ST] does not claim IKEv1.

Test Result	Pass; The TOE does not support IKEv1.

Test Number	2
Test Objective	Verify that the TOE is able to conduct NAT traversal.
Test Steps Performed	Configure the test environment so that peer gateway will be behind NAT server. Verify that the test environment and TOE is configured correctly. Establish an IPsec tunnel. Observe network packets. Verify that the Tunnel is established through the NAT server, using NAT traversal.
Test Result	Pass; The TOE correctly established the IPsec tunnel while traversing switches or routers in NAT configuration.

FCS_IPSEC_EXT.1.6

Test Number	1
Test Objective	Verify the TOE uses ciphersuites defined in the [ST] for IKE (AES-GCM-256 and AES-CBC-256).
Test Steps Performed	Configure the testGW to accept only AES-GCM-256 with SHA384. Verify that the testGW is configured. Establish a connection. Verify that the Connection is established using AES-GCM cipher. Configure the testGW to accept only AES-CBC-256 with SHA384. Verify that the testGW is configured. Establish a connection. Verify that the Connection is established using AES-CBC cipher.
Test Result	Pass; the TOE accepted the configured cipher, and only the configure cipher.

FCS_IPSEC_EXT.1.7

Test Number	1
Test Objective	Verify the TOE performs Phase 1 re-negotiation, as defined in [ST].
Test Steps Performed	The [ST] does not contain “number of bytes” selection, so this test is not applicable to the TOE.
Test Result	Pass; The TOE does not implement “number of bytes” lifetimes.

Test Number	2
Test Objective	Verify the TOE performs re-negotiation, as defined in [ST]. [ST] defines the lifetime as configured from 4 to 120 hours.
Test Steps Performed	Using the local console, configure the TOE Phase1 SA lifetime to be 7 hours. Configure the TestGW Phase 1 SA lifetime to be 36 hours. Verify that the TOE and TestGW are configured. Establish a connection. Note connection time. Note the current time. Maintain a connection for 24 hours. Investigate the logs to determine when IKE renegotiation was performed. Verify that the TOE initiates an IKE re-negotiation no later than every 7 hours.

Test Result	Pass; The TOE renegotiated the IKE session every 7 hours.

FCS_IPSEC_EXT.1.8

Test Number	1
Test Objective	Verify the TOE performs Phase 2 re-negotiation, as defined in [ST].
Test Steps Performed	The [ST] does not contain “number of bytes” selection, so this test is not applicable to the TOE.
Test Result	Pass; The TOE does not implement “number of bytes”.

Test Number	2
Test Objective	Verify the TOE performs re-negotiation, as defined in [ST]. [ST] defines the lifetime as configured from 4 to 120 hours.
Test Steps Performed	Using the local console configure the TOE Phase2 SA lifetime to be 6 hours. Configure the TestGW Phase 2 SA lifetime to be 13 hours. Verify that the TOE and TestGW are configured. Establish a connection. Note connection time. Note the current time on the TOE. Maintain a connection for 8 hours. Investigate the logs to determine when Child SA renegotiation was performed. Verify that ESP phase re-negotiation is performed no later than every 6 hours.
Test Result	Pass; The TOE renegotiated the ESP phase session every 6 hours.

FCS_IPSEC_EXT.1.10

Test Number	1
Test Objective	Verify that the Nonce generated for DH group corresponds with the group length.
Test Steps Performed	This test is not applicable as the first selection is not chosen in the [ST].
Test Result	Pass; The TOE does not implement DH groups.

Test Number	2
Test Objective	Verify that the Nonce generated is at least 128 bits in strength and at least half the size of the negotiated PRF function (PRF-HMAC-SHA2-384/HMAC_SHA2-384).
Test Steps Performed	Perform this test when performing test FCS_IPSEC_EXT.11. Inspect the packet capture for the channel establishment done using ECP256 group. Verify that the Nonce is at least 192 bits long. Inspect the packet capture for the channel establishment done using ECP384 group. Verify that the Nonce is at least 192 bits long.
Test Result	Pass; The TOE used ECP256 with 192 bit nonces when establishing the session.

FCS_IPSEC_EXT.1.11

Test Number	1
Test Objective	Very that the TOE supports DH groups 19 (256-bit Random ECP), 20 (384-bit Random ECP) according to RFC 5114, for IKEv2; the TOE does

	not claim support for IKEv1.
Test Steps Performed	Configure the TestGW to accept only DH group 19 (ECP256). Verify that the TestGW is configured. Establish a connection. Verify that the connection is established using ECP256 group. Configure the TestGW to accept only DH group 20 (ECP384). Verify that the TestGW is configured. Establish a connection. Verify that the connection is established using ECP256 group.
Test Result	Pass; The TOE implements only the listed key establishment groups.

FCS_IPSEC_EXT.1.12

Test Number	1
Test Objective	Verify that the TOE can establish an IPsec connection using AES-256-GCM and AES-256-CBC, and HMAC-SHA2-384 for IKEv2. The TOE does not claim support for IKEv1.
Test Steps Performed	This test is resolved by testing in FCS_IPSEC_EXT.1.6 Test 1.
Test Result	Pass; The TOE established IPsec connections using the listed ciphers and hashing ciphers.

Test Number	2
Test Objective	Verify that the TOE does not support a connection with ESP encryption with a key larger than 256 bits.
Test Steps Performed	This test is implicitly resolved as the TOE uses AES-256-GCM with a 256-bit key which is the maximum for IKE SA.
Test Result	Pass; The TOE used the listed ciphers to establish the connection.

Test Number	3
Test Objective	Verify that the TOE would not establish an IKE SA with algorithms other than AES-256-GCM or AES-256-CBC with SHA2-384 for IKEv2. The TOE does not claim support for IKEv1.
Test Steps Performed	Configure the TestGW to only accept AES-128-CBC encryption with SHA2-384 for IKE SA. Verify that the TestGW is configured. Try to establish a connection. Verify that session Establishment is rejected. Configure the TestGW to only accept AES-256-CBC encryption with SHA2-256 for IKE SA. Verify that the TestGW is configured. Try to establish a connection. Verify that session Establishment is rejected.
Test Result	Pass; The TOE correctly denied session establishment with incorrect ciphers.

Test Number	4
Test Objective	Verify that the TOE supports only the Encryption algorithm identified in

	[ST] for IKEv2 ESP; the TOE does not claim support for IKEv1.
Test Steps Performed	Configure the TestGW to use only AES128-GCM for ESP. Verify that the TestGW is configured as expected. Try to establish a connection. Verify that establishment is rejected.
Test Result	Pass; The TOE correctly rejected connections which did not use supported bulk encryption algorithms.

FCS_IPSEC_EXT.1.13

Test Number	1
Test Objective	Verify that IKE protocols perform peer authentication using ECDSA that use X.509v3 certificates that conform to RFC 4945.
Test Steps Performed	This test is resolved by tests FIA_X509_EXT.1.
Test Result	Pass; The TOE correctly performed peer identification and authentication using x.509v3 certificates.

FCS_IPSEC_EXT.1.14

Test Number	1
Test Objective	Verify that the TOE will accept peer certificates with correctly identified CN containing IP address and will **not** accept a certificate with correctly identified CN IP address and incorrectly identified SAN IP address.
Test Steps Performed	[ST] does not claim CN identifiers support for IPsec.
Test Result	Pass; The TOE does not support CNs.

Test Number	2
Test Objective	Verify that the TOE supports claimed SAN identifiers (IP address in SAN field).
Test Steps Performed	[ST] does not claim SAN identifiers support for IPsec.
Test Result	Pass; The TOE does not support SAN for IPsec.

Test Number	3
Test Objective	Verify the TOE performs bit-by-bit comparison of the CN.
Test Steps Performed	[ST] does not claim CN identifiers support for IPsec.
Test Result	Pass; The TOE does not support CNs for IPsec.

Test Number	4
Test Objective	Verify the TOE will not accept a certificate with an incorrect SAN identifier but correct corresponding identifier in the DN.
Test Steps Performed	[ST] does not claim SAN identifiers support for IPsec.
Test Result	Pass; The TOE does not support SAN Identifiers for IPsec.

Test Number	5
Test Objective	Verify that the TOE correctly authenticates peers using DN.
Test Steps Performed	Check peer DN. Configure the TOE to use DN identifier to match peer DN using rest API. Verify that the TOE is configured.

	Establish a connection. Check connection status on TOE and peer. Verify that a Connection is established.
Test Result	Pass; The TOE authenticated peers using DNs.

Test Number	6
Test Objective	Verify that the TOE correctly performs bit-wise comparison on DN for authentication.
Test Steps Performed	Configure the TestGW to use a leaf certificate with duplicate CN field. Verify that the TestGW is configured. Establish a connection. Verify that the Connection is rejected. Configure the TestGW to use a leaf certificate with /0 appended to non-CN field. Verify that the TestGW is configured. Establish a connection. Verify that the connection is rejected.
Test Result	Pass; The TOE correctly performed bit-wise comparisons of the DN when performing authentication.

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

Documentation shall be produced—and the evaluator shall perform the activities—in accordance with Appendix D of [NDcPP].

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.

Guidance Documentation

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

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.

2.17.1 TSS

[ST] Section 7.2.5 describes the TOE as using CTR_DRBG(AES) with 256 bits. The DRBG is seeded using the Intel CPU-provided RDSEED instruction with the 640 bits from entropy source, that provides 256 bits of min-entropy.

2.17.2 Guidance Documentation

The TOE does not support the configuration of the DRBG.

2.17.3 Tests

Test Number	1
Test Objective	Verify the TOE RNG is working correctly. This test is resolved by a CAVP certificate.
Test Steps Performed	CAVP Certificate A4427 covers Counter-DRBG.
Test Result	Pass; The TOE implements DRBG correctly.

2.18 FCS_TLSC_EXT.1 Extended: TLS Client Protocol without Mutual Authentication (Selection-Based)

TSS

FCS_TLSC_EXT.1.1

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

FCS_TLSC_EXT.1.2

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

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

If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE’s conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC 5952 for IPv6, RFC 3986 for IPv4) is enforced.

FCS_TLSC_EXT.1.4

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

Guidance Documentation

FCS_TLSC_EXT.1.1

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

FCS_TLSC_EXT.1.2

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

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

FCS_TLSC_EXT.1.4

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

Tests

For all tests in this chapter the TLS server used for testing of the TOE shall be configured not to require mutual authentication.

FCS_TLSC_EXT.1.1

Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

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

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

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

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

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

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

Test 5: The evaluator performs the following modifications to the traffic:

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

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

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

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

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

c) Modify at least one byte in the server's nonce in the Server Hello handshake message and verify that the client rejects the Server Key Exchange handshake message (if using a DHE or ECDHE ciphersuite) or that the server denies the client's Finished handshake message.

FCS_TLSC_EXT.1.2

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

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

or

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

or

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

Note that for some tests additional conditions apply.

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

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.
- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

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

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

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

b) Test 2 [conditional]: The evaluator shall present a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported SAN type (e.g. IPv4, IPv6, FQDN, URI). When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the SAN.

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

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

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

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

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

f) Test 6 [conditional]: If IP addresses are supported, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with an asterisk (*) (e.g. CN=192.168.1.* when connecting to 192.168.1.20, CN=2001:0DB8:0000:0000:0008:0800:200C:* when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6).

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

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

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

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

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

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

FCS_TLSC_EXT.1.3

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

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

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

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

FCS_TLSC_EXT.1.4

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

2.18.1 TSS

FCS_TLSC_EXT.1.1
[ST] Section 7.2.6 describes that the TOE implements TLS version 1.2 according to RFC 4492, 5246, 5289, 6125. It describes that the TOE supports only TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384. This is consistent with the selections in FCS_TLSC_EXT.1.1.

FCS_TLSC_EXT.1.2
[ST] Section 7.2.6 describes that the TOE generates reference identifiers from the syslog DNS name. The TOE checks the server certificate for the SAN extension with DNS-ID matching the reference identifier, if the SAN extension is not present, the TOE compares the CN field to the DNS reference identifier. This comparison is performed according to Section 6.4 of RFC 6125. The TSF does not support wildcards in the reference identifiers. IP addresses in CN and/or SAN are not supported by the TSF.

FCS_TLSC_EXT.1.4
[ST] Section 7.2.6 describes that the TOE uses a secp384r1 elliptic curve for the TLS handshake. This functionality is not configurable.

2.18.2 Guidance Documentation

FCS_TLSC_EXT.1.1

[ST] Section 7.2.6 describes that the TLS client supports TLS v1.2 and a TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 ciphersuite with a secp384r1 curve. It describes that the ciphersuite and curves are not configurable.

[AGD] Sections 7.1 and 5.1.1 describe the TLS Client functionality consistently with [ST]. [AGD] Section 7.1 describes that the TOE does not provide configuration for those TLS parameters.

FCS_TLSC_EXT.1.2
[AGD] Section 5.1.1 describes that the TOE supports domain name identifier in the SAN or CN fields of the certificates. It describes how reference identifiers can be configured in the TOE. The TOE does not support IP address reference identifiers.

FCS_TLSC_EXT.1.4
TSS describes that the ciphersuite and curves for the TLS clients are not configurable in the TOE. [AGD] Section 6.1 and 7.1 describe that the TOE does not provide configuration for those TLS parameters.

2.18.3 Tests

FCS_TLSC_EXT.1.1

Test Number	1
Test Objective	Verify that the ciphersuite(s) claimed in the [ST] can be negotiated. [ST] claims one ciphersuite — TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 with NIST P-384 curve.
Test Steps Performed	Configure syslog server to receive connections from the TOE. Establish a connection. Verify the TOE used ECDHE_ECDSA_WITH AES_256_GCM_SHA384 ciphersuite when connecting to the syslog server. Verify that the connection is established.
Test Result	Pass; The TOE correctly established the TLS session with the claimed ciphersuite.

Test Number	2
Test Objective	Verify the TOE will only accept a server certificate with Server Authentication extendedKeyUsage extension set.
Test Steps Performed	Create a certificate that contains a Server Authentication field. Verify that the certificate is created. Use this certificate in a syslog server. Let the TOE establish a connection. Verify that the connection is successful. Create a similar certificate without a Server Authentication purpose in the EKU field. Verify that the certificate is created. Use this certificate in a TLS server. Let the TOE establish a connection. Verify that the Connection is rejected.
Test Result	Pass; The TOE correctly accepted the certificate with the correct EKU field.

Test Number	3
Test Objective	Verify the TOE will not accept a server certificate if it is not matching the negotiated ciphersuite.
Test Steps Performed	Create a certificate with correct parameters and trust chain and an RSA

	public key. Verify that the certificate is created. Configure the test server to present this certificate using UL test tool as follows and initiate a connection with the TOE: /test-tools/uls2n/s2nd --rcert ./rd2crts/TLSC.1.1-Test_3.crt --rkey ./rd2crts/leafRSA.pem --ecert ./rd2crts/TLSC.1.1-Working.crt --ekey ./rd2crts/leaf0.pem 10.3.2.4:5514 1-3 Verify that the TOE rejects the connection.
Test Result	Pass; The TOE correctly rejected the connection with the improper ciphersuite.

Test Number	4
Test Objective	Verify the TOE rejects TLS_NULL_WITH_NULL_NULL ciphersuite. Verify the connection is not established when the server provides a ciphersuite not present in the Client Hello. Verify the TOE rejects the connection using a non-supported EC group (other than P-384).
Test Steps Performed	Configure the server to offer a TLS_NULL_WITH_NULL_NULL using UL test tool as follows: /test-tools/uls2n/s2nd --rcert ./rd2crts/TLSC.1.1-Test_3.crt --rkey ./rd2crts/leafRSA.pem --ecert ./rd2crts/TLSC.1.1-Working.crt --ekey ./rd2crts/leaf0.pem 10.3.2.4:5514 1-4 Initiate a connection with the TOE. Verify that the TOE rejects the connection. Configure the server to select an unnegotiated ciphersuite using UL tool as follows: /test-tools/uls2n/s2nd --rcert ./rd2crts/TLSC.1.1-Test_3.crt --rkey ./rd2crts/leafRSA.pem --ecert ./rd2crts/TLSC.1.1-Working.crt --ekey ./rd2crts/leaf0.pem 10.3.2.4:5514 1-5c Initiate a connection from the TOE. Verify that the connection is rejected. Configure the server to use an unsupported curve in ECDHE using: /test-tools/uls2n/s2nd --rcert ./rd2crts/TLSC.1.1-Test_3.crt --rkey ./rd2crts/leafRSA.pem --ecert ./rd2crts/TLSC.1.1-Working.crt --ekey ./rd2crts/leaf0.pem 10.3.2.4:5514 4 Initiate a connection from the TOE. Verify that the TOE rejects the connection.
Test Result	Pass; The TOE rejected the connection with modified parameters.

Test Number	5
Test Objective	Verify the TOE only supports TLS versions selected in the [ST] – v. 1.2. Verify the TOE rejects the connection with a corrupted signature block in the Server Key Exchange message.
Test Steps Performed	Configure the server to offer a non-supported TLS version 1.3 in the server hello message. /test-tools/uls2n/s2nd –rcert syslog.local-test3.crt –rkey syslog.local-test3.key –ecert syslog.local-working.crt –ekey leaf1.key 10.3.2.4:5514 1-5a

	Verify that the server is configured. Initiate a connection from the TOE. Inspect the packet capture. Verify that the connection is rejected. Configure the server to corrupt the signature block in handshake message using UL tool: /test-tools/uls2n/s2nd –rcert syslog.local-test3.crt –rkey syslog.local-test3.key –ecert syslog.local-working.crt –ekey leaf1.key 10.3.2.4:5514 1-5d Verify that the server is configured. Initiate a connection from the TOE. Inspect the packet capture. Verify that the connection is rejected.
Test Result	Pass; The TOE supports only TLS v1.2 and rejected certificates with corrupted signature blocks.

Test Number	6
Test Objective	Verify the TOE rejects a handshake with a corrupted Server Finished message, garbled message before Server Finished message, and modified nonce in server key exchange message.
Test Steps Performed	Configure the test server to corrupt the Server finished message: /test-tools/tls2y/s2nd –rcert syslog.local-test3.crt –rkey syslog.local-test3.key –ecert syslog.local-working.crt –ekey leaf1.key 10.3.2.4:5514 1-5e Verify that the server is configured. Initiate a connection from the TOE. Inspect the packet capture. Verify that the connection is rejected. Configure the test server to send a garbled message after the server has issued the ChangeCipherSpec using UL test tool: /test-tools/tls2y/s2nd –rcert syslog.local-test3.crt –rkey syslog.local-test3.key –ecert syslog.local-working.crt –ekey leaf1.key 10.3.2.4:5514 1-5f Verify that the server is configured. Initiate a connection from the TOE. Inspect the packet capture. Verify that the connection is rejected. Configure the test server to modify server nonce using UL test tool: /test-tools/tls2y/s2nd –rcert syslog.local-test3.crt –rkey syslog.local-test3.key –ecert syslog.local-working.crt –ekey leaf1.key 10.3.2.4:5514 1-5b Verify that the server is configured. Initiate a connection from the TOE. Inspect the packet capture. Verify that the connection is rejected.
Test Result	Pass; The TOE rejected a connection where the Server Finished message was corrupt, or a garbled message before the Server Finished message, or a message with a modified nonce in the key exchange message.

FCS_TLSC_EXT.1.2

Test Number	1
Test Objective	Verify the TOE rejects the certificate with incorrect CN in the absence of

	the SAN field in the presented certificate: • FQDN in the CN field.
Test Steps Performed	Create server certificate with an incorrect FQDN in the CN field without a SAN field. Verify that the certificate is created correctly. Configure the test server to use the created certificate and initiate a connection with the TOE. Verify that The TOE rejects the connection.
Test Result	Pass; The TOE rejected connections when the certificate had an incorrect CN.

Test Number	2
Test Objective	Verify the TOE rejects the certificate with the correct CN and incorrect identifier in the SAN field in the presented certificate: • FQDN in the SAN field.
Test Steps Performed	Create server certificate with an incorrect FQDN in the SAN field with correct CN fields. Verify that the certificate is created correctly. Configure the test server to use the created certificate and initiate a connection with the TOE. Verify that The TOE rejects the connection.
Test Result	Pass; The TOE rejected connections when the certificate had the correct CN and an incorrect SAN

Test Number	3
Test Objective	Verify the TOE accepts the certificate with a correct CN in the absence of the SAN field in the presented certificate: • FQDN in the CN field.
Test Steps Performed	Create a server certificate with a correct FQDN in the CN field and without a SAN field. Verify that the certificate is created correctly. Configure the test server to use the created certificate with a correct FQDN in the CN field without a SAN and initiate a connection with the TOE. Verify that The TOE accepts the connection.
Test Result	Pass; The TOE correctly accepted connections when the CN was correct and there was no SAN.

