www.GossamerSec.com

Assurance Activity Report for Versa Networks Versa Secure SD-WAN Versa Operating System (VOS) 22.1 running on Versa CSG1500, CSG2500, CSG3500, CSG5000, Dell PowerEdge R7515, and Dell VEP4600, Versa Director 22.1 and Versa Analytics 22.1

Version 0.2
03/28/24

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

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

Document: AAR-VID11431

Revision History

Revision	Date	Authors	Summary
Version 0.1	03/05/24	John Messiha	Initial draft
Version 0.2	03/28/24	John Messiha	Addressed ECR comments

The TOE Evaluation was Sponsored by:

Versa Networks, Inc.
2550 Great America Way #350
Santa Clara, CA 95054

Evaluation Personnel:

- John Messiha

Common Criteria Versions:

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

Common Evaluation Methodology Versions:

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

Table of Contents

1. Introduction

This document presents evaluations results of the Versa networks versa secure SD-wan versa operating system (vos) 22.1 running on csg1500, csg2500, csg3500, csg5000, dell poweredge r7515, and dell vep4600, versa director 22.1, and versa analytics 22.1 NDcPP22e/IPS10/STFFW14e/VPNGW13 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

This section explains why the test subset was adequate to address all product installations.

1.1.1 Evaluated Platform Equivalence

The TOE is the Versa Secure SD-WAN Versa Operating System (VOS) 22.1 running on CSG1500, CSG2500, CSG3500, CSG5000, Dell PowerEdge R7515, and Dell VEP4600, Versa Director 22.1, and Versa Analytics 22.1.

Referring to Figure 1 in the Security Target for a perspective on the TOE's distributed components, the TOE's two components include the (1) Versa Headend and the (2) VOS Branch(es).

1. Versa planned to evaluate the Versa Headend as a vND, tested upon a single combination of hardware and hypervisor. Gossamer tested this single combination; thus, no equivalence claims are needed.
2. Versa planned to evaluate the VOS Branch as both a vND and pND.

    a. For the VOS Branch as a vND, Gossamer tested it as a single combination of hardware and hypervisor, so again, no equivalence claims are needed
    b. For the VOS Branches as pNDs, the evaluation includes eight different hardware platforms, with some tested and the remainder claimed as equivalent. The remainder of this section provides the equivalency argument between the tested and equivalent hardware.

VOS Branches pND Image equivalency

Each VOS Branch pND model uses the same image, which includes Ubuntu 18.04.6 LTS as the underlying OS and includes Intel networking drivers (as all VOS Branch models exclusively use Intel networking chipsets and drivers).

The following Table describes the salient details of the eight VOS Branch pNDs and the list of tested platforms and equivalency claims.

VOS Branch pND Equivalency Matrix

The evaluator tested on the following platforms and microarchitectures. They are marked in green in the table below.

- Dell PowerEdge R7515 - AMD EPYC 7713P (Zen 3)
- Dell VEP4600 - Intel Xeon-D 2187NT (Skylake)
- KVM-VOS (virtual Branch) on Ubuntu 18.04 with KVM - Intel (R) Xeon (R) D-1587 (Broadwell)
- Versa Headend on ESXi 7.0 on Intel (R) Xeon (R) D-1587 (Broadwell)
    - Versa Director VM
    - Versa Analytics VM

- Versa VOS SD-WAN Controller (VOS) VM

In addition to the tested devices, Versa received algorithm certificates on all claimed microarchitectures for completeness. The microarchitectures that have corresponding algorithm certificates are marked in blue in the table below.

	CSG5000	Dell R7515	Dell VEP4600	CSG3500	CSG2500	CSG1500	Versa Headend	KVM-VOS (virtual Branch)
CPU	AMD EPYC 7713P	AMD EPYC 7713P	Intel Xeon-D 2187NT	Intel Xeon-D 2177NT	Intel Xeon Gold 6252N	Intel Xeon-D 2177NT	Intel Xeon-D 1587	Intel Xeon-D 1587
Micro-architecture	Zen 3	Zen 3	Skylake	Skylake	Cascade Lake	Skylake	Broadwell	Broadwell
Networking, NIC vendor	Intel	Intel	Intel	Intel	Intel	Intel	Intel	Intel
Image Tested?	Common	Common	Common	Common	Common	Common	Common	Common
CAVP	Equivalent	Yes	Yes	Equivalent	Yes	Equivalent	Yes	Yes
NDcPP	Equivalent	Yes	Yes	Equivalent	Equivalent	Equivalent	Yes	Yes
VPNGW	Equivalent	Yes	Yes	Equivalent	Equivalent	Equivalent	Yes	Yes
FFW	Equivalent	Yes	Yes	Equivalent	Equivalent	Equivalent	Yes	Yes
IPS	Equivalent	Yes	Yes	Equivalent	Equivalent	Equivalent	Yes	Yes

1.1.2 CAVP EQUIVALENCE

The TOE includes multiple cryptographic modules across the range of TOE components. The CAVP-certified cryptographic modules of the TOE are listed in the table below. The table below also lists the CAVP certificate numbers for each CAVP-certified cryptographic module.

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
Versa Java Crypto Module					
AES	FCS_COP.1/DataEncryption	FIPS PUB 197 CBC	• Direction: Decrypt, Encrypt • Key Length: 128, 256	OpenJDK 11 on Ubuntu 18.04 on ESXi 7.0 on Intel (R) Xeon (R) D-1587 with AES-NI	A5145
	FCS_COP.1/DataEncryption	NIST SP 800-38D GCM	• Direction: Decrypt, Encrypt • IV Generation: External • Key Length: 128, 256 • Tag Length: 96, 128 • IV Length: 96 • Payload Length: 64, 128, 192 • AAD Length: 128, 256		

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
ECDSA	FCS_CKM.1 FCS_COP.1/SigGen	FIPS PUB 186-4 KeyGen KeyVer SigGen SigVer	• Curve: P-256, P-384, P-521 • Secret Generation Mode: Extra Bits, Testing Candidates • Hash Algorithm: SHA2-256, SHA2-384, SHA2-512		
DRBG	FCS_RBG_EXT.1	NIST SP 800-90A HMAC_DRBG	• Prediction Resistance: Yes • Supports Reseed • Mode: SHA2-512 • Entropy Input: 512 • Nonce: 512 • Personalization String Length: 512 • Additional Input: 512 • Returned Bits: 2048		
HMAC	FCS_COP.1/KeyedHash	FIPS PUB 198-1 HMAC-SHA-1 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512	• MAC: 256, 384, 512 Increment 8 • Key Length: 8-2048 Increment 8		
KAS-ECC	FCS_CKM.2	SP 800-56Arev3	• Domain Parameter Generation Methods: P-256, P-384, P-521 • Function: Key Pair Generation, Partial Validation • Scheme: ephemeralUnified: • KAS Role: Initiator, Responder		

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
			• KDF Methods: oneStepKdf: • Auxiliary Function Methods: ○ SHA-256, SHA-384, SHA-512 • MAC Salting Methods: default, random • Fixed Info Pattern: algorithmId||||uPartyInfo||vPartyInfo • Fixed Info Encoding: Concatenation • Key Length: 256		

KAS-FFC-SSC	FCS_CKM.1, FCS_CKM.2	SP 800-56Arev3	• Domain Parameter Generation Methods: FC, MODP-2048, MODP-4096, MODP-8192
			• Scheme:
			  ○ dhEphem
			  ▪ KAS Role: initiator, responder		

RSA	FCS_CKM.1
	
	FCS_COP.1/SigGen
		FIPS PUB 186-4
		
		KeyGen
		
		SigGen
		
		SigVer
			• Key Generation Mode: B.3.3
			  ○ Properties:
			  ▪ Modulo: 2048, 3072
			  ▪ Primality Tests: Table C.2
			• Public Exponent Mode: Random
			• Private Key Format: Chinese Remainder Theorem
			
			• Signature Type: PKCS 1.5, PKCSPSS
			• Properties:
			  ○ Modulo: 2048, 3072
			  ▪ Hash Algorithm: SHA2-256,		