Test Number	4
Test Objective	Verify the TOE accepts a certificate with an incorrect CN and correct SAN field in the presented certificate: • FQDN in the SAN and CN field.
Test Steps Performed	Create a server certificate with an incorrect FQDN in the CN field and with the correct FQDN in the SAN field. Verify that the certificate is created. Configure the test server to use the created certificate with an incorrect FQDN in the SAN field with a correct FQDN in the CN field and initiate a

	connection with the TOE. Verify that the TOE accepts the connection.
Test Result	Pass; The TOE correctly accepted connections when the certificate had an incorrect CN and the correct SAN.

Test Number	5
Test Objective	Verify the TOE supports a certificate with a wildcard in the left-most label in the SAN if claimed in the ST, otherwise confirm that TOE does not support wildcards. [ST] states TSF does not support wildcards. Verify the TOE supports a certificate with a wildcard in the left-most label in the CN if claimed in the ST, otherwise confirm that TOE does not support wildcards. [ST] states TSF does not support wildcards.
Test Steps Performed	Create two server certificates: one with the wildcard DNS reference identifier in the CN field and one with the wildcard DNS reference identifier in the SAN field. Verify that the certificates are created. Configure the TOE to use DNS reference identifier, “foo.syslog.local’. Verify that the TOE is configured successfully. Configure the test server to use the created certificate with a FQDN with a wildcard in the left-most label in the CN and initiate a connection with the TOE. Verify that the TOE rejects the connection. Configure the test server to use the created certificate with a FQDN with a wildcard in the left-most label in the SAN and initiate a connection with the TOE. Verify that the TOE rejects the connection.
Test Result	Pass; The TOE correctly accepted connections with wildcards in the left-most label, but nowhere else.

Test Number	6
Test Objective	Verify the TOE is able to differentiate between IP address identifiers that are not allowed to contain wildcards and other types of identifiers that may contain wildcards.
Test Steps Performed	This test is not applicable. The TOE does not support IP address identifiers.
Test Result	Pass; The TOE does not support IP address identifiers.

Test Number	7
Test Objective	1) The evaluator shall present a server certificate that does not contain an identifier in the Subject (DN) attribute type(s) that matches the reference identifier. The evaluator shall verify that the connection fails. 2) The evaluator shall present a server certificate that contains a valid identifier as an attribute type other than the expected attribute type (e.g. if the TOE is configured to expect id-at-serialNumber=correct_identifier, the certificate could instead include id-at-name=correct_identifier), and does not contain the SAN extension. The evaluator shall verify that the connection fails. Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN

	extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass this test. 3) The evaluator shall present a server certificate that contains a Subject attribute type that matches the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection succeeds. 4) The evaluator shall confirm that all use of wildcards results in connection failure regardless of whether the wildcards are used in the left or right side of the presented identifier. (Remark: Use of wildcards is not addressed within RFC 5280.)
Test Steps Performed	This test not applicable as TLS channel is used for FPT_ITC.1.
Test Result	Pass; The TOE does not implement FPT_ITT.

FCS_TLSC_EXT.1.3

Test Number	1
Test Objective	Verify the TOE is able to create a trust chain using CA's in the TOE trust store.
Test Steps Performed	Configure the test server to not present the full certificate chain to the TOE, so that TOE can use certificates from the trust store to validate the certificate. Configure TOE to connect to test server. Verify that the Test Server is configured: Establish a connection from the TOE. Verify that the connection is established.
Test Result	Pass; The TOE created a trust chain using the CA's in the TOE trust store.

Test Number	2
Test Objective	This test is resolved by testing in FCS_TLSC_EXT.1.2 and FIA_X509_EXT.1/Rev and [ST] FCS_TLSC_EXT.1.3 does not define override mechanisms for the TOE.
Test Steps Performed	None
Test Result	Pass; The TOE was tested in FCS_TLSC_EXT.1.2.

Test Number	3
Test Objective	This test is not applicable, override mechanisms are not defined in [ST] FCS_TLSC_EXT.1.3.
Test Steps Performed	None
Test Result	Pass; [ST] does not define override mechanisms.

FCS_TLSC_EXT.1.4

Test Number	1
Test Objective	Verify that the TOE supports EC/ DHE groups as defined in [ST]. [ST] states the TOE supports only EC secp384r1.
Test Steps Performed	Establish the connection between the TOE and syslog server. Inspect the capture. Verify that the Connection is established using secp384r1 curve.
Test Result	Pass; The TOE supports the defined EC groups.

2.19 FCS_TLSC_EXT.2 Extended: TLS Client Protocol for Mutual Authentication (Optional)

TSS

FCS_TLSC_EXT.2.1

The evaluator shall ensure that the TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Guidance Documentation

FCS_TLSC_EXT.2.1

If the TSS indicates that mutual authentication using X.509v3 certificates is used, the evaluator shall verify that the AGD guidance includes instructions for configuring the client-side certificates for TLS mutual authentication.

Tests

For all tests in this chapter the TLS server used for testing of the TOE shall be configured to require mutual authentication.

FCS_TLSC_EXT.2.1

(covered by FCS_TLSC_EXT.1.1 Test 1 and testing for FIA_X.509_EXT.*).

2.19.1 TSS

[ST] Section 7.2.7 describes that the TOE will provide a client-side certificate when requested by the remote entity. [ST] Section 7.2.6 describes that the TOE will sign a Certificate Verify message with its client certificate to verify itself to the service.

2.19.2 Guidance Documentation

[ST] indicates that mutual authentication using x509 certificates is used.

[AGD] Section 6 describes how the TOE certificate and private key can be configured in the TOE.

2.19.3 Tests

Test Number	1
Test Objective	Verify the TOE provides required fields to perform mutual Auth in response to the Certificate Request from the test server.
Test Steps Performed	Proceed with the configured remote syslog server. Start syslog service on the TOE to initiate the connection. Connection is initiated.
Test Result	Pass; The TOE supports mutual authentication.

2.20 FCS_TLSS_EXT.1 Extended: TLS Server Protocol without Mutual Authentication (Selection-Based)

TSS

FCS_TLSS_EXT.1.1

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

FCS_TLSS_EXT.1.2

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

FCS_TLSS_EXT.1.3

If using ECDHE or DHE ciphers, the evaluator shall verify that the TSS describes the key agreement parameters of the server Key Exchange message.

FCS_TLSS_EXT.1.4

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

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

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

Guidance Documentation

FCS_TLSS_EXT.1.1

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

FCS_TLSS_EXT.1.2

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

FCS_TLSS_EXT.1.3

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

Tests

FCS_TLSS_EXT.1.1

Test 1: The evaluator shall establish a TLS connection using each of the ciphersuites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an HTTPS session. It is sufficient to observe the successful negotiation of a ciphersuite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server’s ST and verify that the server denies the connection.

Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

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

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

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

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

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

FCS_TLSS_EXT.1.2

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

FCS_TLSS_EXT.1.3

Test 1: [conditional] If ECDHE ciphersuites are supported:

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

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

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

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

FCS_TLSS_EXT.1.4

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2.20.1 TSS

FCS_TLSS_EXT.1.1
[ST] Section 7.2.8 describes that the TOE is using a TLSv1.2 server according to RFCs 4492, 5246, 5289, and 6125. The TOE supports the following ciphersuite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384. This is consistent with the selections in the FCS_TLSS_EXT.1.1.

FCS_TLSS_EXT.1.2
[ST] Section 7.2.8 describes that TOE is rejecting connections with TLS version 1.1 or older and SSLv3.0 and older.

FCS_TLSS_EXT.1.3
[ST] Section 7.2.8 describes that the TOE uses secp384r1 elliptic curve for ECDHE.

FCS_TLSS_EXT.1.4
[ST] Section 6.1.2.14 FCS_TLSS_EXT.1 states TSF supports no session resumption or session tickets.
[ST] Section 7.2.3 states TLS server does not support session resumption.

2.20.2 Guidance Documentation

FCS_TLSS_EXT.1.1
[ST] Section 7.2.8 describes that the TLS client supports TLS v1.2 and the TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 ciphersuite with a secp384r1 curve. It describes that ciphersuite and curves are not configurable.

[AGD] Section 7.1 describes that the TOE does not provide configuration for those TLS parameters.

FCS_TLSS_EXT.1.2
[AGD] Section 7.1 describes that the TOE does not provide configuration for those TLS parameters.

FCS_TLSS_EXT.1.3
[AGD] Section 7.1 describes that the TOE does not provide configuration for those TLS parameters.

[ST] Section 7.2.8 describes that the ECDHE agreement parameters are not configurable.

FCS_TLSS_EXT.1.4
[AGD] Section 7.1 describes that the TOE does not provide configuration for those TLS parameters.

2.20.3 Tests

FCS_TLSS_EXT.1.1

Test Number	1
Test Objective	Verify the ciphersuite(s) claimed in [ST] can be negotiated. [ST] claims one ciphersuite — TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 with NIST P-384 curve.
Test Steps Performed	Configure the syslog client to establish a connection to the TOE using ECDHE_ECDSA_WITH AES_256_GCM_SHA384 ciphersuite. Verify that the Connection is established.
Test Result	Pass; The TOE supports the claimed ciphersuite and negotiates connections with it.

Test Number	2
Test Objective	Verify the TOE denies a connection when using TLS_NULL_WITH_NULL_NULL ciphersuite and with not-supported ciphersuite.
Test Steps Performed	Attempt a connection to the TOE with TLS_NULL_WITH_NULL_NULL ciphersuite using the UL s2n test tool: /test-tools/tls2y/s2nc 10.3.2.100:443 1-2 Verify that the Connection is rejected. Attempt a connection to the TOE with RSA-based ciphersuites using the openssl tool: openssl s_client -connect 10.3.2.100:443 -cipher RSA Verify that the connection is rejected.
Test Result	Pass; The TOE denied connections using the NULL and RSA ciphersuites.

Test Number	3
Test Objective	Verify the TOE rejects a connection with a modified client finished handshake message and does not send any unencrypted data.
Test Steps Performed	Attempt a connection with a modified client finished handshake message using the UL s2n test tool: /test-tools/tls2y/s2nc 10.3.2.100:443 1-4c Verify that the connection is denied. Establish a client session to the TOE using openssl tool. Inspect the packet capture. Verify that the connection is established, application data is encrypted. Inspect Server Finished message and compare with decrypted client data. Verify that the Finished message is encrypted.
Test Result	Pass; The TOE rejected connections which use modified client data in the handshake finished message.

FCS_TLSS_EXT.1.2

Test Number	1
Test Objective	Verify the TOE rejects protocol versions: SSL 2.0, SSL 3.0, TLS 1.0, TLS

	1.1.
Test Steps Performed	Connect to the TOE using each TLS version the UL s2n test tool: s2nc 10.3.2.100:443 2-td0156 Verify that connections using SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1 are rejected. Inspect the packet capture. Verify that Connections using SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1 are rejected.
Test Result	Pass; The TOE rejects the unsupported protocol versions.

FCS_TLSS_EXT.1.3

Test Number	1
Test Objective	Verify the TOE establishes the connection using a specified curve (secp384r1) and rejects the connection using unsupported curves.
Test Steps Performed	Connect to the TOE using a TLS client. Inspect the packet capture. Verify that the TOE establishes a connection using ECDHE and P-384. Establish a connection using unsupported EC curve (secp256) using openssl tool: openssl s_client -connect 10.3.2.100:443 -msg -curves prime256v1 Verify that the Connection is rejected.
Test Result	Pass; The TOE established TLS connections using only the specified curve.

Test Number	2
Test Objective	The TOE does not support DHE ciphersuites.
Test Steps Performed	None
Test Result	Pass; The TOE does not support DHE.

Test Number	3
Test Objective	The TOE does not support RSA ciphersuites.
Test Steps Performed	None
Test Result	Pass; The TOE does not support RSA.

FCS_TLSS_EXT.1.4

Test Number	1
Test Objective	Demonstrate the TOE does not support session resumption.
Test Steps Performed	Establish a TLS connection from a client using a zero-length session identifier, and zero-length session ticket. Verify that the Session is established. Verify that a New Session ticket is never sent by the TOE at any point of handshake in packet capture. Verify that The TOE does not send a new session ticket at any point of the handshake. Check Server hello message. Verify that the Server Hello message contains a zero-length session identifier.

Test Result	Pass; The TOE did not support session resumption.

Test Number	2
Test Objective	Demonstrate the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption).
Test Steps Performed	[ST] does not claim support for session resumption; this test is not applicable.
Test Result	Pass; The TOE does not support session resumption.

Test Number	3
Test Objective	Demonstrate the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption).
Test Steps Performed	[ST] does not claim support for session resumption; this test is not applicable.
Test Result	Pass; The TOE does not support session resumption.

2.21 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).

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.

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.

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).

   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.

2.21.1 TSS

[ST] Section 7.3.1 describes that each successive unsuccessful authentication is tracked using a counter that is reset on a successful login. It states that the remote administrator is prevented from logging into the TOE after a configured limit is reached. The TOE restores the ability for a remote administrator to log in after the configurable time interval elapses.

[ST] Section 7.3.1 describes that local console login is not subject to lock out, so there is no situation where no administrative access to the TOE is available.

2.21.2 Guidance Documentation

[AGD] Section 7.5 describes how a number of failed login attempts can be configured and how the remote management interface lockout time is configured. It contains a notice that access through the local console is always available.

2.21.3 Tests

Test Number	1
Test Objective	Verify the TOE does not grant access after a set number of failed authentication attempts.
Test Steps Performed	Configure the number of unsuccessful login attempts using the REST API to be "4". Verify that the TOE is configured. Configure the incorrect login limit lockout via the REST API to be 180 sec. Verify that the TOE is configured. Verify that the TOE does not allow access after a set number of unsuccessful authentication attempts. Verify that the TOE does not allow a login with valid credentials after the failed authentication limit is reached.
Test Result	Pass; The TOE denied authentication after a configured number of failed authentication attempts.

Test Number	2
Test Objective	Verify the TOE does not allow access within a set lockout time after exceeding set authentication login failures.
Test Steps Performed	Attempt a login prior to the lockout time period being exceeded.

	Verify that Login is rejected. Attempt a login right after the lockout time period is exceeded. Verify that Login is accepted.
Test Result	Pass; The TOE enforced the lockout time.

2.22 FIA_PMG_EXT.1 Password Management

TSS

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

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.

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.

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.

2.22.1 TSS

[ST] Section 7.3.2 contains a list of special characters, supported in a password, it describes that minimal password length can be configured to be between 6 and 20 characters.

2.22.2 Guidance Documentation

[AGD] Section 4.1.2 describes what characters the password can compose of.

[AGD] Section 4.1.3 describes that the minimal password length is configurable between 6 and 20 characters. [AGD] Section 8.3 describes how this minimal length can be configured.

[AGD] Section 4.1.3 contains a note with the recommendation of a strong password composition.

2.22.3 Tests

Test Number	1
Test Objective	Verify the TOE is able to set a minimum length for the password between 6 and 20 characters and is able to accept any combination of upper- and

	lower-case letters, numbers, and special characters.
Test Steps Performed	Configure the TOE minimum accepted password length to be 8 characters. Verify that the TOE is configured. Configure the password to be “PASS!@#$%^&*()”. Verify that the Password is accepted. Configure the password to be “pass12345678990”. Verify that the Password is accepted.
Test Result	Pass; The TOE allowed configuration of passwords using the specified complexity factors.

Test Number	2
Test Objective	Verify the TOE does not support passwords not meeting the password policy.
Test Steps Performed	Configure the password to be empty. Verify that the Password is rejected. Configure the password to be “newpass” (shorter than a configured minimum length). Verify that the Password is rejected. Configure the password to be “passw[]rd”. Verify that the Password is rejected.
Test Result	Pass; The TOE correctly enforced the complexity requirement.

2.23 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”.

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

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

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

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.

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:

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

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

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

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

2.23.1 TSS

[ST] Section 7.3.3 describes that the TOE supports remote administrative sessions using TLS/HTTPS and local sessions using local console.

[ST] Section 7.3.3 describes that, for local administrative sessions, authentication is performed by providing a username and password at the logon prompt. Successful authentication is denoted by obtaining a command prompt. No administrative actions are permitted prior to authentication.

[ST] Section 7.3.3 describes that remote administrative session authentication is performed using username and password. A successful authentication is denoted by obtaining an authentication token for the RMI interface.

[ST] Section 7.3.3 describes that the following actions are allowed for users, or non-TOE entities, prior to authentication:

- Display the warning banner in accordance with FTA_TAB.1
- Respond to ARP requests with ARP replies
- Automated generation of cryptographic keys for the TLS Server and TLS Client TSF
- Packet forwarding through the IPsec tunnel
- Packet forwarding through BYPASS packet filtering table
- ICMP echo reply (when configured in packet filtering table by the SA)

The TOE is not a distributed TOE.

2.23.2 Guidance Documentation

[AGD] Section 4.1 describes that the local console interface is accessible through the hypervisor.

[AGD] Section 4.1 describes that the remote management interface can be accessed from the local network where the Management network interface is provisioned. It describes how this interface can be provisioned and its IP address and subnet can be configured. [AGD] Section 3.1 describes that built-in certificates are installed, and that the device can be accessible without certificate configuration. [AGD] Section 5.1 describes how new certificates can be created and installed.

[AGD] Section 4.1.1 describes how security administrators can login using the remote management interface.

2.23.3 Tests

Test Number	1
Test Objective	Verify the TOE allows local and remote access using appropriate login credentials and does not allow access for incorrect login credentials.
Test Steps Performed	Attempt a local login using username “admin” and an incorrect password. Verify that the Login is rejected. Attempt a local login using username “admin<TAB>1” and the correct password. Verify that the Login is rejected. Attempt a local login using incorrect username “administrator” and the correct password. Verify that the Login is rejected. Attempt a local login using incorrect username “administrator” and the correct password. Verify that the Login is rejected. Attempt a local login using username “admin” and the correct password. Verify that the Login is accepted. Attempt a remote login using username “admin” and an incorrect password. Verify that the Login is rejected. Attempt a remote login using username “admin<TAB>” and the correct password. Verify that the Login is rejected. Attempt a remote login using username “administrator” and the correct password. Verify that the Login is rejected. Attempt a login using username “admin” and the correct password. Verify that the Login is accepted.
Test Result	Pass; The TOE correctly processed passwords and authentication.

Test Number	2
Test Objective	Verify that only claimed services are available to an external entity. Services claimed in [ST] are: • Display warning banner (resolved by testing FTA_TAB.1) • automated generation of cryptographic keys (Resolved by testing FCS_TLSS_EXT.1, FCS_IPSEC_EXT.1),

	• respond to ARP requests with ARP replies • packet forwarding through the Ipsec tunnel (Resolved by testing FCS_TLSS_EXT.1, FCS_IPSEC_EXT.1), • packet forwarding through BYPASS packet filtering table (Resolved by testing FCS_TLSS_EXT.1, FCS_IPSEC_EXT.1), • ICMP echo reply (when configured in packet filtering table by the SA)
Test Steps Performed	Conduct a network scan to identify external services using NMAP. Services listed in [ST] are available. Configure a TOE firewall to allow ping requests and replies. Verify that the TOE is configured. Perform a TOE ping and initiate packet capture. Verify that the TOE responds to ping and ARP requests. Investigate available ports to ensure no services are available to unauthenticated users except for providing a warning banner: Access TOE on a port 443 via management network, and Use Postman tool to perform API calls without prior authentication: Verify that no services are provided except for the login banner and login operation.
Test Result	Pass; The TOE only offered the listed services prior to authentication.

Test Number	3
Test Objective	Verify that no services are available through the local console prior to authentication.
Test Steps Performed	Open the local console. Determine services available to local administrator. No services available.
Test Result	Pass; The TOE did not allow access to services from the local console.

Test Number	4
Test Objective	The TOE is not a distributed TOE. This test is not applicable.
Test Steps Performed	None
Test Result	Pass; The TOE is not distributed.

2.24 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.

2.25 FIA_UAU.7 Protected Authentication Feedback

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.

Tests

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

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

2.25.1 TSS

None.

2.25.2 Guidance Documentation

No configuration to enable obscured feedback during local login is required.

2.25.3 Tests

Test Number	1
Test Objective	Verify the TOE obscures credentials on input.
Test Steps Performed	Use the local console to log into the TOE. Password entry is obscured.
Test Result	Pass, The TOE obfuscated user input.

2.26 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.

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.

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.)

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).

2.26.1 TSS

[ST] Section 7.3.6 describes that the TOE checks the validity of certificates in the following events:

- When CA is imported on the TOE
- When a signed CSR is imported/installed on the TOE
- When the TSF receives an X.509 certificate from a remote TLS Server peer
- When the TSF receives an X.509 certificate from a remote IPsec peer
- When the TOE Trusted Update functionality is executes

This corresponds to every authentication attempt that TOE performs, and the trusted update functionality of the TOE.

[ST] Section 7.3.6 describes extendedKeyUsage verification rules of the TOE.

[ST] Section 7.3.6 describes that the certificate revocation happens on every certificate validation event, and it is performed for every certificate in the chain of trust, except for the root certificate.

[ST] Section 7.3.6 describes that if peer does not provide intermediate certificates to complete trust chain and the TOE does not have corresponding CA installed in the trust store, the connection is rejected.

2.26.2 Guidance Documentation

[AGD] Section 5.1 describes that the TOE verifies the remote syslog server certificate when establishing a connection.

[AGD] Section 6.2 describes that the CA certificates are validated upon import to the TOE trust store.

[AGD] Section 6.1.1 describes that the device certificate is validated upon import to the TOE trust store.

[AGD] Section 6.3 describes that a certificate revocation check is performed using CRL distribution points when a new certificate is loaded into the device, or when establishing a connection to external entity. This section describes how validation takes place and that the certificate is accepted as valid if revocation information is not acceptable.

[AGD] Section 8.9 describes that the certificate revocation check using the CRL distribution point is performed on a certificate in the trusted update mechanism. This section describes how validation takes place and that the certificate is rejected if revocation information is not acceptable.

[ST] does not describe any required extendedKeyUsage fields not supported or trivially satisfied.

2.26.3 Tests (TLS Client)

Test Number	1
Test Objective	Verify certificate validation will succeed if a chain is full. Verify certificate validation will fail if chain of trust not established.
Test Steps Performed	Remove all CAs from the syslog trust store in the TOE, add Root CA to the TOE trust store. Verify that the TOE is configured. Configure the syslog server to present a full certificate chain. Start syslog service to establish a connection. Verify that the Connection is established. Reconfigure the test syslog server to present only leaf certificate to the TOE. Establish a connection. Verify that the connection is rejected.
Test Result	Pass; The TOE validated certificates correctly.

Test Number	2
Test Objective	Verify the TOE rejects an expired certificate.
Test Steps Performed	Create an expired certificate. Verify that the certificate is created. Configure the test server to use the expired certificate. Initiate a connection from the TOE. Verify that the connection is rejected.
Test Result	Pass; The TOE rejected expired certificates.

Test Number	3
Test Objective	Verify the TOE correctly performs revocation checking on a leaf cert and intermediate CA.
Test Steps Performed	Configure a CRL server to verify an Intermediate CA and leaf certificate. Set revoked status for Intermediate CA. Verify that the server is configured. Continue with the TOE from Test 1, without the Intermediate CA from the trust root storage of the TOE, so that Intermediate CA is provided by the server. Start a service to establish a connection. Connection to CRL server is established. Verify that the TOE rejects the connection. Set Revoked status for peer leaf certificate. Start a service to establish a connection. Verify that the connection to the CRL server is established. The TOE rejects the connection. Present the TOE with the expired CRL. Start a service to establish a connection. Verify that the connection to the CRL server is established. The TOE rejects the connection.
Test Result	Pass; The TOE correctly performed revocation checking.

Test Number	4

Test Objective	Verify the TOE correctly performs revocation checking when the CRL is signed by CA without CLRSign key usage field. The TOE is expected to not accept a connection, according to the [ST] statement that if CRL is invalid, this is treated as validation failure. OCSP not selected.
Test Steps Performed	Configure the TOE to not have intermediate CA in a trust store. Configure the test server to present an intermediate CA without a CRLSign. Configure CRL server to present empty CRLs. Server configured. Start a service to establish a connection.Connection fails.
Test Result	Pass; The TOE rejected the incorrectly signed CRL.

Test Number	5
Test Objective	Verify the TOE rejects a malformed certificate.
Test Steps Performed	Use test tool to present test certificate with the modified first 8 bytes to the TOE. Cause the TOE to initiate the connection. Certificate is presented. Connection is rejected.
Test Result	Pass; The TOE did not accept the improper certificate

Test Number	6
Test Objective	Verify the TOE rejects a certificate with a corrupted signature.
Test Steps Performed	Create a certificate with a modified byte in the Signature value field. Certificate is created.
Test Result	Pass; The TOE rejected corrupted certificate signatures.

Test Number	7
Test Objective	Verify the TOE rejects certificate with a modified public key.
Test Steps Performed	Create a certificate with a modified public key (ending bytes in the public key value replaced with FF). Verify that a Certificate with a corrupted public key is expected. Configure the test server to use this certificate: /home/ulvs# /test-tools/uls2n/s2nd --ekey certsR2/leaf0.key --ecert certsR2/syslog.local-test7.pem 10.3.2.4:5514 1-1 Verify that the Server is configured. Cause the TOE to initiate a connection. Verify that The TOE rejects the connection.
Test Result	Pass; The TOE rejected the corrupted certificates.

Test Number	8a
Test Objective	Verify the TOE supports certificate chain with named curves installed when presented as part of a certificate message.
Test Steps Performed	Configure the syslog server to present a full certificate chain from outside of the TOE. Configure the TOE to contain only root certificate in the trust store. Establish a connection. Verify that the Connection is established as expected.

Test Result	Pass; The TOE supported named curves.

Test Number	8b
Test Objective	Verify the CA certificate with an explicitly defined curve is not accepted by the TOE when provided externally.
Test Steps Performed	Configure the syslog server to present a certificate with an explicitly defined curve as part of handshake from outside of the TOE. Syslog server is configured. Establish a connection. Connection is rejected.
Test Result	Pass; The TOE rejected the certificate with explicitly defined curve.

Test Number	8c
Test Objective	Verify that theTOE will load a certificate with a named curve into the trust store (Resolved by Testing FIA_X509_EXT.3.2 – Test 2). Verify the TOE will not load a certificate with an explicitly defined curve in the trust store.
Test Steps Performed	Continue with the certificate created on previous step. Verify that a Certificate with explicitly defined curve is generated. Try to import a certificate with explicitly defined curves to the TOE trust store. Verify that certificate import is rejected.
Test Result	Pass; The TOE rejected the certificate with explicitly defined curve.

Test Number	FIA_X509_EXT.1.2 test 1
Test Objective	Verify the TOE rejects a certificate without the BasicConstains extension when loaded to the trust store, when presented in a trust chain.
Test Steps Performed	Continue from the TOE state when the TOE has a root CA uploaded successfully. Create an Intermediate CA that does not have basicConstraints extension. Intermediate CA is Try to import the created certificate to the TOE using Rest API. Import is rejected.
Test Result	Pass; The TOE rejected the certificate without the BasicConstains extension.

Test Number	FIA_X509_EXT.1.2 test 2
Test Objective	Verify the TOE rejects a certificate with the CA flag set to FALSE when loaded to trust store, when presented in a trust chain.
Test Steps Performed	Create an Intermediate CA that has CA=False in the BasicConstraints extension. Verify that the certificate is created. Try to import the created certificate to the TOE. Verify that Import is rejected. Try to present this certificate to the TOE in a certificate message. Verify that the connection is rejected.
Test Result	Pass; The TOE rejected the certificate without the CA=True field.

2.26.4 Tests (TLS Server)

Test Number	1
Test Objective	Verify the certificate validation will fail if chain of trust is not established.
Test Steps Performed	This is resolved in testing the import of the certificate to the Trust store in FIA_X509_EXT.3.2-Test 2, because the TOE validates a certificate only on import.
Test Result	Pass

Test Number	2
Test Objective	Verify the TOE rejects an expired certificate.
Test Steps Performed	Create a certificate with the expiration date in the past. Verify that the certificate is created. Attempt to import the certificate. Verify that the import is rejected.
Test Result	Pass; The TOE rejected expired certificates.

Test Number	3
Test Objective	Verify the TOE correctly performs revocation checking on a leaf cert and intermediate CA. Attempt to import the certificate. Verify that the Import is rejected.
Test Steps Performed	Configure the CRL server to revoke an intermediate CA. Verify that the Server is configured. Attempt to import a correct leaf certificate. Verify that the import fails. Create CRL with a revoked leaf certificate, configure CRL server to present it, and empty CRL for Intermediate CA. Verify that CRL server is configured. Attempt to load a leaf certificate with a SN 67E837BEE75269C7EBD3C5F8828FF13532BF8B3A to the TOE. Verify that the import is rejected. Reset CRL server to present empty CRL lists. Verify CRL server is configured. Attempt to import a correct leaf certificate. Verify Import is successful.
Test Result	Pass; The TOE correctly performed revocation checking.

Test Number	4
Test Objective	Verify the TOE correctly performs revocation checking when the CRL is signed by CA without CLRSign key usage field. The TOE is expected to not accept a connection, according to the [ST] statement that if CRL is invalid, this is treated as validation failure. OCSP not selected.
Test Steps Performed	Configure the TOE to have an intermediate CA with no CRLSign usage field. Verify that the Certificate is created and certificate is uploaded.

	Configure the CRL server to present a CRL signed by this CA, revoking leaf certificate. Verify that the CRL server is configured. Import leaf certificate to the TOE. Verify that the revocation check fails and the certificate is accepted.
Test Result	Pass; The TOE rejected the incorrectly signed CRL.

Test Number	5
Test Objective	Verify the TOE rejects a malformed certificate.
Test Steps Performed	Modify a byte in the first 8 bytes of a meaningful section of the test server certificate. Verify that the certificate is created. Import this certificate into the TOE. Verify that the import is rejected.
Test Result	Pass; The TOE did not accept the improper certificate.

Test Number	6
Test Objective	Verify the TOE rejects a certificate with a corrupted signature.
Test Steps Performed	Create a certificate with a modified byte in the Signature value field. Verify that the certificate is created. Import this certificate to the TOE. Verity that the import is rejected.
Test Result	Pass; The TOE rejected corrupted certificate signatures.

Test Number	7
Test Objective	Verify the TOE rejects a certificate with a modified public key.
Test Steps Performed	Modify a byte in the public key of a leaf certificate before importing it into the TOE. Verify that the Modified certificate is created. Import this certificate into the TOE. Verify that the import is rejected.
Test Result	Pass; The TOE rejected the corrupted certificates.

Test Number	8a
Test Objective	Verify the TOE supports certificate chain with named curves installed when presented as part of a certificate message.
Test Steps Performed	This test not applicable as the TLS server does not accept certificates externally.
Test Result	Pass; The TOE supported named curves.

Test Number	8b
Test Objective	Verify the CA certificate with an explicitly defined curve is not accepted by the TOE when provided externally.
Test Steps Performed	This test not applicable as the TLS server does not accept certificates externally.

Test Result	Pass; The TOE rejected the certificate with explicitly defined curve.

Test Number	8c
Test Objective	Verify that the TOE will load a certificate with a named curve into the trust store (Resolved by Testing FIA_X509_EXT.3.2 – Test 2). Verify the TOE will not load a certificate with an explicitly defined curve in the trust store.
Test Steps Performed	Create an Intermediate CA certificate with the explicitly defined curve. Verify that the Certificate is created. Try to import a certificate with an explicitly defined curves to the TOE trust store. Verify that the import is rejected.
Test Result	Pass; The TOE rejected the certificate with explicitly defined curve.

Test Number	FIA_X509_EXT.1.2 test 1
Test Objective	Verify the TOE rejects a certificate without the BasicConstains extension when loaded to the trust store, when presented in a trust chain.
Test Steps Performed	Continue from the TOE state when the TOE has a root CA uploaded successfully. Create an Intermediate CA that does not have basicConstraints extension. Verify that the certificate is created. Try to import the created certificate to the TOE using Rest API. Verify that the import is rejected.
Test Result	Pass; The TOE rejected the certificate without the BasicConstains extension.

Test Number	FIA_X509_EXT.1.2 test 2
Test Objective	Verify the TOE rejects a certificate with the CA flag set to FALSE when loaded to trust store, when presented in a trust chain.
Test Steps Performed	Create an Intermediate CA that has CA=False in the BasicConstraints extension. Verify that the Certificate is created. Try to import the created certificate to the TOE. Verify that the Import is rejected.
Test Result	Pass; The TOE rejected the certificate without the CA=True field.

2.26.5 Tests (IPsec)

Test Number	1
Test Objective	Verify the certificate validation will fail if chain of trust is not established.
Test Steps Performed	Configure the TestGW to contain a chain of two certificates. Verify that TestGW is configured. Establish a connection. Verify that the connection is established. Remove root of trust from the TOE and remove the intermediate CA from the TestGW.

	Verify that the TestGW is configured to contain only root CA, root of trust is removed from the TOE. Establish a connection. Verify that the Connection is not established.
Test Result	Pass; the TOE correctly establishes the connection when the certificate chain is complete and correct, and does not establish the connection if any part of the certificate chain is missing.

Test Number	2
Test Objective	Verify the TOE rejects an expired certificate.
Test Steps Performed	Create an expired certificate. Configure the TestGW to use this certificate. Verify that the certificate is created and server is configured. Verify that the server is configured. Establish a connection. Verify that connection is rejected.
Test Result	Pass; The TOE rejected the expired certificates.

Test Number	3
Test Objective	Verify the TOE correctly performs revocation checking on a leaf cert and intermediate CA. Attempt to import the certificate. Verify that the import is rejected.
Test Steps Performed	Configure the CRL server to revoke an intermediate CA. Verify that the server is configured. Attempt to import a correct leaf certificate. Verify that Import fails. Create CRL with a revoked leaf certificate, configure CRL server to present it, and empty CRL for Intermediate CA. Verify that CRL server is configured. Attempt to load a leaf certificate with a SN 67E837BEE75269C7EBD3C5F8828FF13532BF8B3A to the TOE. Verify that import is rejected. Reset CRL server to present empty CRL lists. Verify CRL server is configured. Attempt to import a correct leaf certificate. Verify import is successful.
Test Result	Pass; The TOE correctly performed revocation checking.

Test Number	4
Test Objective	Verify the TOE correctly performs revocation checking when the CRL is signed by CA without CLRSign key usage field. The TOE is expected to not accept a connection, according to the [ST] statement that if CRL is invalid, this is treated as validation failure. OCSP not selected.
Test Steps Performed	Continue with the configuration from Test 3. Create a CA with the same

	key, but without CRL sign properties. Verify that the certificate is created. Overwrite the normal Int CA certificate with the created certificate on the TOE and TestGW. Verify that the certificate is overwritten. Establish a connection. Verify that the CRL is requested and ignored; connection is established.
Test Result	Pass; The TOE performed revocation checking.

Test Number	5
Test Objective	Verify the TOE rejects a malformed certificate.
Test Steps Performed	Modify a byte in the first 8 bytes of a meaningful section of the test server certificate. Verify that the certificate is created. Import this certificate into the TOE. Verify that the import is rejected.
Test Result	Pass; The TOE did not validate the improper certificate.

Test Number	6
Test Objective	Verify the TOE rejects a certificate with a corrupted signature.
Test Steps Performed	Create a certificate with a modified byte in the Signature value field. Verify that the certificate is created. Import this certificate to the TOE. Verify that the import is rejected. Configure TestGW to provide a certificate with a corrupted signature to the TOE using UL test tool. Initiate a connection. Verify that TestGW is configured. Connection is rejected.
Test Result	Pass; The TOE rejected corrupted certificate signatures.

Test Number	7
Test Objective	Verify the TOE rejects certificate with a modified public key.
Test Steps Performed	Modify a byte in the public key of a leaf certificate before importing it into the TOE. Verify that the modified certificate is created. Import this certificate into the TOE. Verify that the import is rejected. Configure TestGW to provide a certificate with a modified public key to the TOE using UL test tool. Verified that the modified certificate sent during IKE, Connection is not established.
Test Result	Pass; The TOE processed the certificates correctly.

Test Number	8a
Test Objective	Verify the TOE supports certificate chain with named curves installed when presented as part of a certificate message.
Test Steps Performed	Configure TestGW server to present a certificate chain with the named curve.

	Verify that the Server is configured. Remove an Intermediate CA from the TOE IPsec trust store. Establish a connection. Verify that the connection is established. Remove Intermediate CA from the TOE trust store, so that the TOE will use the Intermediate CA with the named curve provided by an external peer to validate the leaf certificate. Verify that the TOE is configured. Establish a connection. Verify that the connection is established.
Test Result	Pass; The TOE supported named curves.

Test Number	8b
Test Objective	Verify the CA certificate with an explicitly defined curve is not accepted by the TOE when provided externally.
Test Steps Performed	Configure TestGW server to present a certificate chain with the explicitly defined curve. Verify that the Server is configured. Establish a connection. Verify that the connection is rejected.
Test Result	Pass; The OE rejected certificate with explicitly defined curve.

Test Number	8c
Test Objective	Verify that the TOE will load a certificate with a named curve into the trust store (Resolved by Testing FIA_X509_EXT.3.2 – Test 2). Verify the TOE will not load a certificate with an explicitly defined curve in the trust store.
Test Steps Performed	Generate an Intermediate CA Cert with explicitly defined curve parameters. Verify that Certificate is generated. Try to import a certificate with explicitly defined curves to the TOE trust store. Verify that Import is rejected.
Test Result	Pass; The TOE rejected certificates correctly.

Test Number	FIA_X509_EXT.1.2 test 1
Test Objective	Verify the TOE rejects a certificate without the BasicConstains extension when loaded to the trust store, when presented in a trust chain.
Test Steps Performed	Continue from the TOE state when the TOE has a root CA uploaded successfully. Create an Intermediate CA that does not have basicConstraints extension. Verify that the certificate is created. Try to import the created certificate to the TOE using Rest API. Verify that the import is rejected. Configure TestGW to present this certificate as part of the IPsec handshake.

	Verify that the Server is configured.
Test Result	Pass; The TOE rejected certificate without the BasicConstains extension.

Test Number	FIA_X509_EXT.1.2 test 2
Test Objective	Verify the TOE rejects a certificate with the CA flag set to FALSE when loaded to trust store, when presented in a trust chain.
Test Steps Performed	Create an Intermediate CA that has CA=False in the BasicConstraints extension. Verify that the Certificate is created. Try to import the created certificate to the TOE. Verify that the import is rejected. Configure TestGW to present this certificate as part of the handshake. Verify that the server is configured. Establish a connection from the TOE. Connection is rejected.
Test Result	Pass; The TOE rejected certificate without the CA=True field.

2.26.6 Tests (Trusted Update)

Test Number	1
Test Objective	Verify certificate validation will succeed if a chain is full. Verify certificate validation will fail if chain of trust is not established.
Test Steps Performed	Attempt to update the TOE using an image signed by a certificate with a correct chain. Verify that the update is successful. Attempt an upgrade using an image signed by a certificate with a broken chain of trust. Verify that the update is rejected.
Test Result	Pass; The TOE validated certificates correctly.

Test Number	2
Test Objective	Verify the TOE rejects an expired certificate.
Test Steps Performed	Change the system date on the TOE to be after the expiration date of the certificate used to sign update image. Verify that the TOE date changed. Attempt to update using a normal update image. Verify that the update is not accepted, verify that the certificate not accepted.
Test Result	Pass; The TOE rejected expired certificates.

Test Number	3
Test Objective	Verify the TOE correctly performs revocation checking on a leaf cert and intermediate CA.
Test Steps Performed	Configure the CRL server to host a CRL that revokes a certificate in an update image. Verify that the CRL server is configured.

	Attempt to update the server with update files containing the revoked certificate. Update is rejected.
Test Result	Pass; The TOE correctly performed revocation checking.

Test Number	4
Test Objective	Verify the TOE correctly performs revocation checking when the CRL is signed by CA without CLRSign key usage field. The TOE is expected to not accept a connection, according to the [ST] statement that if CRL is invalid, this is treated as validation failure. OCSP not selected.
Test Steps Performed	The TOE does not support modifying or supplying external Cas during a trusted update, so this test is not applicable.
Test Result	Pass; The TOE rejected incorrectly signed CRL.

Test Number	5
Test Objective	Verify the TOE rejects a malformed certificate.
Test Steps Performed	Use test tool to present test certificate with the modified first 8 bytes to the TOE. Cause the TOE to initiate the connection. Verify that Certificate is presented. Verify Connection is rejected.
Test Result	Pass; The TOE did not validate the improper certificate.