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
			SHA2-384, SHA2-512		
SHS	FCS_COP.1/Hash	FIPS PUB 180-4 SHA-1 SHA2-256 SHA2-384 SHA2-512	• Message Length: 0-65536 Increment 8		
VOS TLS Cryptographic Module					
AES	FCS_COP.1/DataEncryption	FIPS PUB 197 CBC, CTR	• Direction: Decrypt, Encrypt • Key Length: 128, 256	Ubuntu 18.04 on AMD EPYC 7713P with AES-NI	A5144
	FCS_COP.1/DataEncryption	GCM	• Direction: Decrypt, Encrypt • IV Generation: Internal • IV Generation Mode: 8.2.1 • Key Length: 128, 256 • Tag Length: 32, 64, 96, 104, 112, 120, 128 • IV Length: 96, 1024 • Payload Length: 504, 512, 1016, 1024 • AAD Length: 0, 504, 512, 1016, 1024	Ubuntu 18.04 on ESXi 7.0 on Intel (R) Xeon (R) D-1587 with AES-NI (Versa Controller)	
ECDSA	FCS_CKM.1/IKE FCS_COP.1/SigGen	FIPS PUB 186-4 KeyGen KeyVer SigGen	• P-256, P-384, P-521 • Secret Generation Mode: Extra Bits, Testing Candidates • Curve: P-256, P-384, P-521 • Hash Algorithm: SHA2-256, SHA2-384, SHA2-512	Ubuntu 18.04 on ESXi 7.0 on Intel (R) Xeon (R) D-1587 with AES-NI (Versa Director)	

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
		SigVer		Ubuntu 18.04 on Intel (R) Xeon (R) D-2187NT with AES-NI Ubuntu 18.04 on Intel (R) Xeon (R) Gold 6252N with AES-NI Ubuntu 18.04 on KVM on Ubuntu 18.04 on Intel (R) Xeon (R) D-1587 with AES-NI	
DRBG	FCS_RBG_EXT.1	SP 800-90A CTR_DRBG	• Mode: AES-256 • Derivation Function Enabled: Yes • Additional Input: 0-256 • Entropy Input: 256 • Nonce: 128 • Personalization String Length: 0-256 • Returned Bits: 128		
HMAC	FCS_COP.1/KeyedHash	FIPS PUB 198-1 HMAC-SHA-1 HMAC-SHA2-256 HMAC-SHA2-384 HMAC-SHA2-512	• MAC: 160, 256, 384, 512 • Key Length: 8-512000 Increment 8		

KAS-ECC	FCS_CKM.1/IKE
	
	FCS_CKM.2
		SP 800-56Arev3	• Domain Parameter Generation Methods: P-256, P-384
			• Scheme:
			  ○ ephemeralUnified:
			  ▪ KAS Role: initiator, responder		

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate

RSA	FCS_CKM.1
	
	FCS_COP.1/ SigGen
		FIPS PUB 186-4
		
		KeyGen
		
		SigGen
		
		SigVer
			• Key Generation Mode: B.3.3
			  ○ Properties:
			  ▪ Modulo: 2048, 3072
			  ▪ Primality Tests: Table C.2
			• Public Exponent Mode: Random
			• Private Key Format: Standard
			
			• Signature Type: PKCS 1.5, PKCSPSS
			• Properties:
			  ○ Modulo: 2048, 3072
			  ▪ Hash Algorithm: SHA2-256, SHA2-384, SHA2-512		

SHS	FCS_COP.1/Hash	FIPS 180-4 SHA-1 SHA2-256 SHA2-384 SHA2-512	• Message Length: 0-51200 Increment 8		
VOS IPsec Cryptographic Module					

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
AES	FCS_COP.1/Data Encryption	FIPS PUB 197 CBC, CTR	• Direction: Decrypt, Encrypt • Key Length: 128, 256	Ubuntu 18.04 on AMD EPYC 7713P with AES-NI Ubuntu 18.04 on ESXi 7.0 on Intel (R) Xeon (R) D-1587 with AES-NI	A5147
	FCS_COP.1/Data Encryption	GCM	• Direction: Decrypt, Encrypt • IV Generation: Internal • IV Generation Mode: 8.2.1 • Key Length: 128, 256 • Tag Length: 64, 96, 128 • IV Length: 96 • Payload Length: 504-896 Increment 8 • AAD Length: 256-1024 Increment 8	Ubuntu 18.04 on Intel (R) Xeon (R) D-2187NT with AES-NI Ubuntu 18.04 on Intel (R) Xeon (R) Gold 6252N with AES-NI	
ECDSA	FCS_COP.1/SigGen	FIPS PUB 186-4 SigGen SigVer	• P-256, P-384, P-521 • Secret Generation Mode: Extra Bits, Testing Candidates • Curve: P-256, P-384, P-521 • Hash Algorithm: SHA2-256, SHA2-384, SHA2-512	Ubuntu 18.04 on KVM on Ubuntu 18.04 on Intel (R) Xeon (R) D-1587 with AES-NI	
HMAC	FCS_COP.1/KeyedHash	FIPS PUB 198-1 HMAC-SHA-1 HMAC-SHA2-256	• MAC: 160, 256, 384, 512 • Key Length: 8-512000 Increment 8		

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
		HMAC-SHA2-384 HMAC-SHA2-512			

KAS-ECC-SSC	FCS_CKM.2	SP 800-56Arev3	• Domain Parameter Generation Methods: P-256, P-384
			• Scheme:
			  ○ ephemeralUnified:
			  ▪ KAS Role: initiator, responder		

RSA	FCS_COP.1/ SigGen	FIPS PUB 186-4
		
		SigGen
		
		SigVer
			• Key Generation Mode: B.3.3
			  ○ Properties:
			  ▪ Modulo: 2048, 3072
			  ▪ Primality Tests: Table C.2
			• Public Exponent Mode: Random
			• Private Key Format: Standard
			
			• Signature Type: PKCS 1.5
			• Properties:
			  ○ Modulo: 2048, 3072
			  ▪ Hash Algorithm: SHA2-256, SHA2-384, SHA2-512		

SHS	FCS_COP.1/Hash	FIPS 180-4 SHA-1 SHA2-256	• Message Length: 0-51200 Increment 8		

Algorithm	Requirement (SFR)	Mode/Method	Capabilities	Operational Environment	CAVP Certificate
		SHA2-384 SHA2-512			

1.2 References

The following evidence was used to complete the Assurance Activities:

- NDcPP ST Versa Networks v1.9, Version 1.9, March 28, 2024 **(ST)**
- Configure for NIAP Common Criteria, Version 1.0, March 29, 2024 **(Admin Guide)**

2. Protection Profile SFR Assurance Activities

This section of the AAR identifies each of the assurance activities included in the claimed Protection Profiles and describes the findings in each case.

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For the administrative task of generating/import of, changing, or deleting of cryptographic keys as defined in FAU_GEN.1.1c, the TSS should identify what information is logged to identify the relevant key.

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

Section 7.1.1 of the ST states Audit messages are generated for actions performed by users of the TOE. The VOS Branch devices and the SD-WAN Controllers generates alarms for critical event logs, and audit logs for any administrative operations performed on the TOE. The VOS and Director devices send their logs to an Analytics node.

It also contains the following table which demonstrates which events are generated and recorded by each TOE component:

Log Type	Component	Description
Audit start/stop	VOS, Director, Analytics	Startup and shutdown of appliance and/or auditing function
Logon/Logoff events	Director	Success and failed administrator logins, lockout events, session timeouts, session terminations
Config/Management operations	Director	All management functions
NTP logs	VOS, Director, Analytics	Clock synchronization, server configuration
Firewall logs	VOS	Firewall logs, netflows, rule configuration
IDP logs	VOS	Threat logs, signature-based matches, anomaly-based matches, IP filter matches, rule configuration
IPsec VPN logs	VOS, Director, Analytics	IPSec session start/stop, failures
HTTPS logs	Director	HTTPS session start/stop, failures
SSH logs	Director	SSH session start/stop, failures
Device alarms	VOS	Critical device errors and alarms

Software upgrades	VOS, Director, Analytics	Initiation and result (success or failure)
Registration/ deregistration	VOS, Director	Adding or removing a branch site or TOE component
X.509 logs	VOS, Director, Analytics	Failure to validate, Import of CA trust anchor
Key generation	VOS, Director, Analytics	IPsec, SSH, and TLS key generation events. Key is identified by key name record or filename.
Self-tests	VOS, Director, Analytics	Self-test outcomes (success or failure)

Note the table mentions for key related events, the key is identified by key name record or filename.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation and ensure that it provides an example of each auditable event required by FAU_GEN.1 (i.e. at least one instance of each auditable event, comprising the mandatory, optional and selection-based SFR sections as applicable, shall be provided from the actual audit record).

The evaluator shall also make a determination of the administrative actions related to TSF data related to configuration changes. The evaluator shall examine the guidance documentation and make a determination of which administrative commands, including subcommands, scripts, and configuration files, are related to the configuration (including enabling or disabling) of the mechanisms implemented in the TOE that are necessary to enforce the requirements specified in the cPP. The evaluator shall document the methodology or approach taken while determining which actions in the administrative guide are related to TSF data related to configuration changes. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

Section “Audit Events” in the **Admin Guide** states that each appliance generates an audit event for each user interaction with the web interface. Each event includes at least a timestamp, the username of the user whose action generated the event, a source IP, and text describing the event. This section provides a table identifying all

of the required audit events consistent with the ST along with a sample of each record for each TOE component and SFR where applicable.

From a review of the ST, the Guidance and through testing the evaluator also determined that the guidance contains all of the administrative actions and their associated audit events that are relevant to the PP and to use of the TOE. These administrative actions are consistent with the security requirements implemented in the TOE and were found to have appropriate management capabilities identified in the guidance documentation.

Component Testing Assurance Activities: The evaluator shall test the TOE's ability to correctly generate audit records by having the TOE generate audit records for the events listed in the table of audit events and administrative actions listed above. This should include all instances of an event: for instance, if there are several different I&A mechanisms for a system, the FIA_UIA_EXT.1 events must be generated for each mechanism. The evaluator shall test that audit records are generated for the establishment and termination of a channel for each of the cryptographic protocols contained in the ST. If HTTPS is implemented, the test demonstrating the establishment and termination of a TLS session can be combined with the test for an HTTPS session. When verifying the test results, the evaluator shall ensure the audit records generated during testing match the format specified in the guidance documentation, and that the fields in each audit record have the proper entries.

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

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

The evaluator created a list of the required audit events. The evaluator then collected the audit event when running the other security functional tests described by the protection profiles. For example, the required event for FPT_STM_EXT.1 is Changes to Time. The evaluator collected these audit records when modifying the clock using administrative commands. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.2 Security Audit Data Generation (STFFW14E:FAU_GEN.1)

2.1.2.1 STFFW14E:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

No additional Evaluation Activities are specified.

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

Section "Audit Events" of the **Admin Guide** states that the appliances that are part of the TOE generate an audit record for each user interaction with the web interface, and also record system status messages in the system log. For the CLI, the appliance also generates an audit record for every command executed. Each event includes at least a timestamp, the username of the user whose action generated the event, a source IP, and text describing the event. This section provides a table identifying all of the required audit events consistent with the ST along with a sample of each record for each TOE component and SFR where applicable. The evaluator verified that all audit records specified in Table 2 of the PP-Module and those required by the Base-PP are described.

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

The evaluator created a list of the required audit events. The evaluator then collected the audit event when running the other security functional tests described by the protection profiles. For example, the required events for FFW_RUL_EXT.1 are indications that an access-list rule is processed and the traffic is handled accordingly. The evaluator sends traffic matching access-list rules and ensures that the TOE logs the events. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.3 Audit Data Generation (IPS) - per TD0595 (IPS10:FAU_GEN.1/IPS)

2.1.3.1 IPS10:FAU_GEN.1.1/IPS

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.3.2 IPS10:FAU_GEN.1.2/IPS

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

Section 7.1.1 of the ST states LEF profiles may be associated with IP-filtering profiles, NGFW access policies, DoS protection profiles, and vulnerability profiles, which ensure that the matching traffic is logged.

Thresholds may be configured on a vulnerability profile according to the following:

- Track By—Select the threshold tracking based on either source address, destination address, or both source and destination addresses.
- Interval—Enter an interval, in seconds.
- Threshold—Enter the number of hits per interval based on the traffic direction.

Signature based detection applies a set of pre-defined rules or custom rules. Rules are based on the snort rule format. Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information, and a series of customizable rule options which cover all the fields described in IPS_SBD_EXT.1.

The same section also states that the following traffic types will result in a counter increment:

- IPv4 packets containing specific ID, FLAGS, FRAG, TTL, CHECKSUM, SRC, DST, or OPTIONS header field values.
- ICMP packets containing specific TYPE, CHECKSUM, or CODE header field values.
- TCP packets containing specific SPORT, RESERVED, FLAGS, or OPTIONS header field values.
- UDP packets containing specific SPORT header field values
- IPv6 packets containing specific PLEN header field value, or containing two routing headers
- ICMPv6 packets containing specific CODE or CHECKSUM header field values.
- ICMPv4 and ICMPv6 payloads containing a detection string.
- IPv4 packets with same SRC and DST IP
- TCP packets containing NULL flag
- TCP packets containing SYN + FIN flags
- TCP packets containing FIN + ONLY flags
- TCP packets containing SYN + RST flags
- ICMP or IPv4 packet containing IP Fragment Overlap

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

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

Section “Configure Intrusion Detection and Prevention” in the **Admin Guide** describes the creation of IPS events and what options are available to a system administrator to monitor or react to rules being triggered. It also describes how to enable or disable a rule and specifies that when an event is enabled (either to log or log and drop), a log will be generated each time the policy is triggered. If an event is disabled, no log will be generated. Subsection “configure DoS protection settings” describes how to modify intrusion event thresholds including a count of events and timeframe of events.

Component Testing Assurance Activities: The evaluator shall test that the interfaces used to configure the IPS polices yield expected IPS data in association with the IPS policies. A number of IPS policy combination and ordering scenarios need to be configured and tested by attempting to pass both allowed and anomalous network traffic matching configured IPS policies in order to trigger all required IPS events. Note the following:

- This activity should have been addressed with a combination of the Test EAs for the other IPS requirements.

- As part of testing this activity, the evaluator shall also ensure that the audit data generated to address this SFR can be handled in the manner that FAU_STG_EXT.1 requires for all audit data.

This requirement has been satisfied by the collecting of audits during IPS testing in IPS_SBD_EXT.1, IPS_ABD_EXT.1, and IPS_IPB_EXT.1. The audit events specified by FAU_GEN.1/IPS were all collected and recorded in the DTR in conjunction with the auditable events collected and recorded as part of NDcPP22e:FAU_GEN.1. Trusted channel communications with an external syslog server and the TOE behavior when local audit storage is exceeded were tested as part of FAU_STG_EXT.1. All of the audits captured were collected off the remote syslog server which received the audits from the TOE devices encrypted over the trusted channel in accordance with FAU_STG_EXT.1.

2.1.4 Audit Data Generation (VPN Gateway) (VPNGW13:FAU_GEN.1/VPN)

2.1.4.1 VPNGW13:FAU_GEN.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 VPNGW13:FAU_GEN.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

Refer to NDcPP22e:FAU_GEN.1 which describes the audit behavior of the TOE. The VPN functionality uses the same audit mechanism.

Refer to NDcPP22e:FAU_STG_EXT.1 which describes how the audits are transmitted over ipsec to a remote audit server. VPN related audits use the same channel for protecting audits.

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

Section "Audit Events" in the **Admin Guide** states that each appliance generates an audit event for each user interaction with the web interface. Each event includes at least a timestamp, the username of the user whose action generated the event, a source IP, and text describing the event. This section provides a table identifying all of the required audit events consistent with the ST along with a sample of each record for each TOE component and SFR where applicable.

From a review of the ST, the Guidance and through testing the evaluator also determined that the guidance contains all of the administrative actions and their associated audit events that are relevant to the PP and to use of the TOE. These administrative actions are consistent with the security requirements implemented in the TOE and were found to have appropriate management capabilities identified in the guidance documentation.

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

The evaluator created a list of the required audit events. The evaluator then collected the audit event when running the other security functional tests described by the protection profiles. The evaluator then recorded these audit events in the proprietary Detailed Test Report (DTR). The security management events are handled in a similar manner. When the administrator was required to set a value for testing, the audit record associated with the administrator action was collected and recorded in the DTR.

2.1.5 User identity association (NDcPP22e:FAU_GEN.2)

2.1.5.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

See FAU_GEN.1

Component Testing Assurance Activities: This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

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

See NDcPP22e:FAU_GEN.1.1

2.1.6 Security Audit Generation (NDcPP22e:FAU_GEN_EXT.1)

2.1.6.1 NDcPP22e:FAU_GEN_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs, the requirements on TSS, Guidance Documentation and Tests regarding FAU_GEN_EXT.1 are already covered by the corresponding requirements for FAU_GEN.1.

For distributed TOEs, the requirements on TSS, Guidance Documentation and Tests regarding FAU_GEN_EXT.1 are already covered by the corresponding requirements for FAU_GEN.1.

Component Guidance Assurance Activities: For distributed TOEs, the requirements on TSS, Guidance Documentation and Tests regarding FAU_GEN_EXT.1 are already covered by the corresponding requirements for FAU_GEN.1.

See NDcPP22e:FAU_GEN.1.1

Component Testing Assurance Activities: For distributed TOEs, the requirements on TSS, Guidance Documentation and Tests regarding FAU_GEN_EXT.1 are already covered by the corresponding requirements for FAU_GEN.1.