Test Number	6
Test Objective	Verify the TOE rejects a certificate with a corrupted signature.
Test Steps Performed	Create a certificate with a modified byte in the Signature value field. Certificate is created.
Test Result	Pass; The TOE rejected corrupted certificate signatures.

Test Number	7
Test Objective	Verify the TOE rejects a certificate with a modified public key.
Test Steps Performed	Create a certificate with a modified public key (ending bytes in the public key value replaced with FF). Verify that a Certificate with a corrupted public key is expected. Configure the test server to use this certificate: /home/ulvs# /test-tools/uls2n/s2nd –ekey certsR2/leaf0.key –ecert certsR2/syslog.local-test7.pem 10.3.2.4:5514 1-1 Verify that the server is configured. Cause the TOE to initiate a connection. Verify that the TOE rejects the connection.
Test Result	Pass; The TOE rejected corrupted certificate.

Test Number	8a
Test Objective	Verify the TOE supports certificate chain with named curves installed when presented as part of a certificate message.

Test Steps Performed	The TOE does not have a configurable trust store, nor does it expose hardcoded Cas for the update process, nor does it allow Cas embedded in the update image. This test is not applicable.
Test Result	Pass; The TOE supported named curves.

Test Number	8b
Test Objective	Verify the CA certificate with an explicitly defined curve is not accepted by the TOE when provided externally.
Test Steps Performed	The TOE does not have a configurable trust store, nor does it expose hardcoded Cas for the update process, nor does it allow Cas embedded in the update image. This test is not applicable.
Test Result	Pass; The TOE rejected the certificate with explicitly defined curve.

Test Number	8c
Test Objective	Verify the TOE will not load a certificate with an explicitly defined curve in the trust store.
Test Steps Performed	The TOE does not have a configurable trust store, nor does it expose hardcoded Cas for the update process, nor does it allow Cas embedded in the update image. This test is not applicable.
Test Result	Pass; The TOE rejected the certificate with explicitly defined curve.

Test Number	FIA_X509_EXT.1.2 test 1
Test Objective	Verify the TOE rejects a certificate without the BasicConstains extension when loaded to the trust store, when presented in a trust chain.
Test Steps Performed	The TOE does not have a configurable trust store for the update process. The TOE does not support updated images containing Cas. This test is not applicable.
Test Result	Pass; The TOE rejected the certificate without the BasicConstains extension.

Test Number	FIA_X509_EXT.1.2 test 2
Test Objective	Verify the TOE rejects a certificate with the CA flag set to FALSE when loaded to trust store, when presented in a trust chain.
Test Steps Performed	The TOE does not have a configurable trust store for the update process. The TOE does not support updated images containing Cas. This test is not applicable.
Test Result	Pass; The TOE rejected the certificate without the CA=True field.

2.27 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.

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.

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.

2.27.1 TSS

[ST] Section 7.3.7 describes that the security administrator configures the certificates the TOE uses for RMI, Ipsec and TLS Client (Syslog connection) using separate trust stores for each service. Trust store for the Trusted updates is not configurable.

[ST] Section 7.3.7 describes that for certificate revocation check to be successful, the CRL has to be accessible through the TOE management interface.

[ST] Section 7.3.7 describes that, when connection for revocation information cannot be made for certificates used for TLS authentication, either on import, or presented by remote entities and/or Ipsec peers, the TSF will accept certificates as valid when their revocation status is not available from the OE, given that all other required X.509 validation steps are successful.

[ST] Section 7.3.6 describes that if CRL is signed with the CA without CRL sign key usage, TSF will reject the CRL information and will not accept corresponding certificate.

The TOE does not support the configuration of this functionality.

2.27.2 Guidance Documentation

[AGD] Section 6.3 describes that the firewall configuration is necessary for the TOE to enable a successful revocation check. It also describes that DNS entries can be configured on the TOE to enable revocation checks. Note that [AGD] Section 6.3 describes that CRL downloads are performed through the TOE management network interface.

[AGD] Section 6.3 describes that if a connection to the CRL distribution point cannot be established the certificate is accepted as valid.

[AGD] Section 8.8 describes that when the certificate revocation check using the CRL distribution point is not available when verifying the certificate in the trusted update mechanism, the update is rejected.

2.27.3 Tests (TLS Client)

Test Number	1
Test Objective	Verify the TOE performs action described in the ST (accepts certificate) if unable to download CRL revocation server.
Test Steps Performed	Disable the CLR server when importing Leaf certificate to the TOE.

	Certificate is accepted as stated in [ST]. Disable the CLR server when establishing a Syslog connection. Certificate is accepted as stated in [ST]. Use Rest API to create a CSR request for syslog service, using /files/:service/csr access point and parameters for DN: C=US, O=ULVS, OU=CC Team, CN=toe.local, EC curve, P-384. CSR is created.
Test Result	Pass; The TOE took the ST-selected action when unable to download the CRL.

Test Number	2
Test Objective	Verify the path is validated when CSR is imported.
Test Steps Performed	Disable the CLR server when importing Leaf certificate to the TOE. Certificate is accepted as stated in [ST]. Disable the CLR server when establishing a Syslog connection. Certificate is accepted as stated in [ST]. Use Rest API to create a CSR request for syslog service, using /files/:service/csr access point and parameters for DN: C=US, O=ULVS, OU=CC Team, CN=toe.local, EC curve, P-384. CSR is created.
Test Result	Pass; The TOE took the ST-selected action when unable to download the CRL.

2.27.4 Tests (TLS Server)

Test Number	1
Test Objective	Verify the TOE performs action described in [ST] (accepts certificate) if unable to download CRL revocation server.
Test Steps Performed	Disable the CRL server when importing the Intermediate CA and Leaf certificate to the TOE. Import the Intermediate CA and leaf certificate. Certificate is accepted as stated in [ST].
Test Result	Pass; The TOE took the ST-selected action when unable to download the CRL.

Test Number	2
Test Objective	Verify the path is validated when CSR is imported.
Test Steps Performed	Use REST API to generate a CSR for the TOE Rest API service using DN: “C=US, O=ULVS, OU=CC Team, CN=rest.toe.local”, EC Curve P-384 Verify that the CSR is generated. Inspect the CSR using openssl tool. Verify that the CSR contains fields for CN, O, Country, OU, public key.
Test Result	Pass; The TOE took the ST-selected action when unable to download the CRL.

2.27.5 Tests (IPSEC)

Test Number	1
Test Objective	Verify the TOE performs action described in [ST] (accepts certificate) if unable to download CRL revocation server.
Test Steps Performed	Establish an Ipsec session and inspect the CRL server connections.

	Verify that the connection is established. The TOE performs the CRL requests. Disable the CRL server. Re-establish an Ipsec connection. Verify that CRL requests fail and an Ipsec connection is established.
Test Result	Pass; The TOE took the ST-selected action when unable to download the CRL.

Test Number	2
Test Objective	Verify the path is validated when CSR is imported.
Test Steps Performed	Use REST API to generate a CSR for the TOE Rest API service using DN: “C=US, O=ULVS, OU=CC Team, CN=rest.toe.local”, EC Curve P-384 Verify that the CSR is generated. Inspect the CSR using openssl tool. Verify that the CSR contains fields for CN, O, Country, OU, public key.
Test Result	Pass; The TOE took the ST-selected action when unable to download the CRL.

2.27.6 Tests (Trusted Update)

Test Number	1
Test Objective	Verify the TOE performs action described in [ST] (accepts certificate) if unable to download CRL revocation server.
Test Steps Performed	The TOE does not have a configurable trust store for the update process. The TOE does not support update images containing Cas. This test is not applicable.
Test Result	Pass

Test Number	2
Test Objective	Verify the path is validated when CSR is imported.
Test Steps Performed	Disable the CRL server when performing a trusted update. Verify that the Certificate is not accepted as stated in [ST].
Test Result	Pass; The TOE took the ST-selected action when unable to download the CRL.

2.28 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.

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.

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.

2.28.1 TSS

[ST] Section FIA_X509_EXT.3 does not contain a selection for “device-specific information”.

2.28.2 Guidance Documentation

[ST] Section FIA_X509_EXT.3 contains a selection for Common Name, Organization, Organizational Unit, Country. [AGD] Section 6.1.1 describe how to create a CSR, including providing a Distinguished Name flied.

2.28.3 Tests (TLS Client)

Test Number	1
Test Objective	Verify the TOE can create a CSR containing public key, device-specific information, Common Name, Organization, Organizational Unit, and Country.
Test Steps Performed	Use Rest API to create a CSR request for syslog service, using /files/:service/csr access point and parameters for DN: C=US, O=ULVS, OU=CC Team, CN=toe.local, EC curve, P-384. Verify that the CSR is created. Inspect a CSR using openssl tool. Verify that the CSR contains fields for CN, O, Country, OU, public key.
Test Result	Pass; The TOE created CSRs containing the listed information.

Test Number	2
Test Objective	Verify the path is validated when CSR is imported.
Test Steps Performed	Remove all CA from a syslog root store. Verify that the Root CA and Intermediate CA is removed from trust store. Create a CSR and create a certificate based on it. Upload a signed certificate to the TOE without trust path. Verify that the upload is rejected. Upload Root CA certificate to the TOE. Verify that the Root CA is uploaded. Create a signed certificate based on the CSR created in Test 1. Upload signed CSR without uploading intermediate CA. Verify that the Certificate import is rejected. Upload a valid and correct Intermediate CA with EC defined as a named curve. Verify that the Intermediate CA certificate is uploaded successfully.

	Upload signed certificate. Verify that the Certificate import is accepted.
Test Result	Pass; The TOE validated the trust chain path prior to installing the CSR-generated certificate.

2.28.4 Tests (TLS Server)

Test Number	1
Test Objective	Verify the TOE can create a CSR containing public key, device-specific information, Common Name, Organization, Organizational Unit, and Country.
Test Steps Performed	Use Rest API to create a CSR request for syslog service, using /files/:service/csr access point and parameters for DN: C=US, O=ULVS, OU=CC Team, CN=rest.toe.local, EC curve, P-384. Verify that the CSR is created. Inspect the CSR using openssl tool. Verify that the CSR contains fields for CN, O, Country, OU, public key.
Test Result	Pass; The TOE created CSRs containing the listed information.

Test Number	2
Test Objective	Verify the path is validated when CSR is imported.
Test Steps Performed	Remove all CA from a syslog root store. Verify that the Root CA and Intermediate CA is removed from trust store. Create a CSR and create a certificate based on it. Upload a signed certificate to the TOE without trust path. Verify that the upload is rejected. Upload Root CA certificate to the TOE. Verify that the Root CA is uploaded. Create a signed certificate based on the CSR created in Test 1. Upload signed CSR without uploading intermediate CA. Verify that the certificate import is rejected. Upload a valid and correct Intermediate CA with EC defined as a named curve. Verify that the Intermediate CA certificate is uploaded successfully. Upload signed certificate. Verify that the Certificate import is accepted.
Test Result	Pass; The TOE validated the trust chain path prior to installing the CSR-generated certificate.

2.28.5 Tests (IPSEC)

Test Number	1
Test Objective	Verify the TOE can create a CSR containing public key, device-specific information, Common Name, Organization, Organizational Unit, and Country.
Test Steps Performed	Use Rest API to create a CSR request for syslog service, using /files/:service/csr access point and parameters for DN: C=US, O=ULVS,

	OU=CC Team, CN=ipsec.toe.local, EC curve, P-384. Verify that the CSR is created. Inspect the CSR using openssl tool. Verify that the CSR contains fields for CN, O, Country, OU, public key. Use REST API to generate a CSR for the TOE Ipsec service using DN: "C=US, O=ULVS, OU=CC Team, CN=ipsec.toe.local", EC Curve P-256 Verify that the CSR is generated.
Test Result	Pass; The TOE created CSRs containing the listed information.

Test Number	2
Test Objective	Verify the path is validated when CSR is imported.
Test Steps Performed	Remove all Cas from RestAPI trust store. Verify that the Cas are removed. Create a certificate based on the CSR created for Test 1. Upload a certificate. Verify that the upload is rejected. Continue with CSR from Test 1. Upload Root CA certificate to the TOE. Verify that the Root CA is uploaded. Upload signed CSR without uploading intermediate CA. Verify that the certificate import is rejected. Upload a valid and correct Intermediate CA with EC defined as a named curve. Intermediate CA certificate is uploaded successfully. Upload signed certificate. Verify that Certificate import is accepted.
Test Result	Pass; The TOE validated the trust chain path prior to installing the CSR-generated certificate.

2.29 FMT_MOF.1/ManualUpdate

TSS

For distributed TOEs see chapter 2.4.1.1. There are no specific requirements for non-distributed TOEs.

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).

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.

2.29.1 TSS.

None

2.29.2 Guidance Documentation

The TOE is not a distributed TOE.

[AGD] Section 8.8 contains guidance on how to perform a manual update, and that the device automatically reboots after performing a successful update.

2.29.3 Tests

Test Number	1
Test Objective	Verify a manual update cannot be performed without prior authentication as a Security Administrator.
Test Steps Performed	Attempt to apply a software update without prior authentication using REST API. Verify that the Request is rejected. Attempt to apply a software update with prior authentication using REST API. Verify that the request is accepted.
Test Result	Pass; An update was performed only with authentication.

2.30 FMT_MOF.1/Services (Selection-Based)

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.

Guidance Documentation

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.

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.

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.

2.30.1 TSS

[ST] Section 7.4.1 describes that the TOE allows the system administrator to start and stop services (Syslog client service, RMI and Ipsec service). It describes the interface through which it can be achieved for each service.

The TOE is not a distributed TOE.

2.30.2 Guidance Documentation

The TOE is not a distributed TOE.

[AGD] Section 5.1.1 describes how to start and stop the remote syslog service.

[AGD] Section 4.1.1 describes how to start and stop the Rest API service.

[AGD] Section 7.2.1 describes how to start and stop the Ipsec service.

2.30.3 Tests

Test Number	1
Test Objective	Verify the Management functions for start and stop for a trusted channel service cannot be performed without prior authentication as a Security Administrator.
Test Steps Performed	Attempt to disable and enable the syslog service without prior authentication using Rest API. Verify that the Request is rejected. Attempt to disable and enable the syslog service with prior authentication using Rest API. Verify that the Request is accepted.
Test Result	Pass; Services were stopped only with proper authentication.

2.31 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 areaccessiblethroughaninterfacepriorto administratorlog-inareidentified. For each of these functions, the evaluator shall also confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

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

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.

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.

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.

2.31.1 TSS

[ST] Section 7.4.1 describes that the TOE does not allow administrative actions to be performed prior to authentication. The TOE describes administrative actions available to the TOE administrators and the corresponding administrative interface. [ST] Section 7.4.3 describes that ability to perform administrative actions for non-administrative users are restricted by the TOE's access control TSF.

[ST] Section 7.4.3 describes that the ability to manage the trust store is limited to security administrators by the TOE's access control.

2.31.2 Guidance Documentation

[AGD] identifies each of the TSF-data-manipulating functions implemented by the TOE and ensures only administrators have access to them.

[AGD] Section 4.1.1 describes that no access to the device is allowed prior to successful identification and authentication with the listed exception of displaying remote banner.

[AGD] Sections 6.1 and 6.2 describe managing the TOE trust store in a secure way. It describes the loading of CA certificates (Root and Intermediate Cas) into the trust store. [AGD] Section 5.2 describes that Root Cas are treated as a trust anchor.

2.31.3 Tests

Test Number	1
Test Objective	Verify the management functions not tested as part of other test activity. Management functions available through the local console are not available to an unauthenticated user, as shown by testing in FIA_UIA_EXT.1.2 – Test 3.
Test Steps Performed	Attempt to set DNS mappings without prior authentication. Verify that the request is rejected. Attempt to set DNS mappings with authentication. Verify that the request is accepted.
Test Result	Pass; Configuration was done only with proper authentication.

2.32 FMT_MTD.1/CryptoKeys Management of TSF Data (Selection-Based)

TSS

For distributed TOEs see chapter 2.4.1.1.

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

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.

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.

2.32.1 TSS

[ST] Section 7.4.3 lists the keys that the Security administrator can manage via the RMI and CLI interface:

- TOE Ipsec keys
- TOE TLS server keys
- TOE TLS Client keys
- Certificate authority certificates

[ST] Section 7.4.3 describes that those keys can be generated and deleted by the security administrator.

2.32.2 Guidance Documentation

The TOE is not a distributed TOE.

[AGD] Section 6 describes how the security administrator manages cryptographic keys: generates, deletes keys, imports signed certificates and manages CA certificates.

2.32.3 Tests

Test Number	1
Test Objective	Verify the Management functions that import cryptographic keys for an Ipsec service cannot be performed without prior authentication as a Security Administrator.
Test Steps Performed	Attempt to upload a CA file for Ipsec without prior authentication using Rest API. Verify that the request is rejected. Attempt to upload a CA file for Ipsec with prior authentication using Rest API. Verify that the request is accepted. Attempt to generate private key for the TOE Ipsec without prior authentication using Rest API. Verify that the request is rejected. Attempt to generate private key for TOE Ipsec with prior authentication using Rest API. Verify that the request is accepted.
Test Result	Pass; Configuration could not be performed without prior authentication.

2.33 FMT_SMF.1 Specification of Management Functions

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

TSS (containing also requirements on Guidance Documentation and Tests)

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

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

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

Guidance Documentation

See section 2.4.4.1

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.

2.33.1 TSS

[ST] Section 7.4.1 describes the local administrative interface as a restricted command-line shell.

[ST] Section 7.4.1 and 7.4.3 describes the TOE management functions and administrative interfaces through which they are accessible.

The evaluator verified that the management functions specified in the FMT_SMF.1 are provided by the TOE as observed in functional testing and described in the AGD in the TSS.

Function from FMT_SMF.1	Functional Testing	[AGD]	TSS
Ability to administer the TOE locally and remotely;	X	X	X
Ability to configure the access banner;	X	X	X
Ability to configure the session inactivity time before session termination or locking;	X	X	X
Ability to update the TOE, and to verify the updates using digital signature capability prior to installing those updates;	X	X	X
Ability to configure the authentication failure parameters for FIA_AFL.1;	X	X	X
Ability to manage the cryptographic keys;	X	X	X

Ability to configure the lifetime for Ipsec Sas;	X	X	X
Ability to import X.509v3 certificates to the TOE's trust store;	X	X	X
Ability to start and stop services	X	X	X
Ability to set the time which is used for time-stamps;	X	X	X
Ability to configure the reference identifier for the peer;	X	X	X
Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors;	X	X	X

[AGD] describes local user interface as a local VM console, and easily distinguishable from RMI.

2.33.2 Guidance Documentation

The TOE is not a distributed TOE.

[AGD] Section 3.1 describes the TOE's local interface, available through the Hyper-V local console. The local interface usage and representation significantly differs from the remote management interface, so there is no risk of the administrator confusing them.

[ST] contains the following management functions specified in FMT_SMF.1:

- Ability to administer the TOE locally and remotely.
- Ability to configure the access banner.
- Ability to configure the session inactivity time before session termination or locking
- Ability to update the TOE, and to verify the updates using digital signature and <u>no other </u>capability prior to installing those updates;
- Ability to configure the authentication failure parameters for FIA_AFL.1
- Ability to manage the cryptographic keys
- Ability to configure the lifetime for Ipsec Sas
- Ability to import X.509v3 certificates to the TOE's trust store
- Ability to start and stop services
- Ability to set the time which is used for time-stamps
- Ability to configure the reference identifier for the peer
- Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors

[ST] Section 7.4.1 (TSS) lists the following management functions and their interfaces:

Function	Interface
Ability to configure the access banner	CLI
Manage X.509 certificates and keys and trust anchors	RMI
Configure IKEv2 SA lifetimes	CLI
Configure IKEv2 algorithms	RMI
Configure Ipsec peer identities	RMI
Generate CSR (ECDSA key pair)	RMI
Manage security administrator password	CLI
Configure minimum password length	CLI
Configure the remote administrator inactivity timeout	CLI
Configure the local administrator inactivity timeout	CLI
Manage the failed authentication lockout threshold	RMI
Configure Syslog server connectivity	RMI
Initiate an update to the software	RMI
Set the system date and time	Both CLI and RMI
Definition of packet filtering rules	Ipv4: Both CLI and RMI; Ipv6: CLI
Association of packet filtering rules to network	CLI

Function	Interface
interfaces	
Ordering of packet filtering rules by priority	Both CLI and RMI
Start and stop services	Both CLI and RMI

The [AGD] describes actions that are needed to perform listed management functions through the local interface. [AGD] and [RAR] describe actions needed to perform those administrative functions using a remote management interface.