See NDcPP22e:FAU_GEN.1.1

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

2.1.7.1 NDcPP22e:FAU_STG.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.2 NDcPP22e:FAU_STG.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it describes the amount of audit data that are stored locally and how these records are protected against unauthorized modification or deletion. The evaluator shall ensure that the TSS describes the conditions that must be met for authorized deletion of audit records.

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

Section 7.1.2 of the ST states the TOE is a distributed TOE which stores local audit data on the following components:

- VOS SD-WAN Controller and Branch devices (with the exception of traffic logs which are buffered in memory and forwarded in real-time to Analytics without being locally stored)
- Versa Director
- Versa Analytics

The Versa Director, VOS SD-WAN Controller and Branch devices are configured to send log data to the Analytics node over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from other TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real-time.

The TSF restricts access to audit logs stored on each component to authorized Security Administrators. Audit log settings can be configured to automatically archive and rotate log files, based on size limit. Audit log files may be deleted manually or automatically according to configured parameters. Audit log files can be deleted at the command of administrators with the Admin role, where administrators with super user privileges can manually delete the audit logs. Only authorized Security Administrators can read the audit records.

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

Section “Audit Log Protection” in the **Admin Guide** states that each TOE component is configured to send log data to the Analytics node over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from other TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real time.

The TSF restricts access to audit logs stored on each component to authorized Security Administrators. Audit logs are configured to automatically archive and rotate log files via cron job, based on size limit specified by logrotate.d parameters. Audit log files can be deleted at the command of administrators with the admin role, where administrators with super user privileges can manually delete the audit logs. Only authorized Security Administrators can read the audit records.

Section “Configure IPsec” in the **Admin Guide** describes how to configure an IPsec Tunnel profile on the TOE and use it to protect audit log records being transmitted to a remote auditing server.

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

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

shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

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

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

Test 1: Without prior authentication as security administrator, the TOE represents its' login screen to the user to provide correct credentials for login. The evaluator confirmed that no functions (modifying audit data behavior) can be executed prior to authentication.

Test 2: Continuing on from the previous test case, the evaluator provided the correct credentials for login. Once the evaluator logged in successfully to the TOE, only then, he was able to perform functions (modifying audit data behavior) and execute them. The evaluator confirmed that with prior authentication, the administrator is able to delete the audit records and verify that only the records authorized for deletion are deleted, and modify the behavior of the transmission of audit data.

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

2.1.8.1 NDcPP22e:FAU_STG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.8.2 NDcPP22e:FAU_STG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.8.3 NDcPP22e:FAU_STG_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

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

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

Section 7.1.2 of the ST states The TOE is a distributed TOE which stores local audit data on the following components:

- VOS SD-WAN Controller and Branch devices (with the exception of traffic logs which are buffered in memory and forwarded in real-time to Analytics without being locally stored)

- Versa Director
- Versa Analytics

The Versa Director, VOS SD-WAN Controller and Branch devices are configured to send log data to the Analytics node over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from other TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real-time.

Audit log settings can be configured to automatically archive and rotate log files via cron job, based on size limit specified by logrotate.d parameters (defined by number of bytes). By default, log files are rotated daily, or when the log file reaches 50MB. A maximum of 7 log files are retained, with the oldest log file deleted as rollover occurs. Audit log files may also be deleted manually or automatically according to configured parameters. Audit log files can be deleted at the command of administrators with the Admin role, where administrators with super user privileges can manually delete the audit logs. Only authorized Security Administrators can read the audit records.

The Security Administrator enables the log export functionality (LEF) on the VOS device. VOS devices export log data in IPFIX and syslog formats. To export log data from VOS devices to an Analytics node, the Security Administrator configures a log export template, a collector, and a LEF profile, and then selects the LEF profile when configuring a feature or service. The logs for the feature or service are forwarded to the active collector named in the LEF profile. The LEF profile can be applied in one of the following ways:

- Associate the LEF profile with a feature or service.
- Associate the LEF profile with a traffic-monitoring policy rule.
- Associate the LEF profile with the logging control configuration.
- Assign a LEF profile to be the default.

The method used depends on the type of logs being exported.

Logs are generated in syslog format and include a label, called the syslog identifier, identifying the log type. Each feature or service has one or more associated syslog identifiers. For logs sent to Analytics clusters and Netflow collectors, LEF adds an IPFIX overhead.

When a LEF profile is associated with a traffic-monitoring policy rule, the VOS device generates syslog messages for selected types of traffic monitoring flows.

When a LEF is associated with the logging control configuration, the VOS device generates syslog messages globally for all traffic-monitoring flows.

A LEF profile automatically sends logs of the following types to the active collector once they have been specified in the profile. The syslog identifier or identifiers corresponding to each log type are displayed in parentheses.

- Alarm logs (alarmLog)
- SD-WAN link and rule statistics monitoring (bwMonLog, infUtilLog)
- System logs (systemLoadLog)
- Firewall Logs (accessLog, sfwAccessLog, denyLog, monStatsLog)
- Flow logs (flowIdLog, flowMonHttpLog, flowMonLog)
- Packet Capture Logs (pcapLog)
- DDoS Logging (dosThreatLog)
- IP filtering logs (ipfLog)
- IP guard logs (ipguardLog)
- IDP logging (idpLog)
- Malformed packet log (malformedPktLog)
- Secure access logs (secAccGlobalStatsLog, secAccUserStatsLog)
- Traffic detection function logs (tdfPeakBwLog, tdfTcpPerfLog, tdfUsageReport)
- URL filtering logs (urlfLog)
- Antivirus log (avLog)

The Security Administrator must initially configure an Analytics node to collect logs. These nodes are called Analytics log collector nodes, and they collect log messages (simply called logs), which include alarms from all the VOS Branch and Controller devices in the network, in IPFIX format. Analytics log collector nodes run two programs, called the log collector exporter and the Versa Analytics driver, to accept and process the incoming logs. The log collector exporter program listens for incoming connections containing logs, called log export functionality (LEF) connections, and stores the logs on the Analytics node. The Analytics driver then processes these logs into the Analytics datastore.

The System Administrator configures the log collector exporter stream incoming logs to an external third-party collector by configuring a remote collector. A remote collector streams the logs to one or more third-party collectors in syslog format via IPsec in realtime.

An Analytics log collector node processes incoming logs in the following sequence:

1. The local collector on the Analytics log collector node receives logs sent from Versa Operating SystemTM (VOSTM) devices.

2. The local collector stores the logs in clear text files in its log storage directory, with one subdirectory for each organization. Each organization subdirectory contains a further subdirectory named for the routing instance that forwarded the log, and the incoming logs are collected into log files in these subdirectories.

3. Any log files created in the routing instance subdirectories under /var/tmp/log are automatically processed into the cluster datastores by the Versa Analytics driver.

4. After processing the log files, the Versa Analytics driver moves the log file into a backup directory under the /var/tmp/log directory.

5. A cron job stored in /etc/cron.d/log-archive periodically archives all log files stored in the backup directories under /var/tmp/log. Also, any additional log archive cron jobs that you have configured archive logs stored in non-default log storage directories. The log archive cron jobs convert the clear text files in the log storage directories to compressed gzip format and move them to a log archive directory. The archiving time interval can be hourly, weekly, or daily.

6. Authorized Security Administrators may delete archived logs manually using a CLI command which can be scripted and periodically run or executed on a schedule with a cron job.

Component Guidance Assurance Activities: The evaluator shall also examine the guidance documentation to ensure it describes how to establish the trusted channel to the audit server, as well as describe any requirements on the audit server (particular audit server protocol, version of the protocol required, etc.), as well as configuration of the TOE needed to communicate with the audit server.

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

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

Section “Configure Syslog Server” in the **Admin Guide** provides instruction for configuring the Director and Branches to transmit audit records securely to an external audit server. To securely transmit log messages to an audit server, IPsec is used between the Director and the syslog-ng server and between the Branches and the syslog-ng server. To securely send the logs to a trusted audit server, a static route must be defined to tunnel the traffic through the IPsec tunnel and a successful IPsec tunnel must be established between the TOE component forwarding the traffic and the receiving syslog-ng server.

The same section also indicates that audit messages can be configured to be transmitted directly to a remote syslog server, in which case each message will be simultaneously transmitted to the remote logging server as the message is written locally.

Section “Configure Log Exports” in the **Admin Guide** states that VOS data plane logs and other auditable events such as device alarms, are forwarded via IPFIX or syslog formats over an IPsec encrypted tunnel to Versa Analytics using Log Export Functionality (LEF). You can apply LEF profiles to inspection policies, firewall rules, and other configuration elements that forward logs specific to a security function.