2.33.3 Tests

Test Number	1
Test Objective	Verify the following TOE management functions are evaluated: • Ability to administer the TOE locally and remotely; • Ability to configure the access banner (tested in FTA_TAB.1); • Ability to configure the session inactivity time before session termination or locking (tested in FTA_SSL_EXT.1, FTA_SSL.3); • Ability to update the TOE, and to verify the updates using digital signature capability prior to installing those updates; (tested in FPT_TUD_EXT.2) • Ability to configure the authentication failure parameters for FIA_AFL.1 (tested in FIA_AFL.1); • Ability to manage the cryptographic keys; (tested in FMT_MTD.1/CryptoKeys): • Ability to configure the lifetime for Ipsec Sas (tested by FCS_IPSEC_EXT.1.7 and FCS_IPSEC_EXT.1.8); • Ability to import X.509v3 certificates to the TOE's trust store (tested in FIA_X509_EXT.3.2); • Ability to start and stop services (tested in FMT_MTD.1/Services); • Ability to set the time which is used for time-stamps (tested in FPT_STM_EXT.1); • Ability to configure the reference identifier for the peer; • Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors (tested in FIA_X509_EXT.1.2);
Test Steps Performed	Use Rest API to configure the IP address and reference identifier for Ipsec peer as per [AGD] Section 6.2.3, using ipsec/remote_host access point. Command is accepted. Read the configuration from the TOE. Verify that the saved configuration contains the configured ID. Attempt to use the management function to remove all Cas with authentication as a Security Administrator. Verify that the Certificates are deleted successfully.
Test Result	Pass; The TOE performed management functions as defined.

2.34 FMT_SMF.1/VPN Specification of Management Functions [MOD]

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.

Guidance
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.

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.

2.34.1 TSS

The Evaluator verified that the management functions specified in the FMT_SMF.1/VPN are provided by the TOE as observed in functional testing and described in the AGD in the TSS.

Function from FMT_SMF.1	**Functional Testing**	**[AGD]**	**TSS**
Definition of packet filtering rules	X	X	X
Association of packet filtering rules to network interfaces;	X	X	X
Ordering of packet filtering rules by priority;	X	X	X

[ST] Section 7.4.1 lists corresponding administrative interfaces for those functions.

[ST] Section 7.4.1 describes the local administrative interface as a restricted command-line shell.

2.34.2 Guidance Documentation

[ST] FMT_SMF.1/VPN lists the following management functions:

- Definition of packet filtering rules
- Association of packet filtering rules to network interfaces
- Ordering of packet filtering rules by priority
- No other capabilities.

[AGD] Section 3.1 describes the TOE's local interface as available through the Hyper-V local console. [AGD] Sections 7.2.4 and 7.2.6 describe actions that are needed to perform listed management functions through the local interface. [RAR] describes actions that are needed to perform management functions through the RMI.

2.34.3 Tests

Test Number	1
Test Objective	Verify the TOE is able to perform the following management functions: • Definition of packet filtering rules (resolved by testing in FPF_RUL_EXT.1) • Association of packet filtering rules to network interfaces • Ordering of packet filtering rules by priority (resolved by testing in FPF_RUL_EXT.1.5)
Test Steps Performed	Configure the TOE to accept and reply on ICMP requests from 0.0.0.0/0 on interface eth2 and deny ICMP requests on interface eth0 from 0.0.0.0/0 and LOG those packets using the local console.

	-A INPUT -p icmp -I eth2 -j ACCEPT -A INPUT -p icmp -i eth1 -j DROP -I INPUT 1 -p icmp -j LOG–-log-prefix''IP'' -A OUTPUT -p icmp -j ACCEPT The TOE is configured. Initiate ICMP pings from 10.2.2.3 via eth2. Verify that Packets are allowed by the TOE.
Test Result	Pass; The TOE correctly applied the packet filtering rules as configured.

2.35 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.

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.

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.

2.35.1 TSS

[ST] Section 7.1.1 describes that the TOE supports only one user role and that is the Security Administrator role.

2.35.2 Guidance Documentation

[AGD] Section 4.1 describes how the TOE can be administered both locally and remotely. It notes that default certificates used by the TOE for the remote management interface will not be trusted by a commercial HTTPS client. It contains guidance on how to configure the TOE to use administrator-defined certificates.

2.35.3 Tests

Test Number	1
Test Objective	Annotate the interface usage results during the course of testing into table.
Test Steps Performed	The Evaluator ensured that administrative actions to the TOE are possible through available local and remote management interfaces by using remote and management interfaces during the course of the evaluation.

Table 1 - Security Management Interfaces
X = Yes, O = No, N = N/A

SFR	Local	REST API / TLS
FAU_GEN.1 – Test 1	**X**	**X**
FAU_GEN.1/VPN – Test 1)	**X**	**X**

FAU_GEN.2 – Test 1	**N**	**N**
FAU_STG.1 – Test 1	**X**	**O**
FAU_STG.1 – Test 2	**X**	**O**
FAU_STG_EXT.1 – Test 1	**X**	**O**
FAU_STG_EXT.1 – Test 2	**X**	**O**
FAU_STG_EXT.1 – Test 3	**N**	**N**
FAU_STG_EXT.1 – Test 4	**N**	**N**
FAU_STG_EXT.2.1/LocSpace – Test 1	**X**	**O**
FCS_CKM.1.1 – Test Suite	**N**	**N**
FCS_CKM.1.1/IKE – Test Suite	**N**	**N**
FCS_CKM.2.1 – Test Suite	**O**	**O**
FCS_COP.1.1/DataEncryption – Test Suite	**N**	**N**
FCS_COP.1.1/SigGen – Test Suite	**N**	**N**
FCS_COP.1.1/Hash – Test Suite	**N**	**N**
FCS_COP.1.1/KeyedHash – Test Suite	**N**	**N**
FCS_HTTPS_EXT	**N**	**N**
FCS_IPSEC_EXT.1.1 – Test 1	**X**	**X**
FCS_IPSEC_EXT.1.1 – Test 2	**X**	**X**
FCS_IPSEC_EXT.1.2 – Test 1	**N**	**N**
FCS_IPSEC_EXT.1.3 – Test 1	**O**	**O**
FCS_IPSEC_EXT.1.3 – Test 2	**N**	**N**
FCS_IPSEC_EXT.1.4 – Test 1	**O**	**O**
FCS_IPSEC_EXT.1.5 – Test 1	**N**	**N**
FCS_IPSEC_EXT.1.5 – Test 2	**O**	**O**
FCS_IPSEC_EXT.1.6 – Test 1	**O**	**O**
FCS_IPSEC_EXT.1.7 – Test 1	**N**	**N**
FCS_IPSEC_EXT.1.7 – Test 2	**X**	**X**
FCS_IPSEC_EXT.1.8 – Test 1	**X**	**X**
FCS_IPSEC_EXT.1.8 – Test 2	**O**	**O**
FCS_IPSEC_EXT.1.10 – Test 1	**O**	**O**
FCS_IPSEC_EXT.1.10 – Test 2	**O**	**O**
FCS_IPSEC_EXT.1.11 – Test 1	**O**	**O**
FCS_IPSEC_EXT.1.12 – Test 1	**N**	**N**
FCS_IPSEC_EXT.1.12 – Test 2	**N**	**N**
FCS_IPSEC_EXT.1.12 – Test 3	**O**	**O**
FCS_IPSEC_EXT.1.12 – Test 4	**O**	**O**
FCS_IPSEC_EXT.1.13 – Test 1	**N**	**N**
FCS_IPSEC_EXT.1.14 – Test 1	**N**	**N**
FCS_IPSEC_EXT.1.14 – Test 2	**N**	**N**
FCS_IPSEC_EXT.1.14 – Test 3	**N**	**N**
FCS_IPSEC_EXT.1.14 – Test 4	**N**	**N**
FCS_IPSEC_EXT.1.14 – Test 5	**O**	**X**
FCS_IPSEC_EXT.1.14 – Test 6	**O**	**O**
FCS_RBG_EXT.1.2 – Test 1	**N**	**N**
FCS_TLSC_EXT.1.1 – Test 1	**O**	**O**
FCS_TLSC_EXT.1.1 – Test 2	**O**	**O**
FCS_TLSC_EXT.1.1 – Test 3	**O**	**O**
FCS_TLSC_EXT.1.1 – Test 4	**O**	**O**
FCS_TLSC_EXT.1.1 – Test 5	**O**	**O**
FCS_TLSC_EXT.1.1 – Test 6	**O**	**O**
FCS_TLSC_EXT.1.2 – Test 1	**O**	**O**
FCS_TLSC_EXT.1.2 – Test 2	**O**	**O**
FCS_TLSC_EXT.1.2 – Test 3	**O**	**O**
FCS_TLSC_EXT.1.2 – Test 4	**O**	**O**
FCS_TLSC_EXT.1.2 – Test 5	**O**	**X**
FCS_TLSC_EXT.1.2 – Test 6	**N**	**N**
FCS_TLSC_EXT.1.2 – Test 7	**N**	**N**
FCS_TLSC_EXT.1.3 – Test 1	**O**	**O**
FCS_TLSC_EXT.1.3 – Test 2	**N**	**N**
FCS_TLSC_EXT.1.3 – Test 3	**N**	**N**
FCS_TLSC_EXT.1.4 – Test 1	**O**	**O**

FCS_TLSC_EXT.2.1 – Test 1	**N**	**N**
FCS_TLSS_EXT.1.1 – Test 1	**O**	**O**
FCS_TLSS_EXT.1.1 – Test 2	**O**	**O**
FCS_TLSS_EXT.1.1 – Test 3	**O**	**O**
FCS_TLSS_EXT.1.2 – Test 1	**O**	**O**
FCS_TLSS_EXT.1.3 – Test 1	**O**	**O**
FCS_TLSS_EXT.1.3 – Test 2	**N**	**N**
FCS_TLSS_EXT.1.3 – Test 3	**N**	**N**
FCS_TLSS_EXT.1.4 – Test 1	**O**	**O**
FCS_TLSS_EXT.1.4 – Test 2	**O**	**O**
FCS_TLSS_EXT.1.4 – Test 3	**O**	**O**
FIA_AFL.1.2 – Test 1	**O**	**X**
FIA_AFL.1.2 – Test 2	**O**	**X**
FIA_PMG_EXT.1.1 – Test 1	**X**	**O**
FIA_PMG_EXT.1.1 – Test 2	**X**	**O**
FIA_UIA_EXT.1.2 – Test 1	**X**	**X**
FIA_UIA_EXT.1.2 – Test 2	**X**	**X**
FIA_UIA_EXT.1.2 – Test 3	**X**	**O**
FIA_UIA_EXT.1.2 – Test 4	**N**	**N**
FIA_UAU.7.1 – Test 1	**X**	**O**
FIA_X509_EXT.1.1/REV – Test 1 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 2 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 3 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 4 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 5 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 6 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 7 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8a (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8b (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8c (syslog)	**O**	**X**
FIA_X509_EXT.1.2/REV – Test 1 (syslog)	**O**	**X**
FIA_X509_EXT.1.2/REV – Test 2 (syslog)	**O**	**X**
FIA_X509_EXT.2.2 – Test 1 (syslog)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 1 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 2 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 3 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 4 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 5 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 6 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 7 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8a (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8b (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8c (restapi)	**O**	**X**
FIA_X509_EXT.1.2/REV – Test 1	**O**	**X**

(restapi)		
FIA_X509_EXT.1.2/REV – Test 2 (restapi)	**O**	**X**
FIA_X509_EXT.2.2 – Test 1 (restapi)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 1 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 2 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 3 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 4 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 5 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 6 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 7 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8a (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8b (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8c (ipsec)	**O**	**X**
FIA_X509_EXT.1.2/REV – Test 1 (ipsec)	**O**	**X**
FIA_X509_EXT.1.2/REV – Test 2 (ipsec)	**O**	**X**
FIA_X509_EXT.2.2 – Test 1 (ipsec)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 1 (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 2 (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 3 (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 4 (update)	**N**	**N**
FIA_X509_EXT.1.1/REV – Test 5 (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 6 (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 7 (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8a (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8b (update)	**O**	**X**
FIA_X509_EXT.1.1/REV – Test 8c (update)	**O**	**X**
FIA_X509_EXT.1.2/REV – Test 1 (update)	**O**	**X**
FIA_X509_EXT.1.2/REV – Test 2 (update)	**O**	**X**
FIA_X509_EXT.2.2 – Test 1 (update)	**O**	**X**
FIA_X509_EXT.3.2 – Test 1 (syslog)	**O**	**X**
FIA_X509_EXT.3.2 – Test 1 (restapi)	**O**	**X**
FIA_X509_EXT.3.2 – Test 1 (ipsec)	**O**	**X**
FIA_X509_EXT.3.2 – Test 2	**O**	**X**
FIA_X509_EXT.3.2 – Test 2 restapi	**O**	**X**
FIA_X509_EXT.3.2 – Test 2	**O**	**X**

ipsec		
FMT_MOF.1.1/ManualUpdate – Test 1	**O**	**X**
FMT_MOF.1.1/Services – Test 1	**O**	**X**
FMT_MTD.1.1/CryptoKeys – Test 1	**O**	**X**
FMT_MTD.1.1/CoreData	**O**	**X**
FMT_SMF.1.1 – Test 1	**O**	**X**
FMT_SMF.1.1/VPN – Test 1	**X**	**O**
FMT_SMR.2.3 – Test 1	**N**	**N**
FPF_RUL_EXT.1.1 – Test 1	**X**	**O**
FPF_RUL_EXT.1.1 – Test 2	**X**	**O**
FPF_RUL_EXT.1.4 – Test 1	**X**	**O**
FPF_RUL_EXT.1.4 – Test 2	**X**	**O**
FPF_RUL_EXT.1.5 – Test 1	**X**	**O**
FPF_RUL_EXT.1.5 – Test 2	**X**	**O**
FPF_RUL_EXT.1.6 – Test 1	**X**	**X**
FPF_RUL_EXT.1.6 – Test 2	**X**	**X**
FPF_RUL_EXT.1.6 – Test 3	**X**	**X**
FPF_RUL_EXT.1.6 – Test 4	**X**	**X**
FPF_RUL_EXT.1.6 – Test 5	**X**	**X**
FPF_RUL_EXT.1.6 – Test 6	**X**	**X**
FPF_RUL_EXT.1.6 – Test 7	**X**	**X**
FPF_RUL_EXT.1.6 – Test 8	**X**	**X**
FPF_RUL_EXT.1.6 – Test 9	**X**	**X**
FPF_RUL_EXT.1.6 – Test 10	**X**	**X**
FPT_TST_EXT.1.1 – Test 1	**X**	**O**
FPT_TUD_EXT.1.1 – Test 1	**X**	**X**
FPT_TUD_EXT.1.1 – Test 2	**O**	**X**
FPT_TUD_EXT.1.1 – Test 3	**N**	**N**
FPT_TUD_EXT.2.2 – Test 1	**X**	**X**
FPT_STM_EXT.1.2 – Test 1	**X**	**X**
FPT_STM_EXT.1.2 – Test 2	**N**	**N**
FPT_STM_EXT.1.2 – Test 3	**N**	**N**
FTA_SSL_EXT.1.1 – Test 1	**X**	**O**
FTA_SSL.3.1 – Test 1	**O**	**X**
FTA_SSL.4.1 – Test 1	**X**	**O**
FTA_SSL.4.1 – Test 2	**O**	**X**
FTA_TAB.1.1 -Test 1	**X**	**X**
FTP_ITC.1.3 – Test 1	**N**	**N**
FTP_ITC.1.3 – Test 2	**N**	**N**
FTP_ITC.1.3 – Test 3	**N**	**N**
FTP_ITC.1.3 – Test 4	**O**	**X**
FTP_ITC.1 /VPN – Test 1	**N**	**N**
FTP_TRP.1.3/Admin – Test 1	**N**	**N**
FTP_TRP.1.3/Admin – Test 2	**O**	**X**

Test Result	Pass; The TOE permited administration via the local and remote interfaces.

2.36 FPF_RUL_EXT.1 Packet Filtering Rules [MOD]

TSS

The evaluator shall verify that the TSS provide a description of the TOE's initialization and 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.

Guidance

The operational guidance associated with this requirement is assessed in the subsequent test EAs.

Tests
The evaluator shall perform the following 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 EAs.

FPF_RUL_EXT.1.2
There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.
FPF_RUL_EXT.1.3
There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.
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 8200)
  - 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.

Guidance
The evaluator 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)

    - destination address
    - protocol
- IPv6 (RFC 8200)
    - 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.

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
        - Destination Address
        - Protocol
    - IPv6
        - Source address
        - Destination Address
        - Next Header (Protocol)
    - TCP
        - Source Port
        - Destination Port
    - UDP
        - 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 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 combinations of protocols and attributes 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.

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.

Guidance

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.

Tests
The evaluator shall perform the following tests:

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

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. The evaluator shall verify the TSS describes when the IPv4 and IPv6 protocols supported by the TOE differ from the full list provided in the RFC Values for IPv4 and IPv6 table.

Guidance
The evaluator shall verify that the operational guidance describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall verify that the operational guidance provides the appropriate instructions to configure the behavior to discard packets with no matching rules. The evaluator shall verify that the operational guidance describes the range of IPv4 and IPv6 protocols supported by the TOE.

Tests
The evaluator shall perform the following tests:

- **Test 1:** The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.
- **Test 2:** The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.
- **Test 3:** The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the

TOE to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

- **Test 4:** The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.
- **Test 5:** The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.
- **Test 6:** The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.
- **Test 7:** The evaluator shall configure the TOE to permit and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are permitted (i.e., by capturing the packets after passing through the TOE) and logged.
- **Test 8:** The evaluator shall configure the TOE to discard and log protocol 6 (TCP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination TCP ports in order to ensure that they are denied (i.e., by capturing no applicable packets passing through the TOE) and logged.
- **Test 9:** The evaluator shall configure the TOE to permit and log protocol 17 (UDP) using a selected source port, a selected destination port, and a selected source and destination port combination. The evaluator shall generate packets matching the configured source and destination UDP ports in order to ensure that they are permitted (i.e., by capturing the packets

after passing through the TOE) and logged. Here the evaluator ensures that the UDP port 500 (IKE) is included in the set of tests.