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

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

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

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

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

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

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

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

Test 1: The successful establishment of the IPsec syslog connection for all TOE devices was demonstrated in FTP_ITC.1. In each case the TOE initiated the connection without administrator intervention. The use of IPsec ensured that no audits were viewed in cleartext. The audits collected as part of FAU_GEN.1 throughout testing were gathered from the remote syslog server running rsyslog version 8.16.0 thus demonstrating that audits were successfully received by the remote syslog server.

Test 2: The evaluator also continued to generate audit data until the local storage space was exceeded. The evaluator verified that when the local audit storage was filled to the maximum, the existing audit data was rotated and archived. The evaluator also verified that the TOE requires authorized Security Administrators to delete audit log archives.

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

Test 4: Tests 1 and 2 cover all auditing scenarios, which includes components that store audit data locally and components that send audit data to an external audit server.

2.1.9 Protected Local Audit Event Storage for Distributed TOEs (NDcPP22e:FAU_STG_EXT.4)

2.1.9.1 NDcPP22e:FAU_STG_EXT.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to confirm that it describes which TOE components store their security audit events locally and which send their security audit events to other TOE components for local storage. For the latter, the target TOE component(s) which store security audit events for other TOE components shall be identified. For every sending TOE component the corresponding receiving TOE component(s) need to be identified. For every transfer of audit information between TOE components it shall be described how the data is secured during transfer according to FTP_ITC.1 or FPT_ITT.1.

For each TOE component which does not store audit events locally by itself, the evaluator confirms that the TSS describes how the audit information is buffered before sending to another TOE component for local storage.

Section 7.1.2 of the ST states The TOE is a distributed TOE which stores local audit data on the following components:

- VOS SD-WAN Controller and Branch devices (with the exception of traffic logs which are buffered in memory and forwarded in real-time to Analytics without being locally stored)

- Versa Director
- Versa Analytics

The Versa Director, VOS SD-WAN Controller and Branch devices are configured to send log data to the Analytics node over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from other TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real-time.

The transfer of log data between TOE components aligns with the claims for FTP_ITC.1.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure that it describes how the link between different TOE components is established if audit data is exchanged between TOE components for local storage. The guidance documentation shall describe all possible configuration options for local storage of audit data and provide all instructions how to perform the related configuration of the TOE components.

The evaluator shall also ensure that the guidance documentation describes for every TOE component which does not store audit information locally how audit information is buffered before transmission to other TOE components.

Section “Audit Log Protection” in the **Admin Guide** states that each TOE component is configured to send log data to the Analytics node over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from other TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real time.

The TSF restricts access to audit logs stored on each component to authorized Security Administrators. Audit logs are configured to automatically archive and rotate log files via cron job, based on size limit specified by logrotate.d parameters. Audit log files can be deleted at the command of administrators with the admin role, where administrators with super user privileges can manually delete the audit logs. Only authorized Security Administrators can read the audit records.

Section “Configure IPsec” in the **Admin Guide** describes how to configure an IPsec Tunnel profile on the TOE and use it to protect audit log records being transmitted to a remote auditing server.

Component Testing Assurance Activities: For at least one of each type of distributed TOE components (sensors, central nodes, etc.), the following tests shall be performed using distributed TOEs.

Test 1: For each type of TOE component, the evaluator shall perform a representative subset of auditable actions and ensure that these actions cause the generation of appropriately formed audit records. Generation of such records can be observed directly on the distributed TOE component (if there is appropriate interface), or indirectly after transmission to a central location.

Test 2: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to the external audit server (as specified in FTP_ITC.1), the evaluator shall configure a trusted channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

Test 3: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to another TOE component (as specified in FTP_ITT.1 or FTP_ITC.1, respectively), the evaluator shall configure a secure channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

While performing these tests, the evaluator shall verify that the TOE behaviour observed during testing is consistent with the descriptions provided in the TSS and the Guidance Documentation. Depending on the TOE configuration, there might be a large number of different possible configurations. In such cases, it is acceptable to perform subset testing, accompanied by an equivalency argument describing the evaluator's sampling methodology.

Test 1: Each TOE component generates its own set of audit events. Throughout testing, a subset of audits was captured for each device and a table was created to show that all expected audits were captured. Refer to NDcPP22e:FAU_GEN.1 for more audit event details. The audits were collected for each component from the remote syslog server running rsyslog version 8.16.0. The IPsec tunnel transporting those audits was demonstrated in NDcPP22e:FTP_ITC.1.

Test 2: The successful establishing of the IPsec tunnel connection for all devices was demonstrated in FTP_ITC.1. In each case the TOE initiated the connection without administrator intervention. The use of IPsec ensured no audits were viewed in cleartext. The audits collected as part of FAU_GEN.1 throughout testing were gathered from the remote syslog server running rsyslog version 8.16.0 thus demonstrating that audits were successfully received by the remote syslog server.

Test 3: The Branches are configured to send log data to the Analytics node (part of the Director) over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from all TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real-time. That same IPsec channel (that is used to send log data internally between TOE components) has been tested throughout the course of the evaluation as part of IPsec testing. The successful establishing of the IPsec syslog connection for each TOE component is demonstrated in FTP_ITC.1. The use of IPsec ensured no audits were viewed in cleartext.

2.1.10 Protected Remote Audit Event Storage for Distributed TOEs (NDcPP22e:FAU_STG_EXT.5)

2.1.10.1 NDcPP22e:FAU_STG_EXT.5.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to confirm that it describes which TOE components store their security audit events locally and which send their security audit events to other TOE components for local storage. For the latter, the target TOE component(s) which store security audit events for other TOE components shall be identified. For every sending TOE component the corresponding receiving TOE component(s) need to be identified. For every transfer of audit information between TOE components it shall be described how the data is secured during transfer according to FTP_ITC.1 or FPT_ITT.1. For each TOE component which does not store audit events locally by itself, the evaluator confirms that the TSS describes how the audit information is buffered before sending to another TOE component for local storage.

Section 7.1.2 of the ST states The TOE is a distributed TOE which stores local audit data on the following components:

- VOS SD-WAN Controller and Branch devices (with the exception of traffic logs which are buffered in memory and forwarded in real-time to Analytics without being locally stored)
- Versa Director
- Versa Analytics

The Versa Director, VOS SD-WAN Controller and Branch devices are configured to send log data to the Analytics node over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from other TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real-time.

The transfer of log data between TOE components aligns with the claims for FTP_ITC.1.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure that it describes how the link between different TOE components is established if audit data is exchanged between TOE components for local storage. The guidance documentation shall describe all possible configuration options for local storage of audit data and provide all instructions how to perform the related configuration of the TOE components. The evaluator shall also ensure that the guidance documentation describes for every TOE component

which does not store audit information locally how audit information is buffered before transmission to other TOE components.

Section “Audit Log Protection” in the **Admin Guide** states that each TOE component is configured to send log data to the Analytics node over an internal IPsec tunnel in real-time. The Analytics node performs data analysis and provides reports and data visualization on syslogs received from other TOE components and is configured to export all TOE logs to an external syslog receiver via IPsec in real time.

The TSF restricts access to audit logs stored on each component to authorized Security Administrators. Audit logs are configured to automatically archive and rotate log files via cron job, based on size limit specified by logrotate.d parameters. Audit log files can be deleted at the command of administrators with the admin role, where administrators with super user privileges can manually delete the audit logs. Only authorized Security Administrators can read the audit records.

Section “Configure IPsec” in the **Admin Guide** describes how to configure an IPsec Tunnel profile on the TOE and use it to protect audit log records being transmitted to a remote auditing server.

Component Testing Assurance Activities: For at least one of each type of distributed TOE components (sensors, central nodes, etc.), the following tests shall be performed using distributed TOEs.

Test 1: For each type of TOE component, the evaluator shall perform a representative subset of auditable actions and ensure that these actions cause the generation of appropriately formed audit records. Generation of such records can be observed directly on the distributed TOE component (if there is appropriate interface), or indirectly after transmission to a central location.

Test 2: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to the external audit server (as specified in FTP_ITC.1), the evaluator shall configure a trusted channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

Test 3: For each type of TOE component that, in the evaluated configuration, is capable of transmitting audit information to another TOE component (as specified in FTP_ITT.1 or FTP_ITC.1, respectively), the evaluator shall configure a secure channel and confirm that audit records generated as a result of actions taken by the evaluator are securely transmitted. It is sufficient to observe negotiation and establishment of the secure channel with the TOE component and the subsequent transmission of encrypted data to confirm this functionality. Alternatively, the following steps shall be performed: The evaluator induces audit record transmission, then reviews the packet capture around the time of transmission and verifies that no audit data is transmitted in the clear.

While performing these tests, the evaluator shall verify that the TOE behaviour observed during testing is consistent with the descriptions provided in the TSS and the Guidance Documentation. Depending on the TOE

configuration, there might be a large number of different possible configurations. In such cases, it is acceptable to perform subset testing, accompanied by an equivalency argument describing the evaluator's sampling methodology.

Test 1: The results for NDcPP22e:FAU_GEN.1, demonstrate that appropriate audit records are recorded for each TOE component.

Test 2: The results for NDcPP22e:FAU_STG_EXT.1 test 1, demonstrate that each TOE component established a successful connection to the external audit log server in order to pass audit events in a secure manner.

Test 3: The packet captures in NDcPP22e:FPT_ITC.1, demonstrate the successful establishing of the IPsec syslog connection for each TOE component; the same secure channel (IPsec) is used by each TOE component to communicate internally and has been tested throughout the course of the evaluation as part of IPsec testing.

2.2 Communication (FCO)

2.2.1 Component Registration Channel Definition (NDcPP22e:FCO_CPC_EXT.1)

2.2.1.1 NDcPP22e:FCO_CPC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.1.2 NDcPP22e:FCO_CPC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.2.1.3 NDcPP22e:FCO_CPC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: If the TOE is not a distributed TOE then no evaluator action is necessary. For a distributed TOE the evaluator carries out the activities below. In carrying out these activities the evaluator shall determine answers to the following questions based on a combination of documentation analysis and testing (possibly also using input from carrying out the Evaluation Activities for the relevant registration channel, such as FTP_TRP.1(2)/Join), and shall report the answers.

Questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities:

a) What stops [The intent of the phrasing 'what stops...' as opposed to 'what secures...' is for the evaluator to pursue the answer to its lowest level of dependency, i.e. a level at which the security can clearly be seen to depend on things that are under appropriate control. For example, a channel may be protected by a public key that is provided to the relying party in a self-signed certificate. This enables cryptographic mechanisms to be applied to provide authentication (and therefore invites an answer that 'the check on the public key certificate secures...'), but does not ultimately stop an attacker from apparently authenticating because the attacker can produce their own self-signed certificate. The question 'what stops an unauthorised component from successfully communicating...' focuses attention on what an attacker needs to do, and therefore pushes the answer down to the level of whether a self-signed certificate could be produced by an attacker. Similarly a well-known key, or a key that is common to a type of device rather than an individual device, may be used in a confidentiality mechanism but does not provide confidentiality because an attacker can find the well-known key or obtain his own instance of a device containing the non-unique key.] a component from successfully communicating with TOE components (in a way that enables it to participate as part of the TOE) before it has properly authenticated and joined the TOE?

b) What is the enablement step? (Describe what interface it uses, with a reference to the relevant section and step in the operational guidance).

1) What stops anybody other than a Security Administrator from carrying out this step?

2) How does the Security Administrator know that they are enabling the intended component to join? (Identification of the joiner might be part of the enablement action itself or might be part of secure channel establishment, but it must prevent unintended joining of components)

c) What stops a component successfully joining if the Security Administrator has not carried out the enablement step; or, equivalently, how does the TOE ensure that an action by an authentic Security Administrator is required before a component can successfully join?

d) What stops a component from carrying out the registration process over a different, insecure channel?

e) If the FTP_TRP.1(2)/Join channel type is selected in FCO_CPC_EXT.1.2 then how do the registration process and its secure channel ensure that the data is protected from disclosure and provides detection of modification?

f) Where the registration channel does not rely on protection of the registration environment, does the registration channel provide a sufficient level of protection (especially with regard to confidentiality) for the data that passes over it?

g) Where the registration channel is subsequently used for normal internal communication between TOE components (i.e. after the joiner has completed registration), do any of the authentication or encryption features of the registration channel result in use of a channel that has weaker protection than the normal FPT_ITT.1 requirements for such a channel?

h) What is the disablement step? (Describe what interface it uses, with a reference to the relevant section and step in the operational guidance).

i) What stops a component successfully communicating with other TOE components if the Security Administrator has carried out the disablement step?

The evaluator shall examine the TSS to confirm that it:

a) Describes the method by which a Security Administrator enables and disables communications between pairs of TOE components.

b) Describes the relevant details according to the type of channel in the main selection made in FCO_CPC_EXT.1.2:

- First type: the TSS identifies the relevant SFR iteration that specifies the channel used

- Second type: the TSS (with support from the operational guidance if selected in FTP_TRP.1.3/Join) describes details of the channel and the mechanisms that it uses (and describes how the process ensures that the key is unique to the pair of components) - see also the Evaluation Activities for FTP_TRP.1(2)/Join.

The evaluator shall confirm that if any aspects of the registration channel are identified as not meeting FTP_ITC.1 or FPT_ITT.1, then the ST has also selected the FTP_TRP.1(2)/Join option in the main selection in FCO_CPC_EXT.1.2.

Section 7.1.12 of the ST states VOS devices are added to the distributed TOE using Zero-Touch Provisioning.

Two enablement methods are supported in the evaluated configuration:

- URL-based ZTP—URL-based ZTP allows an onsite administrator to activate a VOS device. The administrator connects a laptop to the VOS device and clicks an email link to a staging Controller node, which completes the activation process.

- From the CLI—A site administrator can connect to the CLI on a VOS device and run a staging script that activates the device.

Both methods require a Security Administrator with direct control of the device and requires positive enablement on each component prior to joining the distributed TOE. No registration channel is used. Once activated, TOE components will communicate via the IPsec trusted channel defined in FPT_ITC.1. Without performing the enablement steps, the TOE components will not have the parameters necessary to establish a connection. Only Security Administrators may disable and remove TOE components via the Director.

During startup of the Branch appliance, an IPsec tunnel is established with the SD-WAN Controller. All communication between Versa VOS (Branch appliance) and the Versa Director/Analytics is sent via the IPsec tunnel.

The TOE supports establishing trusted channels between VOS and Controller, using the protocols/formats described in the table below.

Source	Destination	Protocol / Format
VOS Branch	VOS Controller	IPsec
VOS Branch	VOS Controller	BGP Routing information of current Branch within IPsec.
VOS Controller	VOS Branch	BGP Routing information of other Branches within IPsec.
VOS Branch	VOS Controller	Security Association Information of current Branch within IPsec.
VOS Controller	VOS Branch	Security Association information of other Branches within IPsec.

The TOE supports establishing trusted channels between VOS and Director/Analytics, using the protocols/formats described in the table below.

Source	Destination	Protocol / Format
VOS Branch	Analytics	IPFIX logs within IPsec.

Source	Destination	Protocol / Format
Director	VOS Branch	Configuration information in NETCONF format within SSH session within IPsec.
Director	VOS Branch	Rest API calls using HTTPS session within IPsec.

The TOE supports establishing trusted channels between Controller and Director/Analytics, using the protocols/formats described in the table below.

Source	Destination	Protocol / Format
Director	VOS Controller	Configuration information in NETCONF format within SSH session using Management Interface of Controller (to configure the Controller itself) within IPsec
Director	VOS Controller	Rest API calls using HTTPS session using Management Interface of Controller (to monitor the Controller itself) within IPsec
Director	VOS Controller	Configuration information in NETCONF format within SSH session using Control Interface of Controller which is routed over IPSec to Branch (to configure the Branch)
Director	VOS Controller	Rest API calls using HTTPS session using Control Interface of Controller which is routed over IPSec to Branch (to monitor the Branch)
VOS Controller	Analytics	IPFIX Logs during the operation of Controllers from Control Interface of Controller to Analytics via IPsec

Source	Destination	Protocol / Format
VOS Controller	Analytics	IPFIX Logs from Branch Appliances (received via IPSec) routed using Control Interface of Controller to Analytics via IPsec

Component Guidance Assurance Activities: (Note: paragraph 274 lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

The evaluator shall examine the guidance documentation to confirm that it contains instructions for enabling and disabling communications with any individual component of a distributed TOE. The evaluator shall confirm that the method of disabling is such that all other components can be prevented from communicating with the component that is being removed from the TOE (preventing the remaining components from either attempting to initiate communications to the disabled component, or from responding to communications from the disabled component).

The evaluator shall examine the guidance documentation to confirm that it includes recovery instructions should a connection be unintentionally broken during the registration process.