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

The following table identifies the RFC defined values for the protocol fields for IPv4 and IPv6 to be used in configuring and otherwise testing packet filtering rule definition and enforcement:

Protocol	**Defined Attributes**
IPv4	• Transport Layer Protocol 1 - Internet Control Message • Transport Layer Protocol 2 - Internet Group Management • Transport Layer Protocol 3 - Gateway-to-Gateway • Transport Layer Protocol 4 - IP in IP (encapsulation) • Transport Layer Protocol 5 - Stream • Transport Layer Protocol 6 - Transmission Control • Transport Layer Protocol 7 - UCL • Transport Layer Protocol 8 - Exterior Gateway Protocol • Transport Layer Protocol 9 - Any private interior gateway • Transport Layer Protocol 10 - BBN RCC Monitoring • Transport Layer Protocol 11 - Network Voice Protocol • Transport Layer Protocol 12 - PUP • Transport Layer Protocol 13 - ARGUS • Transport Layer Protocol 14 - EMCON • Transport Layer Protocol 15 - Cross Net Debugger • Transport Layer Protocol 16 - Chaos • Transport Layer Protocol 17 - User Datagram • Transport Layer Protocol 18 - Multiplexing • Transport Layer Protocol 19 - DCN Measurement Subsystems • Transport Layer Protocol 20 - Host Monitoring

- Transport Layer Protocol 21 - Packet Radio Measurement
- Transport Layer Protocol 22 - XEROX NS IDP
- Transport Layer Protocol 23 - Trunk-1
- Transport Layer Protocol 24 - Trunk-2
- Transport Layer Protocol 25 - Leaf-1
- Transport Layer Protocol 26 - Leaf-2
- Transport Layer Protocol 27 - Reliable Data Protocol
- Transport Layer Protocol 28 - Internet Reliable Transaction
- Transport Layer Protocol 29 - ISO Transport Protocol Class 4
- Transport Layer Protocol 30 - Bulk Data Transfer Protocol
- Transport Layer Protocol 31 - MFE Network Services Protocol
- Transport Layer Protocol 32 - MERIT Internodal Protocol
- Transport Layer Protocol 33 - Sequential Exchange Protocol
- Transport Layer Protocol 34 - Third Party Connect Protocol
- Transport Layer Protocol 35 - Inter-Domain Policy Routing Protocol
- Transport Layer Protocol 36 - XTP
- Transport Layer Protocol 37 - Datagram Delivery Protocol
- Transport Layer Protocol 38 - IDPR Control Message Transport Protocol
- Transport Layer Protocol 39 - TP++ Transport Protocol
- Transport Layer Protocol 40 - IL Transport Protocol
- Transport Layer Protocol 41 - Simple Internet Protocol
- Transport Layer Protocol 42 - Source Demand Routing Protocol
- Transport Layer Protocol 43 - SIP Source Route
- Transport Layer Protocol 44 - SIP Fragment
- Transport Layer Protocol 45 - Inter-Domain Routing Protocol
- Transport Layer Protocol 46 - Reservation Protocol
- Transport Layer Protocol 47 - General Routing Encapsulation
- Transport Layer Protocol 48 - Mobile Host Routing Protocol

- Transport Layer Protocol 49 - BNA
- Transport Layer Protocol 50 - SIPP Encap Security Payload
- Transport Layer Protocol 51 - SIPP Authentication Header
- Transport Layer Protocol 52 - Integrated Net Layer Security TUBA
- Transport Layer Protocol 53 - IP with Encryption
- Transport Layer Protocol 54 - NBMA Next Hop Resolution Protocol
- Transport Layer Protocol 61 - Any host internal protocol
- Transport Layer Protocol 62 - CFTP
- Transport Layer Protocol 63 - Any local network
- Transport Layer Protocol 64 - SATNET and Backroom EXPAK
- Transport Layer Protocol 65 - Kryptolan
- Transport Layer Protocol 66 - MIT Remote Virtual Disk Protocol
- Transport Layer Protocol 67 - Internet Pluribus Packet Core
- Transport Layer Protocol 68 - Any distributed file system
- Transport Layer Protocol 69 - SATNET Monitoring
- Transport Layer Protocol 70 - VISA Protocol
- Transport Layer Protocol 71 - Internet Packet Core Utility
- Transport Layer Protocol 72 - Computer Protocol Network Executive
- Transport Layer Protocol 73 - Computer Protocol Heart Beat
- Transport Layer Protocol 74 - Wang Span Network
- Transport Layer Protocol 75 - Packet Video Protocol
- Transport Layer Protocol 76 - Backroom SATNET Monitoring
- Transport Layer Protocol 77 - SUN ND PROTOCOL-Temporary
- Transport Layer Protocol 78 - WIDEBAND Monitoring
- Transport Layer Protocol 79 - WIDEBAND EXPAK
- Transport Layer Protocol 80 - ISO Internet Protocol
- Transport Layer Protocol 81 - VMTP
- Transport Layer Protocol 82 - SECURE-VMTP

	• Transport Layer Protocol 83 - VINES • Transport Layer Protocol 84 - TTP • Transport Layer Protocol 85 - NSFNET-IGP • Transport Layer Protocol 86 - Dissimilar Gateway Protocol • Transport Layer Protocol 87 - TCF • Transport Layer Protocol 88 - IGRP • Transport Layer Protocol 89 - OSPFIGP • Transport Layer Protocol 90 - Sprite RPC Protocol • Transport Layer Protocol 91 - Locus Address Resolution Protocol • Transport Layer Protocol 92 - Multicast Transport Protocol • Transport Layer Protocol 93 - AX.25 Frames • Transport Layer Protocol 94 - IP-within-IP Encapsulation Protocol • Transport Layer Protocol 95 - Mobile Internetworking Control Protocol • Transport Layer Protocol 96 - Semaphore Communications Security Protocol • Transport Layer Protocol 97 - Ethernet-within-IP Encapsulation • Transport Layer Protocol 98 - Encapsulation Header • Transport Layer Protocol 99 - Any private encryption scheme • Transport Layer Protocol 100 - GMTP
IPv6	• Transport Layer Protocol 1 - Internet Control Message • Transport Layer Protocol 2 - Internet Group Management • Transport Layer Protocol 3 - Gateway-to-Gateway • Transport Layer Protocol 4 - IPv4 encapsulation • Transport Layer Protocol 5 - Stream • Transport Layer Protocol 6 - Transmission Control • Transport Layer Protocol 7 - CBT • Transport Layer Protocol 8 - Exterior Gateway Protocol • Transport Layer Protocol 9 - Any private interior gateway

- Transport Layer Protocol 10 - BBN RCC Monitoring
- Transport Layer Protocol 11 - Network Voice Protocol
- Transport Layer Protocol 12 - PUP
- Transport Layer Protocol 13 - ARGUS
- Transport Layer Protocol 14 - EMCON
- Transport Layer Protocol 15 - Cross Net Debugger
- Transport Layer Protocol 16 - Chaos
- Transport Layer Protocol 17 - User Datagram
- Transport Layer Protocol 18 - Multiplexing
- Transport Layer Protocol 19 - DCN Measurement Subsystems
- Transport Layer Protocol 20 - Host Monitoring
- Transport Layer Protocol 21 - Packet Radio Measurement
- Transport Layer Protocol 22 - XEROX NS IDP
- Transport Layer Protocol 23 - Trunk-1
- Transport Layer Protocol 24 - Trunk-2
- Transport Layer Protocol 25 - Leaf-1
- Transport Layer Protocol 26 - Leaf-2
- Transport Layer Protocol 27 - Reliable Data Protocol
- Transport Layer Protocol 28 - Internet Reliable Transaction
- Transport Layer Protocol 29 - Transport Protocol Class 4
- Transport Layer Protocol 30 - Bulk Data Transfer Protocol
- Transport Layer Protocol 31 - MFE Network Services Protocol
- Transport Layer Protocol 32 - MERIT Internodal Protocol
- Transport Layer Protocol 33 - Datagram Congestion Control Protocol
- Transport Layer Protocol 34 - Third Party Connect Protocol
- Transport Layer Protocol 35 - Inter-Domain Policy Routing Protocol
- Transport Layer Protocol 36 - XTP
- Transport Layer Protocol 37 - Datagram Delivery Protocol

- Transport Layer Protocol 38 - IDPR Control Message Transport Protocol
- Transport Layer Protocol 39 - TP++ Transport Protocol
- Transport Layer Protocol 40 - IL Transport Protocol
- Transport Layer Protocol 41 - IPv6 encapsulation
- Transport Layer Protocol 42 - Source Demand Routing Protocol
- Transport Layer Protocol 43 - Intentionally blank
- Transport Layer Protocol 44 - Intentionally blank
- Transport Layer Protocol 45 - Inter-Domain Routing Protocol
- Transport Layer Protocol 46 - Reservation Protocol
- Transport Layer Protocol 47 - General Routing Encapsulation
- Transport Layer Protocol 48 - Dynamic Source Routing Protocol
- Transport Layer Protocol 49 - BNA
- Transport Layer Protocol 50 - Intentionally Blank
- Transport Layer Protocol 51 - Intentionally Blank
- Transport Layer Protocol 52 - Integrated Net Layer Security
- Transport Layer Protocol 53 - IP with Encryption
- Transport Layer Protocol 54 - NBMA Address Resolution Protocol
- Transport Layer Protocol 55 - Mobility
- Transport Layer Protocol 56 - Transport Layer Security Protocol using Kryptonet key management
- Transport Layer Protocol 57 - SKIP
- Transport Layer Protocol 58 - ICMP for IPv6
- Transport Layer Protocol 59 - No Next Header for IPv6
- Transport Layer Protocol 60 - Intentionally Blank
- Transport Layer Protocol 61 - Any host internal protocol
- Transport Layer Protocol 62 - CFTP
- Transport Layer Protocol 63 - Any local network
- Transport Layer Protocol 64 - SATNET and Backroom EXPAK

- Transport Layer Protocol 65 - Kryptolan
- Transport Layer Protocol 66 - MIT Remote Virtual Disk Protocol
- Transport Layer Protocol 67 - Internet Pluribus Packet Core
- Transport Layer Protocol 68 - Any distributed file system
- Transport Layer Protocol 69 - SATNET Monitoring
- Transport Layer Protocol 70 - VISA Protocol
- Transport Layer Protocol 71 - Internet Packet Core Utility
- Transport Layer Protocol 72 - Computer Protocol Network Executive
- Transport Layer Protocol 73 - Computer Protocol Heart Beat
- Transport Layer Protocol 74 - Wang Span Network
- Transport Layer Protocol 75 - Packet Video Protocol
- Transport Layer Protocol 76 - Backroom SATNET Monitoring
- Transport Layer Protocol 77 - SUN ND PROTOCOL-Temporary
- Transport Layer Protocol 78 - WIDEBAND Monitoring
- Transport Layer Protocol 79 - WIDEBAND EXPAK
- Transport Layer Protocol 80 - ISO Internet Protocol
- Transport Layer Protocol 81 - VMTP
- Transport Layer Protocol 82 - SECURE-VMTP
- Transport Layer Protocol 83 - VINES
- Transport Layer Protocol 84 - TTP
- Transport Layer Protocol 85 - Internet Protocol Traffic Manager
- Transport Layer Protocol 86 - NSFNET-IGP
- Transport Layer Protocol 87 - Dissimilar Gateway Protocol
- Transport Layer Protocol 88 - TCF
- Transport Layer Protocol 89 - EIGRP
- Transport Layer Protocol 90 - OSPFIGP
- Transport Layer Protocol 91 - Sprite RPC Protocol
- Transport Layer Protocol 92 - Locus Address Resolution Protocol

- Transport Layer Protocol 93 - Multicast Transport Protocol
- Transport Layer Protocol 94 - AX.25 Frames
- Transport Layer Protocol 95 - IP-within-IP Encapsulation Protocol
- Transport Layer Protocol 96 - Mobile Internetworking Control Pro.
- Transport Layer Protocol 97 - Semaphore Communications Sec. Pro.
- Transport Layer Protocol 98 - Ethernet-within-IP Encapsulation
- Transport Layer Protocol 99 - Encapsulation Header
- Transport Layer Protocol 100 - GMTP
- Transport Layer Protocol 101 - Ipsilon Flow Management Protocol
- Transport Layer Protocol 102 - PNNI over IP
- Transport Layer Protocol 103 - Protocol Independent Multicast
- Transport Layer Protocol 104 - ARIS
- Transport Layer Protocol 105 - SCPS Transport Layer Protocol
- Transport Layer Protocol 106 - QNX
- Transport Layer Protocol 107 - Active Networks
- Transport Layer Protocol 108 - Payload Compression Protocol
- Transport Layer Protocol 109 - Sitara Networks Protocol
- Transport Layer Protocol 110 - Compaq Peer Protocol
- Transport Layer Protocol 111 - IPX in IP
- Transport Layer Protocol 112 - Virtual Router Redundancy Protocol
- Transport Layer Protocol 113 - PGM Reliable Transport Protocol
- Transport Layer Protocol 114 - Any 0-hop protocol
- Transport Layer Protocol 115 - Layer Two Tunneling Protocol
- Transport Layer Protocol 116 - D-II Data Exchange (DDX)
- Transport Layer Protocol 117 - Interactive Agent Transfer Protocol
- Transport Layer Protocol 118 - Schedule Transfer Protocol
- Transport Layer Protocol 119 - SpectraLink Radio Protocol
- Transport Layer Protocol 120 - UTI

- Transport Layer Protocol 121 - Simple Message Protocol
- Transport Layer Protocol 122 - SM
- Transport Layer Protocol 123 - Performance Transparency Protocol
- Transport Layer Protocol 124 - ISIS over IPv4
- Transport Layer Protocol 125 - FIRE
- Transport Layer Protocol 126 - Combat Radio Transport Protocol
- Transport Layer Protocol 127 - Combat Radio User Datagram
- Transport Layer Protocol 128 - SSCOPMCE
- Transport Layer Protocol 129 - IPLT
- Transport Layer Protocol 130 - Secure Packet Shield
- Transport Layer Protocol 131 - Private IP Encapsulation within IP
- Transport Layer Protocol 132 - Stream Control Transmission Protocol
- Transport Layer Protocol 133 - Fibre Channel
- Transport Layer Protocol 134 - RSVP-E2E-IGNORE
- Transport Layer Protocol 135 - Mobility Header
- Transport Layer Protocol 136 - UDPLite
- Transport Layer Protocol 137 - MPLS-in-IP
- Transport Layer Protocol 138 - MANET Protocols
- Transport Layer Protocol 139 - Host Identity Protocol
- Transport Layer Protocol 140 - Shim6 Protocol
- Transport Layer Protocol 141 - Wrapped Encapsulating Security Payload
- Transport Layer Protocol 142 - Robust Header Compression

: RFC Values for IPv4 and IPv6

2.36.1 TSS

FPF_RUL_EXT.1.1

[ST] Section 7.4.2 describes the TOE network interfaces as disabled during boot before power-on self-test are completed and not enabled until packet filtering rules are initialized. It describes that once the network interface is enabled, the TSF ensures that the packet flow is controlled by firewall rules. [ST] Section 7.2.4 describes the Linux iptables service is enforcing packet filtering.

[ST] Section 7.4.2 describes that firewall rules defined as IPsec Traffic Selectors are only enforced when the IPsec service is started on the TOE. Manually configured rules are still applied regardless of that.

FPF_RUL_EXT.1.4

[ST] Section 7.4.2 describes that the TOE supports packet filtering based on all the required fields for each of required protocols, and identifies corresponding RFCSs. It also describes that the Security Administrator can apply packet filtering rules to any particular network interface.

[ST] Section 7.4.2 describes that network interfaces covered by this TSF are Black, Red and management network interfaces.

[ST] Section 7.4.2 describes that the TOE supports the following packet filtering actions: Allow (Protect with IPsec), Drop, Bypass, Log.

[ST] Section 7.4.2 describes that protocol compliance has been tested using third party interoperability testing with known-good implementations.

FPF_RUL_EXT.1.5

[ST] Section 7.4.2 describes that TSF verifies that all packets that the TOE receives are processed through firewall rulesets. (Applied to inbound and outbound traffic). Rules are composed of IPsec SPD rules and device firewall rules defined by the security administrator. When IPsec service is enabled, rules required for basic IPsec operations are automatically created, and rules defined in IPsec SPD are automatically enforced and added to the end of the packet filtering rule-chain.

[ST] Section 7.4.2. contains description of rules the TOE automatically creates for TOE services.

[ST] Section 7.4.2 describes that the firewall rules are applied in the order specified by the security administrator.

[ST] Section 7.2.4 describes that packets not matching a rule are dropped by a hard coded 'DROP' rule.

FPF_RUL_EXT.1.6

[ST] Section 7.4.2 describes that the TOE supports the processing of all IPv4/IPv6 protocols listed in Table 3 of the [MODSD], for packet filtering functionality. Routing packets through IPsec tunnel is not supported for the IPv6.

[ST] Section 7.2.4 describes that packet, not matching a rule are dropped by a hard coded rule.

2.36.2 Guidance Documentation

FPF_RUL_EXT.1.4

[AGD] Section 7.2.4 describes that the TOE supports filtering by the required protocols and parameters. It describes that the permit and deny actions can be performed and that the rules can generate a security audit log.

[AGD] Section 7.2.4 describes that the TOE supports IPv4 and Ipv6 protocols, but only the Block and Bypass actions are available for Ipv6 protocols (no routing of Ipv6 traffic through the IPsec tunnel is supported).

[AGD] Section 7.2.6.1 and [AGD] Appendix B describe how rules can be associated with network interfaces.

FPF_RUL_EXT.1.5

[AGD] Section 7.2.4 state that the first matching rule is triggered.

[AGD] Section 7.2.5 describes that the traffic selectors are provisioned in the order they are provided. [AGD] Section 7.2.6 describes how rules are configured via local console and remote interface. [AGD] Section 7.2.6.1 and Appendix B describe the -I command, which is used to insert a rule in a specific place in the local console firewall rule chain.

[AGD] Section 7.2.7 describes the services for which the TOE automatically creates FW rules.

FPF_RUL_EXT.1.6

[AGD] Section 7.2.5 describes that the default deny rule will be enforced for packets that do not match any explicit rule.

[AGD] Section 7.2.4 describes that the TOE supports IPv4 and IPv6 protocols, but only the Block and Bypass actions are available for IPv6 protocols.

2.36.3 Tests

Test Number	FPF_RUL_EXT.1.1 – Test 1
Test Objective	Verify the TOE filters traffic as expected, and no traffic is permitted until the TOE initializes and Packet filtering rules are applied after the TOE initialization.
Test Steps Performed	Continue with the TOE state from FCS_IPSEC.1.1 -Test 2. Configure the TOE to drop UDP and log packets from 10.1.2.1 to 10.2.2.3 port 443 using local console. Save the firewall. Verify that The TOE is configured. Start a packet capture on both sides of the TOE. Start stream of UDP packets from 10.1.2.1 to 10.2.2.3:443 Verify that the Stream is started. Initiate a TOE reboot. Inspect the packet capture. Verify that Packets are not passing through the TOE before, during, and after reboot.
Test Result	Pass; The TOE correctly implemented the configured packet filtering rules, and did not permit traffic to flow until after the TOE was completely booted.

Test Number	FPF_RUL_EXT.1.1 – Test 2
Test Objective	Verify the TOE filters traffic as expected, and no traffic is permitted until the TOE initializes and Packet filtering rules are applied after TOE initialization.
Test Steps Performed	Continue with the TOE state from FCS_IPSEC.1.1 -Test 2. Configure the TOE to permit UDP packets from 10.1.2.1 to 10.2.2.8 port 443 using local console. Save the firewall. Start a packet capture on both sides of the TOE. Start stream of UDP packets from 10.1.2.1 to 10.2.2.3:443. Verify that Stream is started. The TOE is configured. Initiate a TOE reboot. Inspect the packet capture. Packets are not passing through the TOE during reboot.
Test Result	Pass; The TOE applied the packet filtering rules as expected.

Test Number	FUF_RUL_EXT.1.4 – Test 1
Test Objective	Verify that rules with required parameters for each protocol can be created. Rules for IPv4 TCP and UDP were tested in FPF_RUL_EXT.1.6.
Test Steps Performed	Configure the TOE to permit IPv6 TCP traffic to packethost and to and from specified ports and drop UDP packets to packethost to and from specified ports using the local console, allow ICMP packets. Verify that the TOE is configured. Use nc tool to initiate a TCP connection using Ipv6 with ports matching the rule (nc -6 fd00:0:0:2::3 443 -p 1500) and not matching the rule (Packets

	traverse the TOE when match the configured tcp rule). Use nc tool to initiate an UDP connection using IPv6. With ports matching the rule (nc -6 -u -p 1600 fd00:0:0:2::3 160) and not matching the rule (nc -6 -u -p 1600 fd00:0:0:2::3 166) Verify that Packets traverse the TOE when matching the rule and not traverse when not matching rule.
Test Result	Pass; The TOE applied the packet filtering rules for TCP and UDP IPv6 traffic.

Test Number	FPF_RUL_EXT.1.4-Test 2
Test Objective	Verify the TOE SFR works consistently across interface types.
Test Steps Performed	The TOE supports only ethernet interface type, therefore this test is satisfied.
Test Result	Pass; The TOE supported only one interface type.

Test Number	FPF_RUL_EXT.1.5-Test 1
Test Objective	Verify that the TOE is applying rules based on the rules order.
Test Steps Performed	Using local console configure the TOE to add a drop rule for UDP packets to 10.2.2.8 port 443. Verify that the rule is added. Initiate UDP packets through the TOE. Verify that packets are allowed through the TOE. Configure the TOE to add a drop rule for UDP packets to 10.2.2.8 port 443 with higher priority than allow rule. Verify that the rule is added. Initiate UDP packets through the TOE. Verify that packets are dropped.
Test Result	Pass; The TOE correctly applied packet filtering rules in a “top down” order.

Test Number	FPF_RUL_EXT.1.5 Test 2
Test Objective	Verify the TOE applies packet filtering rules sequentially regardless of specificity.
Test Steps Performed	Using the local console add a rule that will allow UDP packets to subnet 10.2.2.0/24 port 443 at position 19. Verify that the TOE is configured. Initiate packet flow from 10.1.2.1 to 10.2.2.8 to port 443. Verify that packets are allowed to pass through the TOE. Add a rule that will drop UDP packets to 10.2.2.8 port 443 at position 19. Verify that the TOE is configured. Initiate a packet flow from 10.1.2.1 to 10.2.2.8 to port 443. Verify that packets are not allowed to pass through the TOE.
Test Result	Pass; The TOE applied packet filtering rules in top down order, regardless of specificity.