If the TOE uses a registration channel for registering components to the TOE (i.e. where the ST author uses the FTP_ITC.1/FPT_ITT.1 or FTP_TRP.1(2)/Join channel types in the main selection for FCO_CPC_EXT.1.2) then the evaluator shall examine the Preparative Procedures to confirm that they:

a) describe the security characteristics of the registration channel (e.g. the protocol, keys and authentication data on which it is based) and shall highlight any aspects which do not meet the requirements for a steady-state inter-component channel (as in FTP_ITC.1 or FPT_ITT.1)

b) identify any dependencies between the configuration of the registration channel and the security of the subsequent inter-component communications (e.g. where AES-256 inter-component communications depend on transmitting 256 bit keys between components and therefore rely on the registration channel being configured to use an equivalent key length)

c) identify any aspects of the channel can be modified by the operational environment in order to improve the channel security, and shall describe how this modification can be achieved (e.g. generating a new key pair, or replacing a default public key certificate).

As background for the examination of the registration channel description, it is noted that the requirements above are intended to ensure that administrators can make an accurate judgement of any risks that arise from the default registration process. Examples would be the use of self-signed certificates (i.e. certificates that are not chained to an external or local Certification Authority), manufacturer-issued certificates (where control over

aspects such as revocation, or which devices are issued with recognised certificates, is outside the control of the operational environment), use of generic/non-unique keys (e.g. where the same key is present on more than one instance of a device), or well-known keys (i.e. where the confidentiality of the keys is not intended to be strongly protected - note that this need not mean there is a positive action or intention to publicise the keys).

In the case of a distributed TOE for which the ST author uses the FTP_TRP.1(2)/Join channel type in the main selection for FCO_CPC_EXT.1.2 and the TOE relies on the operational environment to provide security for some aspects of the registration channel security then there are additional requirements on the Preparative Procedures as described in section 3.5.1.2.

Section “Deploy a VOS Branch Device” of the **Admin Guide** provides instructions on how to register a Branch with the Director, as specified in the ST. Network settings must be correctly configured on the Branch device and the management network should be an internal, trusted network separated physically or logically from the monitored network.

In order for the Director and Branch to communicate, they must successfully complete a registration process, which requires administrative actions on the Director and corresponding administrative actions on the Branch. The administrative actions on Director and Branch require the administrator to create and deploy a work flow template that the two devices will use to authenticate their initial IPsec communications. A work flow template includes several configuration parameters like IP addresses and interfaces that will be utilized by the Branch. During the registration process, the Director and Branch confirm they have a matching work flow template, and then the post staging process initiates. This process basically deploys the work flow configuration created on the Director onto the Branch.

If device registration fails due to mismatched work flow template, or incorrect IP address or hostname, the information on the Director and /or Branch should be corrected and registration should be reinitiated from the Director. If the connection between the Director and Branch is broken during device registration, the Director and Branch will not attempt to reconnect and retry registration but rather the registration should be reinitiated from the Director.

The communication between the Director and Branch is protected by IPsec. IPsec provides authentication, key exchange, encryption and integrity protection of all data transmitted between the TOE components. If connectivity is lost between Director and Branch after device registration each endpoint will automatically attempt to re-initiate connection to the other until connectivity is restored, no administrative action is required other than resolving any connectivity issues in the networks between the Director and Branch. The current status of each device can be viewed on the Devices page through the Director’s Web UI where an icon indicates the current status of each deployed Branch.

Section “Delete a Device” in the **Admin Guide** provides the specific instructions for device de-registration from the Director. It states that when a device is deleted, communications between the Director and the deleted device are disabled and you cannot push policies to that device. Essentially, you cannot manage that device using Director, unless you redeploy if from an existing configuration backup.

Component Testing Assurance Activities: (Note: paragraph 274 lists questions for which the evaluator needs to determine and report answers through the combination of the TSS, Guidance Documentation, and Tests Evaluation Activities.)

The evaluator shall carry out the following tests:

a) Test 1.1: the evaluator shall confirm that an IT entity that is not currently a member of the distributed TOE cannot communicate with any component of the TOE until the non-member entity is enabled by a Security Administrator for each of the non-equivalent TOE components [An 'equivalent TOE component' is a type of distributed TOE component that exhibits the same security characteristics, behaviour and role in the TSF as some other TOE component. In principle a distributed TOE could operate with only one instance of each equivalent TOE component, although the minimum configuration of the distributed TOE may include more than one instance (see discussion of the minimum configuration of a distributed TOE, in section B.4). In practice a deployment of the TOE may include more than one instance of some equivalent TOE components for practical reasons, such as performance or the need to have separate instances for separate subnets or VLANs.] that it is required to communicate with (non-equivalent TOE components are as defined in the minimum configuration for the distributed TOE)

b) Test 1.2: the evaluator shall confirm that after enablement, an IT entity can communicate only with the components that it has been enabled for. This includes testing that the enabled communication is successful for the enabled component pair, and that communication remains unsuccessful with any other component for which communication has not been explicitly enabled.

Some TOEs may set up the registration channel before the enablement step is carried out, but in such a case the channel must not allow communications until after the enablement step has been completed.

The evaluator shall repeat Tests 1.1 and 1.2 for each different type of enablement process that can be used in the TOE.

c) Test 2: The evaluator shall separately disable each TOE component in turn and ensure that the other TOE components cannot then communicate with the disabled component, whether by attempting to initiate communications with the disabled component or by responding to communication attempts from the disabled component.

d) Test 3: The evaluator shall carry out the following tests according to those that apply to the values of the main (outer) selection made in the ST for FCO_CPC_EXT.1.2.

1) If the ST uses the first type of communication channel in the selection in FCO_CPC_EXT.1.2 then the evaluator tests the channel via the Evaluation Activities for FTP_ITC.1 or FPT_ITT.1 according to the second selection - the evaluator shall ensure that the test coverage for these SFRs includes their use in the registration process.

2) If the ST uses the second type of communication channel in the selection in FCO_CPC_EXT.1.2 then the evaluator tests the channel via the Evaluation Activities for FTP_TRP.1(2)/Join.

3) If the ST uses the 'no channel' selection then no test is required.

e) Test 4: The evaluator shall perform one of the following tests, according to the TOE characteristics identified in its TSS and operational guidance:

1) If the registration channel is not subsequently used for inter-component communication, and in all cases where the second selection in FCO_CPC_EXT.1.2 is made (i.e. using FTP_TRP.1(2)/Join) then the evaluator shall confirm that the registration channel can no longer be used after the registration process has completed, by attempting to use the channel to communicate with each of the endpoints after registration has completed.

2) If the registration channel is subsequently used for intercomponent communication then the evaluator shall confirm that any aspects identified in the operational guidance as necessary to meet the requirements for a steady-state intercomponent channel (as in FTP_ITC.1 or FPT_ITT.1) can indeed be carried out (e.g. there might be a requirement to replace the default key pair and/or public key certificate).

f) Test 5: For each aspect of the security of the registration channel that operational guidance states can be modified by the operational environment in order to improve the channel security (cf. AGD_PRE.1 refinement item 2 in (cf. the requirements on Preparative Procedures in 3.5.1.2), the evaluator shall confirm, by following the procedure described in the operational guidance, that this modification can be successfully carried out.

Test 1: (1.1 and 1.2) The evaluator registered the VOS branches with the Director for TOE distributed communication and verified that the communication was successful. For the next test (1.2), the evaluator verified that after the registration, communications were allowed to flow only between the two paired devices. The Director was not able to communicate with or push policies to the any of the unpaired VOS branches.

Test 2: The evaluator verified that the VOS branch was registered and that distributed communication was enabled with the Director, the evaluator then unregistered the VOS branch and verified that the device was no longer registered with the Director. The evaluator then verified that the TOE components could not communicate with the device after it was no longer registered.

Test 3 - The ST uses the 'no channel' selection and therefore, no test is required.

Test 4 - There are no necessary actions to meet the requirements for a steady-state inter-component channel.

Test 5 - No actions exist to improve channel security.

2.3 Cryptographic support (FCS)

2.3.1 Cryptographic Key Generation (NDcPP22e:FCS_CKM.1)

2.3.1.1 NDcPP22e:FCS_CKM.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 7.1.3 of the ST states The TSF provides the following key generation capabilities:

- Using RSA 3072 bits or greater as described in FIPS PUB 186-4 B.3.3
- Using ECDSA P-256, P-384, or P-521 as described in FIPS PUB 186-4 B.4.1 and B.4.2
- Using FFC Schemes using “safe-prime” groups per NIST SP 800-56Arev3 (available for TLS v1.2 connections only).