Test Number	FPF_RUL_EXT.1.6 – Test 1

Test Objective	Verify all protocols can be permitted for: • Specific source address and specific destination address • Specific source address and wildcard destination address • Wildcard source address and specific destination address • Wildcard source address and wildcard destination address
Test Steps Performed	Configure the TOE to accept packets from a specified host to a specified host (10.1.2.1 to 10.2.2.3) using Rest API. Configure the TOE to log packets using the local console. Verify that The TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and inspect the capture. Verify that all packet types traverse the TOE successfully. Configure the TOE to accept packets from a specific address to a wildcard address using RestAPI. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and inspect the capture. Verify that all packet types traverse the TOE successfully. Configure the TOE to accept packets from a wildcard subnet to a specified host using RestAPI. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and inspect capture. Verify that all packet types traverse the TOE successfully. Configure the TOE to accept packets from a wildcard subnet to a wildcard subnet using RestAPI. Configure the TOE to log packets using the local console. Verify that The TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and the inspect capture. Verify that All packet types traverse the TOE successfully.
Test Result	Pass; The TOE permited the traffic when configured to do so.

Test Number	FPF_RUL_EXT.1.6-Test 2
Test Objective	Verify all protocols can be denied for: • Specific source address and specific destination address • Specific source address and wildcard destination address • Wildcard source address and specific destination address • Wildcard source address and wildcard destination address
Test Steps Performed	Configure the TOE to log traffic from a wildcard to a wildcard using the local console. Configure the TOE to drop traffic from a wildcard to a wildcard using RESTAPI. Verify that the TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and the inspect capture.

	Verify that all packet types are blocked from traversing the TOE. Configure the TOE to log traffic from a wildcard to a wildcard using the local console. Configure the TOE to drop traffic from a wildcard to a wildcard using RESTAPI. Verify that the TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and inspect the capture. Verify that all packet types are blocked from traversing the TOE. Configure the TOE to log traffic from a wildcard to a specified host 10.2.2.3 using the local console. Configure the TOE to drop traffic from wildcard to host 10.2.2.3 using RESTAPI. Verify that the TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and inspect the capture. Verify that all packet types are blocked from traversing the TOE. Configure the TOE to log traffic from a specified host 10.1.2.1 to a specified host 10.2.2.3 using the local console. Configure the TOE to drop traffic from a specified host 10.1.2.1 to a specified host 10.2.2.3using REST API. Verify that the TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and inspect the capture. Verify that all packet types are blocked from traversing the TOE.
Test Result	Pass; The TOE correctly denied traffic when configured to do so.

Test Number	FPF_RUL_EXT.1.6-Test 3
Test Objective	Verify an implicit deny rule is applied for all protocols after all permit and deny rules are applied.
Test Steps Performed	Using Rest API and local console to configure the TOE to accept packets. • From 10.1.2.6 to 172.16.2.1 • From 10.1.2.5 to 172.16.2.0/24 • From 172.16.2.0/24 to 10.1.2.5 • From 10.1.2.0/24 to 10.2.3.0/24 and deny packets: • From 10.1.2.2 to 10.2.2.5 • From 10.1.2.3 to 10.2.2.0/24 • From 10.2.2.0/24 to 10.1.2.3 • From 10.2.3.0/24 to 10.1.2.0/24 Log all packets received and log packets that are being forwarded to implicit rule. Verify that the TOE is configured. Start a packet capture and initiate ipv4 packets of all possible protocols from 10.1.2.1 to 10.2.2.3 and inspect the capture. Verify that all packet types are blocked from traversing the TOE.
Test Result	Pass; The TOE implicitly denied all traffic which did not match at least one rule in the packet filtering rules set.

Test Number	FPF_RUL_EXT.1.6-Test 4
Test Objective	Verify that all IPv6 protocols can be permitted for: • Specific source address and specific destination address • Specific source address and wildcard destination address • Wildcard source address and specific destination address • Wildcard source address and wildcard destination address
Test Steps Performed	Using the local console configure the TOE to accept and log packets from a specified host to a specified host (fd00:0:0:2::3 to fd00:0:0:1::1) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types traverse the TOE successfully. Configure the TOE to accept and log packets from a specified host to subnet (fd00:0:0:2::3 to fd00:0:0:1::/64) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types traverse the TOE successfully. Configure the TOE to accept and log packets from subnet to subnet (fd00:0:0:2::/64 to fd00:0:0:1::/64) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types traverse the TOE successfully. Configure the TOE to accept and log packets from subnet to host (fd00:0:0:2::/64 to fd00:0:0:1::1) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types traverse the TOE successfully.
Test Result	Pass; The TOE correctly permited traffic when configured to do so.

Test Number	FPF_RUL_EXT.1.6-Test 5
Test Objective	Verify that all IPv6 protocols can be denied based on rules for: • Specific source address and specific destination address • Specific source address and wildcard destination address • Wildcard source address and specific destination address • Wildcard source address and wildcard destination address
Test Steps Performed	Using the local console configure the TOE to drop and log packets from a specified host to a specified host (fd00:0:0:2::3 to fd00:0:0:1::1) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols

	from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types are blocked from traversing the TOE successfully. Configure the TOE to drop and log packets from a specified host to subnet (fd00:0:0:2::3 to fd00:0:0:1::/64) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types are blocked from traversing the TOE successfully. Configure the TOE to drop and log packets from a subnet to subnet host (fd00:0:0:2::/64 to fd00:0:0:1::/64) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types are blocked from traversing the TOE successfully. Configure the TOE to drop and log packets from a subnet to a host (fd00:0:0:2::/64 to fd00:0:0:1::1) using the local console. Configure the TOE to log packets using the local console. Verify that the TOE is configured. Start a packet capture and initiate IPv6 packets of all possible protocols from fd00:0:0:2::3 to fd00:0:0:1::1 and inspect the capture. Verify that all packet types are blocked from traversing the TOE successfully.
Test Result	Pass; The TOE correctly denied traffic when configured to do so.

Test Number	FPF_RUL_EXT.1.6 – test 6
Test Objective	Verify that an implicit deny rule is enforced for all protocols after all permit and deny rules.
Test Steps Performed	Configure the TOE to accept and log packets: • From fd00:0:0:1::1 to fd00:0:0:3::1 • From fd00:0:0:1::2 to fd00:0:0:3::0/64 • From fd00:0:0:3::0/64 to fd00:0:0:1::2 • From fd00:0:0:1::0/64 to fd00:0:0:3::0/64 And deny packet: • From fd00:0:0:1::2 to fd00:0:0:3::2 • From fd00:0:0:1::3 to fd00:0:0:2::0/64 • From fd00:0:0:2::0/64 to fd00:0:0:1::3 • From fd00:0:0:3::0/64 to fd00:0:0:1::0/64 Configure the TOE to log packets after those rules. Verify that the TOE was configured. Initiate packets for all protocols from fd00:0:0:1::1 to fd00:0:0:2::3. Verify that Packets are not traversing the TOE, log records are created for packets.

Test Result	Pass; The TOE implemented an implicit deny rule, blocking traffic which did not match at least one traffic filter rule.

Test Number	FPF_RUL_EXT.1.6 – Test 7
Test Objective	Verify that packet filtering can be configured to permit TCP packets using a combination of source and destination ports.
Test Steps Performed	Using the local console and Rest API, create rules to log and permit TCP traffic to port 443 from port 8080 and to port 77 from port 88. Verify that the TOE is configured. Initiate TCP packet to port 443, using nc tool. Verify that packets are accepted and logged. Initiate TCP packet from port 8080 using nc tool. Verify that packets are accepted and logged. Initiate TCP packet from port 88 to port 77 using nc tool. Verify that packets are accepted and logged.
Test Result	Pass; The TOE correctly permited the configured traffic.

Test Number	FPF_RUL_EXT.1.6 – Test 8
Test Objective	Verify packet filtering can be configured to drop TCP packets using combination of source and destination ports.
Test Steps Performed	Using the local console and Rest API, create rules to log and deny TCP traffic to port 443 from port 8080 and to port 77 from port 88. Create a permit rule for all TCP traffic as a control measure. Verify that the TOE is configured. Initiate TCP packet to port 443, using nc tool. Verify that packets are rejected and logged. Initiate TCP packet from port 8080 using nc tool. Verify that packets are rejected and logged. Initiate TCP packet from port 88 to port 77 using nc tool. Verify that packets are rejected and logged.
Test Result	Pass; The TOE correctly denied the configured traffic.

Test Number	FPF_RUL_EXT.1.6 – Test 9
Test Objective	Verify that packet filtering can be configured to permit UDP packets using combination of source and destination ports.
Test Steps Performed	Create rules to log and permit UDP traffic to port 500 from port 4500 and to port 1500 from port 5500 using Rest API and the local console (for logging rules) Verify that The TOE is configured. Initiate UDP packet to port 500, using nc tool. Verify that packets are accepted and logged. Initiate UDP packet from port 4500, using nc tool. Verify that packets are accepted and logged. Initiate UDP packet from port 5500 to port 1500, using nc tool. Verify that packets are accepted and logged.
Test Result	Pass; The TOE correctly permited the configured traffic.

Test Number	FPF_RUL_EXT.1.10
Test Objective	Verify packet filtering can be configured to deny UDP packets using combination of source and destination ports.
Test Steps Performed	Create rules to log and deny UDP traffic to port 500 from port 4500 and to port 1500 from port 5500 using Rest API and the local console (for logging rules). Create a wildcard permitting rule for control. Verify that the TOE is configured. Initiate UDP packet to port 500, using nc tool. Verify that packets are rejected and logged. Initiate UDP packet from port 4500, using nc tool. Verify that packets are rejected and logged. Initiate UDP packet from port 5500 to port 1500, using nc tool. Verify that packets are accepted and logged.
Test Result	Pass; The TOE correctly denied the configured traffic.

2.37 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.

2.37.1 TSS

[ST] Section 7.5.1. describes that the administrative passwords are stored salted and hashed using SHA-512 hashing algorithm. It describes that no TOE interface exists to view this password.

2.37.2 Guidance Documentation

None.

2.37.3 Tests

None.

2.38 FPT_FLS.1/SelfTest Failure with Preservation of Secure State (Self-Test Failures) [MOD]

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 shutdown 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.

Guidance

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.

Tests

There are no test EAs for this component.

2.38.1 TSS

[ST] Section 7.5.2 describes that upon failure of the each of the self-test (noise source health test, KAT test, Integrity check test) TOE provides a notification via CLI and shuts down. So, the TOE is expected to shut down in every case and should not be operational if any of the tests fail.

2.38.2 Guidance Documentation

[AGD] Section 9 describes the self-tests that are being performed on the TOE, and that the TOE will shut down upon failure of any of those tests. It describes that the self-test error message will be shown to the administrator. [AGD] Section 9.1 describes recommended remediation steps for each failure type.

2.38.3 Tests

None.

2.39 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.

2.39.1 TSS

[ST] Section 7.5.4 describes that the TOE does not provide an interface to allow reading of private key data. [ST] Section 7.2.1 describes that the TOE does not persistently store symmetric keys.

[ST] Section 7.5.4 states that all keys are stored on the TOE in plaintext.

2.39.2 Guidance Documentation

None.

2.39.3 Tests

None.

2.40 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.

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.

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.

2.40.1 TSS

[ST] Section 7.5.5 describes that the TOE uses time stamps for audit records, validity checks and CRL validation for x509 certificates.

It describes that the system clock is maintained by underlying Windows 10 Hypervisor.

[ST] Section 7.5.5 states that the TOE obtains time from the underlying virtualization system or time can be set manually by the Security Administrator. [ST] Section 7.5.5 describes that time synchronization with the underlying Hypervisor occurs only at certain times and is dependent on time difference between the TOE and hypervisor.

2.40.2 Guidance Documentation

[AGD] Section 8.9 describes how the time can be set using the local console. [RAR] describes how time can be set using the REST API.

The TOE does not support NTP server. The TOE does not claim obtaining time from the underlying VS and contains guidance to disable this service in the VS during TOE provisioning.

2.40.3 Tests

Test Number	1
Test Objective	Verify the TOE allows the SA to set the time and that the set time is correctly displayed.
Test Steps Performed	Use the local console to login into TOE and navigate to time menu. Verify that Local system time is displayed. Set a new time value to the TOE. Verify that new time is saved. Set a new time to the TOE using REST API. Verify that new time saved.
Test Result	Pass; The TOE correctly allowed the administrator to configure and display the time.

Test Number	2
Test Objective	Verify the TOE is able to use NTP as a time source.
Test Steps Performed	The TOE does not claim NTP support, so this test is implicitly satisfied.
Test Result	Pass; The TOE does not claim NTP time source support.

Test Number	3
Test Objective	Verify the TOE is able to use underlying VS as a time source.
Test Steps Performed	Use the local console to login to the TOE and navigate to the time menu. Note TOE time. Verify that Local system time is displayed. Set a new time value on the Virtualization System. Verify that new time saved on the VS. Initiate a VS pause and resume event. Check the TOE timestamp. Verify that the time is synced between the TOE and VS.
Test Result	Pass; The TOE correctly retrieved time from the underlying virtualization system when configured to do so.

2.41 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.

For distributed TOEs the evaluator shall examine the TSS to ensure that it details which TOE component performs which self-tests and when these self- tests are run.

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.

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.

2.41.1 TSS

[ST] Section 7.5.2 describes the self-tests run by the TOE on power on. It contains a description for each self-test performed. It describes that the TOE performs an entropy noise source health check and Known Answer Test (KAT) for each cryptographic algorithm implemented by the TSF. The TOE also performs integrity checks by verifying digest and digital signature of the TOE firmware.

[ST] Section 7.5.2 makes the argument that those tests combined ensure the correct behavior of the TSF.

The TOE is not a distributed TOE.

2.41.2 Guidance Documentation

[AGD] Section 9 describes the self-tests being performed on the TOE and that the TOE will shut down upon failure of any of those tests.

2.41.3 Tests

Test Number	1
Test Objective	Verify the TOE performs an integrity check for firmware and executable software. Verify the TOE performs verification of the correct operation of the crypto functions necessary for all SFRs: • AES-CBC • AES-CTR (DRBG) • AES-GCM • ECDSA • HMAC-SHA384 • SHA256, SHA384, SHA512 • ECDHE
Test Steps Performed	Mount the TOE hard drive to another VM, modify the TOE files. Verify that the TOE filesystem accesses and modifies files. Start the TOE. Verify that the TOE integrity check fails. Verify the TOE scope of cryptographic self-tests. Self-tests cover crypto functionality for SFR.
Test Result	Pass; The TOE correctly performed integrity checking of the TSF.

2.42 FPT_TST_EXT.3 Self-Test with Defined Methods [MOD]

TSS

The evaluator shall verify 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.

Guidance

There are no guidance EAs for this component.

Tests

There are no test EAs for this component.

2.42.1 TSS

[ST] Section 7.5.2 describes a method used for the self-testing of the TOE's executable code. It describes that checking of the integrity checks is performed using digests and verifying digital signature of the TOE firmware. This is consistent with the assignment in the FPT_TST_EXT.3

2.42.2 Guidance Documentation

None.

2.42.3 Tests

None.

2.43 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.

If the options 'support automatic checking for updates' or 'support automatic updates' are chosen from the selection in FPT_TUD_EXT.1.2, the evaluator shall verify that the TSS explains what actions are involved in automatic checking or automatic updating by the TOE, respectively.

For distributed TOEs, the evaluator shall examine the TSS to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component. Alternatively, this description can be provided in the guidance documentation. In that case the evaluator should examine the guidance documentation instead.

If a published hash is used to protect the trusted update mechanism, then the evaluator shall verify that the trusted update mechanism does involve an active authorization step of the Security Administrator, and that download of the published hash value, hash comparison and update is not a fully automated process involving no active authorization by the Security Administrator. In particular, authentication as Security Administration according to FMT_MOF.1/ManualUpdate needs to be part of the update process when using published hashes.

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.

The evaluator shall verify that the guidance documentation describes how the verification of the authenticity of the update is performed (digital signature verification or verification of published hash). The description shall include the procedures for successful and unsuccessful verification. The description shall correspond to the description in the TSS.

If a published hash is used to protect the trusted update mechanism, the evaluator shall verify that the guidance documentation describes how the Security Administrator can obtain authentic published hash values for the updates.

For distributed TOEs the evaluator shall verify that the guidance documentation describes how the versions of individual TOE components are determined for FPT_TUD_EXT.1, how all TOE components are updated, and the error conditions that may arise from checking or applying the update (e.g. failure of signature verification, or exceeding available storage space) along with appropriate recovery actions. . The guidance documentation only has to describe the procedures relevant for the Security Administrator; it does not need to give information about the internal communication that takes place when applying updates.

If this was information was not provided in the TSS: For distributed TOEs, the evaluator shall examine the Guidance Documentation to ensure that it describes how all TOE components are updated, that it describes all mechanisms that support continuous proper functioning of the TOE during update (when applying updates separately to individual TOE components) and how verification of the signature or checksum is performed for each TOE component.

If this was information was not provided in the TSS: If the ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the Guidance Documentation contains a description of how the certificates are contained on the device. The evaluator also ensures that the Guidance Documentation describes how the certificates are installed/updated/selected, if necessary.

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.

b) Test 2 [conditional]: If the TOE itself verifies a digital signature to authorize the installation of an image to update the TOE the following test shall be performed (otherwise the test shall be omitted). The evaluator first confirms that no updates are pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test. The evaluator obtains or produces illegitimate updates as defined below and attempts to install them on the TOE. The evaluator verifies that the TOE rejects all of the illegitimate updates. The evaluator performs this test using all of the following forms of illegitimate updates:

1) A modified version (e.g. using a hex editor) of a legitimately signed update

2) An image that has not been signed

3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)

4) If the TOE allows a delayed activation of updates the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs depending on the point in time when an attempted update is rejected. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

c) Test 3 [conditional]: If the TOE itself verifies a hash value over an image against a published hash value (i.e. reference value) that has been imported to the TOE from outside such that the TOE itself authorizes the installation of an image to update the TOE, the following test shall be performed (otherwise the test shall be omitted. If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

1) The evaluator obtains or produces an illegitimate update such that the hash of the update does not match the published hash. The evaluator provides the published hash value to the TOE and calculates the hash of the update either on the TOE itself (if that functionality is provided by the TOE), or else outside the TOE. The evaluator confirms that the hash values are different, and attempts to install the update on the TOE, verifying that this fails because of the difference in hash values (and that the failure is logged). Depending on the implementation of the TOE, the TOE might not allow the Security Administrator to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

2) The evaluator uses a legitimate update and tries to perform verification of the hash value without providing the published hash value to the TOE. The evaluator confirms that this attempt fails. Depending on the implementation of the TOE it might not be possible to attempt the verification of the hash value without providing a hash value to the TOE, e.g. if the hash value needs to be handed over to the TOE as a parameter in a command line message and the syntax check of the command prevents the execution of the command without providing a hash value. In that case the mechanism that prevents the execution of this check shall be tested accordingly, e.g. that the syntax check rejects the command without providing a hash value, and the rejection of the attempt is regarded as sufficient verification of the correct behaviour of the TOE in failing to verify the hash. The evaluator then attempts to install the update on the TOE (in spite of the unsuccessful hash verification) and confirms that this fails. Depending on the implementation of the TOE, the TOE might not allow to even attempt updating the TOE after the verification of the hash value fails. In that case the verification that the hash comparison fails is regarded as sufficient verification of the correct behaviour of the TOE

3) If the TOE allows delayed activation of updates, the TOE must be able to display both the currently executing version and most recently installed version. The handling of version information of the most recently installed version might differ between different TOEs. Depending on the point in time when the attempted update is rejected, the most recently installed version might or might not be updated. The evaluator shall verify that the TOE handles the most recently installed version information for that case as described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 3 shall be skipped.

The evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all methods supported (manual updates, automatic checking for updates, automatic updates).

For distributed TOEs the evaluator shall perform Test 1, Test 2 and Test 3 (if applicable) for all TOE components.

2.43.1 TSS

The TOE is not a distributed TOE.

[ST] Section 6.1.5.7 FPT_TUD_EXT.1.1 states that the TOE uses digital signature and x.509 certificate to authenticate the update.

[ST] Section 6.1.5.7 FPT_TUD_EXT.1.2 states only manual update is supported.

[ST] Section 7.5.3 describes that the current software version can be queried through the CLI.

[ST] Section 7.5.3 describes that software updates are installed using the remote management interface, with the update image being obtained from Viasat. It describes that upon an attempt to update the TOE, the digital signature of the image is verified automatically. If the digital signature verification is successful, an upgrade is performed. If unsuccessful, the update is aborted and an audit record is generated.

2.43.2 Guidance Documentation

[AGD] Section 8.8 describes how the current active version can be queried using the local console. The TOE does not support delayed activation of the update.

[AGD] Section 8.8 describes update authenticity is determined by checking the digital signature. It describes that update is performed and audit record is created, or that the update is rejected, and audit record is created depending on the verification outcome. This description is consistent with TSS Section 7.5.3 of the [ST].

The TOE is not a distributed TOE.

[AGD] Section 6 describes the Root CA certificate used to verify the digital signature of the update embedded in the TOE software image. The [AGD] does not describe how this certificate can be installed or updated. This description is consistent with the description in the TSS.

2.43.3 Tests

Test Number	1
Test Objective	Verify the TOE is able to report the software version prior to an update and after an update.
Test Steps Performed	Check the TOE version using the local console. Verify that the version is reported.

	Apply a software update. Check the TOE version. Verify that a new version is reported.
Test Result	Pass; The TOE correctly permited the administrator to check the current version.

Test Number	2
Test Objective	Verify the TOE rejects incorrectly signed images, that are: 1. Modified 2. Not signed. 3. Have invalid signature The TOE does not delay activation.
Test Steps Performed	Modify the update image using a hex editor. Verify that the image is modified. Attempt an update using the modified image using REST API. Verify that the image is rejected. Attempt an update using a not signed update image. Verify that the image is rejected. Modify the update image signature block using a hex editor. Verify that the image is modified. Attempt an update using a modified image. Verify that the image is rejected.
Test Result	Pass; The TOE correctly rejected improperly signed update candidates.

Test Number	3
Test Objective	Verify the TOE update verification can be performed using the published hash.
Test Steps Performed	This test has been skipped as the TOE does not verify update files based on published hashes.
Test Result	Pass; The TOE did not implement a published hash verification.

2.44 FPT_TUD_EXT.2 Trusted Update Based on Certificates (Selection-Based)

TSS

The evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or guidance documentation) describes how the certificates are installed/updated/selected, if necessary.

The evaluator shall verify that the TSS describes how the TOE reacts if X.509 certificates are used for trusted updates and the Security Administrator attempts to perform the trusted update using an expired certificate.

The TSS shall describe the point at which revocation checking is performed and describe whether the Security Administrator can manually provide revocation information. It is expected that revocation checking is performed when a certificate is used when performing trusted updates. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device.

Guidance Documentation

The evaluator shall verify that the guidance documentation describes how the TOE reacts if X.509 certificates are used for trusted updates and the administrator attempts to perform the trusted update using an expired certificate. The evaluator shall verify any Security Administrator actions related to revocation checking, both accepting or rejecting certificates and manually providing revocation information. The description shall correspond to the description in the TSS.

Tests

The evaluator shall verify that the update mechanism includes a certificate validation according to FIA_X509_EXT.1 and a check for the Code Signing purpose in the extendedKeyUsage.

The evaluator shall digitally sign the update with an invalid certificate and verify that update installation fails. The evaluator shall digitally sign the application with a certificate that does not have the Code Signing purpose and verify that application installation fails. The evaluator shall repeat the test using a valid certificate and a certificate that contains the Code Signing purpose and verify that the application installation succeeds. The evaluator shall use a previously valid but expired certificate and verifies that the TOE reacts as described in the TSS and the guidance documentation. Testing for this element is performed in conjunction with the assurance activities for FPT_TUD_EXT.1.

The evaluator shall demonstrate that checking the validity of a certificate is performed at the time a certificate is used when performing trusted updates. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device.

2.44.1 TSS

[ST] Section 7.5.3 describes that the certificates used by the trusted update TSF are factory-installed in the TOE and cannot be modified. It describes that they are stored in plaintext.

[ST] Section 7.5.3 states that if TOE certificates related to trusted update are expired, the TSF will reject the update attempt. This is consistent with the selection in Section 6.1.5.8 FPT_TUD_EXT.2.3.

[ST] Section 7.3.6. describes that revocation status of certificates is checked upon an update attempt. If revocation information for certificates cannot be obtained the certificate is rejected. This is consistent with the selection in Section 6.1.5.8 FPT_TUD_EXT.2.2.

2.44.2 Guidance Documentation

[AGD] Section 8.8 describes that the x509 certificate is checked when the TOE performs a secure update. It describes that, if the certificate is invalid, the update will be rejected.

[AGD] Section 8.8 describes that for the software update process, the TOE will attempt to obtain revocation information from the CRL distribution point. If no revocation information is available, the update is rejected.

[AGD] Section 8.8 describes that if Digital signature verification is failed for any reason the image deemed invalid and the upgrade is aborted. This description is consistent with the description in the TSS.

2.44.3 Tests

Test Number	1
Test Objective	Verify the TOE will reject update signed with an invalid certificate. (Resolved by testing in FIA_X509_EXT.1.1/REV (Update)) Verify the TOE will reject an update signed with a certificate without Code Signing Key usage. Verify the TOE behaves as described in the ST when an update is signed with an expired certificate. (Not accept the certificate)
Test Steps Performed	Use an image signed by the certificate without CodeSigning key usage.

	Verify that the image is signed by certificate without CodeSigning Key usage. Attempt an update using this update image using REST API. Verify that the update is rejected. Change the system date on the TOE to be after the expiration date of the certificate used to sign update image. Verify that the the TOE date changed. Attempt to update using a normal update image. Verify that the Update is not accepted, certificate not accepted.
Test Result	Pass; The TOE correctly rejected update candidates which were signed by an invalid certificate.

2.45 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.

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.

Tests

The evaluator shall perform the following test:

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.

2.45.1 TSS

[ST] Section 7.6.2 describes that the local and remote sessions are terminated upon a configured inactivity timeout and requires the administrator to repeat the login process.

[ST] Section 7.6.2 describes that inactivity settings for the CLI are configurable from 30 to 72000 seconds.

2.45.2 Guidance Documentation

[AGD] Section 4.1.3 describes that the TOE supports a local session inactivity timeout ranging from 30 seconds to 120 minutes. [AGD] Section 8.5 describes how this is configured.

2.45.3 Tests

Test Number	1
Test Objective	Verify the TOE enforces session timeout limits and supports configuring timeout limits for local sessions.
Test Steps Performed	Use the local menu to configure 120 sec as local session timeout per [AGD]. Verify that the new timeout is set.

	Wait 120 seconds. Verify that the session is terminated. Login back in and configure session timeout to be 170 seconds. Verify that a timeout is configured. Wait 170 seconds. Verify that the session is terminated. Attempt to resume session. Verify that Authorization is required.
Test Result	Pass; The TOE enforced session timeouts when configured to do so.

2.46 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.

Guidance Documentation

The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

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.

2.46.1 TSS

[ST] Section 7.6.2 describes that local and remote session are terminated upon a configured inactivity timeout and requires the administrator to repeat the login process. [ST] Section 7.6.2 describes that inactivity settings for the RMI are configurable from 30 to 72000 seconds.

2.46.2 Guidance Documentation

[AGD] Section 4.1.3 describe that TOE supports remote session inactivity timeout in range from 30 seconds to 120 minutes. [AGD] section 8.5 describes how this is configured.

2.46.3 Tests

Test Number	1
Test Objective	Verify the TOE enforces session timeout limits and supports configuring timeout limits for remote sessions.
Test Steps Performed	Use RestAPI to configure Rest API session timeout to be 170 seconds using system/inactivity_timeout access point. Verify that Configuration is accepted. Query the value using REST API /system/inactivity_timeout access point. Verify that the value matches the configured values above. Wait for the configured time period to expire and attempt to use the session.

	Verify that session is terminated. Authenticate using Rest API and request remote session timeout. Verify that the command is successful, timeout value reported as 300 seconds. Wait 300 seconds and repeat the request. Verify that the request is denied as the session is expired.
Test Result	Pass; The TOE correctly applied the session termination function when configured to do so.

2.47 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.

Guidance Documentation

The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

Tests

For each method of remote administration, the evaluator shall perform the following tests:

a) Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

b) Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

2.47.1 TSS

[ST] Section 7.6.3 describes that a remote administrative session can be terminated by issuing a delete request on a login token.

[ST] Section 7.6.3 describes that the local session can be terminated in the appropriate menu prompt using ‘q’ command.

2.47.2 Guidance Documentation

[AGD] Section 4.1.3 describes that remote session can be terminated by using DELETE command on /token Rest API node.

[AGD] Section 4.1.4 describes how local session termination can be achieved.

2.47.3 Tests

Test Number	1
Test Objective	Verify the TOE supports user-initiated session termination for local user sessions.
Test Steps Performed	Login to a local interactive session. Use menu option “Exit and logout”. Verify that the session is terminated.

Test Result	Pass; The TOE correctly permited users to end their own sessions.

Test Number	2
Test Objective	Verify the TOE supports user-initiated session termination for local user sessions.
Test Steps Performed	Use Rest API to authenticate and establish a session using a session token using /token access point. Verify that an Authorization token is created. Verify session token using /token_test access point. Verify that the Token is accepted. Use DELETE operation on /token access point using REST API. Verify that the Command is accepted. Verify session token using /token_test and system/inactivity_timeout access point. Verify that the Token is rejected.
Test Result	Pass; The TOE correctly permited users to end their own sessions.

2.48 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).

Guidance Documentation

The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

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.

2.48.1 TSS

[ST] Section 7.3.3 describes that the administrator can access the TOE using the local console and remote management interface via TLS/HTTPS. [ST] Section 7.6.1 states that the TOE displays a configurable advisory and consent message for the remote management interface and local interface.

2.48.2 Guidance Documentation

[AGD] Section 8.2 describes how the login banner can be configured using the local console.

2.48.3 Tests

Test Number	1
Test Objective	Verify the consent banner is configurable and is displayed for each method of administration: • Local CLI • Remote Rest API
Test Steps Performed	Authenticate using the local CLI interface, enter the Login banner menu, use clb command to configure login banner. Verify that the banner configuration menu is displayed. Configure the login banner to say "Viasat system under evaluation by UL VS". Verify that the configuration is successful. Log out and attempt to establish a new local session. Verify that the new banner is displayed. Attempt to login using: POST on /token?banner_accepted=true access point. Verify that the new banner is displayed. Explicitly request consent banner using /login_banner access point. Verify that the new banner is displayed.
Test Result	Pass; The TOE correctly permitted the administrator to configure an advisory and consent banner, and correctly displayed that banner at the administrative interfaces.

2.49 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 secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

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.

Tests

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

c) Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

d) Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

   The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

   The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

   In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

2.49.1 TSS

[ST] Section 7.7.1 describes that the TOE performs the protected communications with the following authorized entities:

- VNP Gateway Peers, acting as a peer, with authentication using x509 certificates,
- Remote Audit server (syslog), acting as a TLS client, with authentication using x509 certificates.

This description is consistent with the selections in FTP_ITC.1 and FTP_ITC.1/VPN and are sufficient to match those protocols with IPsec Protocol and TLS Client SFRs.

2.49.2 Guidance Documentation

The TOE supports the trusted channels to syslog server and peer IPsec Gateway authorized IT entities.

[AGD] Section 5.1 contain instructions for establishing connection to remote syslog server. [AGD] Section 5.1.1 describes that the service will attempt to resume the connection if connection fails or is broken.

[AGD] Section 7.2 describes how to establish a connection to the IPsec peer gateway. [AGD] Section 7 describes the TOE behavior when connection to the peer is lost and guidance to re-establish the connection when reconnection fails.

2.49.3 Tests

Test Number	1
Test Objective	Verify the TOE is able to establish a trusted channel communication as described. (Remote audit server, IPsec gateway connection).
Test Steps Performed	This test is resolved by evaluation activities in FCS_IPSEC_EXT.1 and TCS_TLSC_EXT.1.
Test Result	Pass; The TOE correctly established trusted channels.

Test Number	2
Test Objective	Verify the TOE is able to establish a trusted channel communication as described.
Test Steps Performed	This test is resolved by Test 4.
Test Result	Pass; The TOE correctly established trusted channels.

Test Number	3
Test Objective	Verify the TOE is able to establish an encrypted trusted channel communication to Remote audit server. (IPsec gateway connection is resolved by testing in FCS_IPSEC_EXT.1.1 Test 1).
Test Steps Performed	Establish a trusted channel to the remote audit server. Inspect packets on the wire. Verify that all Packets are encrypted.
Test Result	Pass; The TOE correctly encrypted trusted channel communications.

Test Number	4
Test Objective	Verify the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities: • Remote audit server • IPsec Peer Gateway
Test Steps Performed	Re-configure the TOE using Rest API so that the remote audit server would be host 10.3.2.6 with a physical connection between the TOE and middlehost. Verify that the remote audit server is configured on the TOE. Configure TOE to log network packets traversing the TOE, initiate a stream of packets to be logged to generate stream of audit records. Verify that records are being generated and sent to the Syslog server. Interrupt the physical connection to syslog server for 4 minutes. Restore the connection. Verify that TLS packets continue to be sent encrypted using same session. Interrupt the physical connection for 16 minutes. Restore the connection. Verify that the TLS session will be re-established, no plaintext packets are sent. For an Ipsec connection, initiate a stream of data from lefthost to righthost.

	Initiate packet capture on a channel and righthost. Ensure that data is being forwarded by the TOE. Physically interrupt the connection between the TOE and the TestGW for 15 seconds and observe the traffic. Verify that the Traffic is interrupted then resumed encrypted using the same session. Initiate stream of packets, interrupt physical connection for 180 seconds. Verify that the connection is renegotiated, no packets are sent in plaintext.
Test Result	Pass; The TOE correctly re-initiated the trusted channel when the connection was broken.

2.50 FTP_ITC.1/VPN Inter-TSF Trusted Channel (VPN Communications) [MOD]

TSS
The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

Guidance
The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

Tests
The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications. Additional testing for IPsec is covered in FCS_IPSEC_EXT.1.

2.50.1 TSS

This work unit is resolved by the activity in FTP_ITC.1

2.50.2 Guidance Documentation

This work unit is resolved by FTP_ITC.1.

2.50.3 Tests

Test Number	1
Test Objective	None defined.
Test Steps Performed	Resolved by testing FTP_ITC.1.
Test Result	Pass; the TOE implemented the correct behaviour.

2.51 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.

Guidance Documentation

The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

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.

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

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

2.51.1 TSS

[ST] Section 7.3.3 describes the trusted path as provided by the TLS/HTTPS interface. This is consistent with the requirement in the FTP_TRP.1/Admin and selection-based requirement FCS_TLSS_EXT.1 included in the [ST].

[ST] Section 7.7.2 describes that the remote management interface is a REST API interface, accessible through HTTPS using API commands listed in the TOE guidance documentation.

2.51.2 Guidance Documentation

The TOE supports one remote administration method – the REST API remote management interface. [AGD] Sections 3.1 and 3.1.1 describe how the remote administrative session using this interface can be established by obtaining an authorization token.

2.51.3 Testing

Test Number	1
Test Objective	Verify that each remote administration method is tested. [ST] describes Rest API as the only remote administration method.
Test Steps Performed	Rest API was tested as part of testing FCS_TLSS_EXT.1.
Test Result	Pass; The TOE permitted administration via restAPI.

Test Number	2
Test Objective	Verify that each remote administration method is tested. [ST] describes Rest API as the only remote administration method.
Test Steps Performed	Establish a Rest API administrative session. Inspect packets on the wire. Packets are encrypted.
Test Result	Pass; The TOE correctly encrypted the administrative session.

3 SAR Assurance Activities and Results

3.1 ASE: Security Target Evaluation

3.1.1 General ASE:

When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2 (Evaluation Activities for SFRs).

Result:
For TSS EAs for SFRs, see Section 2 above.

3.1.2 ASE_TSS.1.1C:

The evaluator shall examine the TSS to determine that it is clear which TOE components contribute to each SFR or how the components combine to meet each SFR.

The evaluator shall verify the sufficiency to fulfil the related SFRs. This includes checking that the TOE as a whole fully covers all SFRs and that all functionality that is required to be audited is in fact audited regardless of the component that carries it out.

Result: The evaluator examined the TOE and verified that the TOE is not distributed. The TOE platform performs all SFRs. All SFRs were evaluated, and all required functionality was audited.

3.2 ADV: Development

3.2.1 Basic Functional Specification (ADV_FSP.1)

The EAs for this assurance component focus on understanding the interfaces (e.g., application programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1- 1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional “functional specification” documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

3.2.2 ADV_FSP.1-1

PP Evaluation Activity:
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.
Result: Pass, the evaluator examined the interface documentation and found that all TSFI were identified.

3.2.3 ADV_FSP.1-2

PP Evaluation Activity:
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.
Result: Pass, the evaluator examined the interface documentation and found that it described the purpose and method of access for each TSFI.

3.2.4 ADV_FSP.1-3

PP Evaluation Activity:
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.
Result: Pass, the evaluator examined the interface documentation to verify that it describes all relevant parameters for each TSFI.

3.2.5 ADV_FSP.1-4

PP Evaluation Activity:
Paragraph 561 from the CEM: “In the case where the developer has provided adequate documentation to perform the analysis called for by the rest of the work units for this component without explicitly identifying SFR-enforcing and SFR-supporting interfaces, this work unit should be considered satisfied.”

Since the rest of the ADV_FSP.1 work units will have been satisfied upon completion of the EAs, it follows that this work unit is satisfied as well.
Result: Pass

3.2.6 ADV_FSP.1-5

PP Evaluation Activity:
The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.
Result: Pass; all TSFI were exercised during the course of the evaluation. A complete mapping was provided in the AGD section.

3.2.7 ADV_FSP.1-6

PP Evaluation Activity:
EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3 and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e. at the TSFI) are covered. Therefore, the intent of this work unit is covered.
Result: Pass

3.2.8 ADV_FSP.1-7

PP Evaluation Activity:
EAs that are associated with the SFRs in Section 2, and, if applicable, Sections 3 and 4, are performed to ensure that all the SFRs where the security functionality is externally visible (i.e. at the TSFI) are

addressed, and that the description of the interfaces is accurate with respect to the specification captured in the SFRs. Therefore, the intent of this work unit is covered.
Result: Pass

3.3 AGD: Guidance Documents

3.3.1 Operational User Guidance (AGD_OPE.1)

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

PP Evaluation Activities:
The evaluator shall ensure the Operational guidance documentation is distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

In addition the evaluator shall ensure that the following requirements are also met.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.
b) The documentation must describe the process for verifying updates to the TOE 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[3]:
    i. 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).
    ii. 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.

Result: Pass, the evaluator examined the AGD, and specific findings are included in the sections above in this document.

3.3.2 Preparative Procedures (AGD_PRE.1)

PP Evaluation Activities:

[3] The Evaluation Activities was modified by TD0536

The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

In addition the evaluator shall ensure that the following requirements are also met.

The preparative procedures must:

a) include instructions to provide a protected administrative capability; and
b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

Result: Pass; the evaluator examined the provided AGD and verified that all preparative procedures are identified. The AGD as a whole contains suitable instruction for the administrator to install the TSF, manage the TSF as part of an operational environment, and configure the TSF into the evaluated configuration.

3.4 ALC: Life-cycle Support

3.4.1 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.
Result: Pass, the TOE provides unique labels.

3.4.2 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.
Result: Pass, the TOE is listed in the vendor's product management system and lifecycle.

3.5 ATE: Tests

3.5.1 Independent Testing (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.

The evaluator should consult Appendix 709 when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation. Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section A.9.3.1.

Result: Pass, the evaluator conducted all test assurance activities and found them passing; specific verdicts and details are listed above.

3.6 AVA: Vulnerability Assessment

3.6.1 Vulnerability Survey (AVA_VAN.1)

While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

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

The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

Result: Pass; The evaluator conducted the vulnerability analysis. the results of which are documented in the separate Vulnerability Analysis document. The evaluator used NVD, cvedetails.com as a source for CVE search. The search was performed on 2023-08-01, 2023-10-11, 2023-10-23, and 2023-11-22. The vendor applied fixes to the identified applicable vulnerabilities, so there are no residual exploitable vulnerabilities present in the TOE.

4 References

Abbr.	Name	Version	Date
[PP]	collaborative Protection Profile for Network Devices	2.2e	March 23, 2020
[MOD]	PP-Module for VPN Gateways	1.2	March 31, 2022
[SD1]	Supporting Document – Mandatory Technical Document Evaluation Activities for Network Device cPP	2.2	December 2019
[SD2]	Supporting Document Mandatory Technical Document PP-Module for VPN Gateways	1.2	Mach 31, 2022
[ST]	Viasat Secure VPN v1.1.7 Security Target	2.6	December 13, 2023
[AGD]	Viasat Secure VPN User Guide	008	December 15, 2023
[RAR]	Viasat Secure VPN, version 1.1.7, REST API Reference	005	September 6, 2023