RSA and ECDSA keypairs are used in the following functions:

- X.509 certificates used in TLS and IPsec (FCS_TLSS_EXT.1, FCS_TLSS_EXT.2, FCS_IPSEC_EXT.1)
- Verification of TSF binary integrity (FPT_TUD_EXT.3)
- SSH host key identification and public key authentication (FCS_SSHS_EXT.1)

The TSF supports the ECDH key establishment with the ephemeralUnified scheme and P-256, P-384, P-521 curves in accordance with SP 800-56Arev3 for the following functions.:

- Administrative sessions to the SSH CLi (FCS_SSHS_EXT.1)
- Administrative sessions to the HTTPS web UI (FCS_HTTPS_EXT.1, FCS_TLSS_EXT.1)
- IKE/IPsec sessions (FCS_IPSEC_EXT.1)

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all cryptographic protocols defined in the Security Target.

Section “Perform Manual Hardening for SSH” in the **Admin Guide** describes how to enable FIPS mode which in return limits only the allowed ciphers as defined in the Common Criteria Security Target. It states that no additional cipher configuration is required.

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** describe how to limit the TLS ciphers to only those that are allowed in the evaluated configuration.

Sections “Generate a Private Key” and “Generate a CSR” in the **Admin Guide** describe generating private keys with the required key sizes.

Component Testing Assurance Activities: Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Generation of long-term cryptographic keys (i.e. keys that are not ephemeral keys/session keys) might be performed automatically (e.g. during initial start-up). Testing of key generation must cover not only administrator invoked key generation but also automated key generation (if supported).

Key Generation for FIPS PUB 186-4 RSA Schemes

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

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

a) Random Primes:

- Provable primes

- Probable primes

b) Primes with Conditions:

- Primes p1, p2, q1, q2, p and q shall all be provable primes

- Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes

- Primes p1, p2, q1, q2, p and q shall all be probable primes

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

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

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

- Primes q and p shall both be provable primes

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

and two ways to generate the cryptographic group generator g:

- Generator g constructed through a verifiable process

- Generator g constructed through an unverifiable process.

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

- len(q) bit output of RBG where 1 <=x <= q-1

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

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

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

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

- g != 0,1

- q divides p-1

- g^q mod p = 1

- g^x mod p = y

for each FFC parameter set and key pair.

FFC Schemes using 'safe-prime' groups

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

(TD0580 applied)

The TOE has been CAVP tested. Refer to the CAVP certificates identified in Section 1.1.2. See CKM.2.1 for safe primes testing.

2.3.2 Cryptographic Key Generation (for IKE Peer Authentication) (VPNGW13:FCS_CKM.1/IKE)

2.3.2.1 VPNGW13:FCS_CKM.1.1/IKE

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

Section 7.1.3 of the ST states The TSF provides the following key generation capabilities:

- Using RSA 3072 bits or greater as described in FIPS PUB 186-4 B.3.3
- Using ECDSA P-256, P-384, or P-521 as described in FIPS PUB 186-4 B.4.1 and B.4.2

There are no exceptions to any of the “shall” or “shall not” type statements made in FIPS PUB 186-4.

There are no TOE specific extensions not described in FIPS PUB 186-4.

Section 7.3 describes the process of how the IPsec related keys are generated.

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

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration.

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Sections “Configure X.509 Certificate”, “Generate a Private Key”, “Generate a CSR”, and “Import CA Chain” in the **Admin Guide** provide steps for enrolling a certificate and specifying the certificate contents including the Key Type: RSA or ECDSA and the Key Size. This key generation process on the Director results in private keys stored in binary format on each TOE component that will use the certificate associated with the generated keys.

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

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

For all other selections:

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

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

2.3.3 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.3.3.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme. It is sufficient to provide the scheme, SFR, and service in the TSS.

The intent of this activity is to be able to identify the scheme being used by each service. This would mean, for example, one way to document scheme usage could be:

Scheme | SFR | Service

RSA | FCS_TLSS_EXT.1 | Administration

ECDH | FCS_SSHC_EXT.1 | Audit Server

ECDH | FCS_IPSEC_EXT.1 | Authentication Server

The information provided in the example above does not necessarily have to be included as a table but can be presented in other ways as long as the necessary data is available.

(TD0580 applied)

Section 7.1.3 of the ST states The TSF provides the following key generation capabilities:

- Using RSA 3072 bits or greater as described in FIPS PUB 186-4 B.3.3
- Using ECDSA P-256, P-384, or P-521 as described in FIPS PUB 186-4 B.4.1 and B.4.2

The TSF supports the ECDH key establishment with the ephemeralUnified scheme and P-256, P-384, P-521 curves in accordance with SP 800-56Arev3 for the following functions.:

- Administrative sessions to the SSH CLi (FCS_SSHS_EXT.1)
- Administrative sessions to the HTTPS web UI (FCS_HTTPS_EXT.1, FCS_TLSS_EXT.1)
- IKE/IPsec sessions (FCS_IPSEC_EXT.1)

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

See FCS_CKM.1

Component Testing Assurance Activities: Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

RSA-based key establishment

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

FFC Schemes using 'safe-prime' groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

(TD0580 applied)

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

Safe primes have been tested as part of TLSS using a known good implementation.

2.3.4 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.3.4.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator examines the TSS to ensure it lists all relevant keys (describing the origin and storage location of each), all relevant key destruction situations (e.g. factory reset or device wipe function, disconnection of trusted channels, key change as part of a secure channel protocol), and the destruction method used in each case. For the purpose of this Evaluation Activity the relevant keys are those keys that are relied upon to support any of the SFRs in the Security Target. The evaluator confirms that the description of keys and storage locations is consistent with the functions carried out by the TOE (e.g. that all keys for the TOE-specific secure channels and protocols, or that support FPT_APW.EXT.1 and FPT_SKP_EXT.1, are accounted for2). In particular, if a TOE claims not to store plaintext keys in non-volatile memory then the evaluator checks that this is consistent with the operation of the TOE.

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

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

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

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

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

Section 7.1.3 of the ST The TOE meets all requirements specified in FIPS 140-2 for destruction of keys. All the keys stored within the TOE can be zeroized.
The list of all relevant keys (including the origin and storage of each), and all key destruction scenarios are described in section 7.3 of the ST.

No configurations or circumstances exist that do not conform to the key destruction requirement.

Section 7.3 of the ST describes the key zeroization provided by the TOE for both volatile memory (RAM) and non-volatile memory (flash). All CSPs, with the exception of the operator password are stored in plaintext and cleared using zeroization. The operator password is stored as a SHA-512 hash. The operator password is overwritten with new hash value. For all plaintext CSPs in RAM, the CSPs are automatically overwritten with zeroes at the end protocol sessions. For all CSPs in flash, the CSPs are zeroized via commands.

Component Guidance Assurance Activities: A TOE may be subject to situations that could prevent or delay key destruction in some cases. The evaluator shall check that the guidance documentation identifies configurations or circumstances that may not strictly conform to the key destruction requirement, and that this description is consistent with the relevant parts of the TSS (and any other supporting information used). The evaluator shall check that the guidance documentation provides guidance on situations where key destruction may be delayed at the physical layer.

For example, when the TOE does not have full access to the physical memory, it is possible that the storage may be implementing wear-levelling and garbage collection. This may result in additional copies of the key that are logically inaccessible but persist physically. Where available, the TOE might then describe use of the TRIM command [Where TRIM is used then the TSS and/or guidance documentation is also expected to describe how the keys are stored such that they are not inaccessible to TRIM, (e.g. they would need not to be contained in a file less than 982 bytes which would be completely contained in the master file table)] and garbage collection to destroy these persistent copies upon their deletion (this would be explained in TSS and Operational Guidance).

The TOE does not have conditions that involve delayed key destruction. The TOE meets all requirements specified in FIPS 140-2 for destruction of keys. All the keys stored within the TOE can be zeroized. No configurations or circumstances exist that do not conform to the key destruction requirement.

Component Testing Assurance Activities: None Defined

2.3.5 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e\VPNGW13:FCS_COP.1/DataEncryption)

2.3.5.1 NDcPP22e\VPNGW13:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 7.1.4 of the ST states the TSF provides data encryption and decryption capabilities in support of IKE/IPsec, TLS, and SSH using 128 and 256 bits AES in CBC, CTR, and GCM modes as described in FIPS PUB 197 and NIST SP 800-38D.

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected mode(s) and key size(s) defined in the Security Target supported by the TOE for data encryption/decryption.

Section "Perform Manual Hardening for SSH" in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration.

Section "Generate Director X.509 Certificate and Configure TLS Ciphers" in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section "Configure IPsec" in the **Admin Guide** provides instructions for creating an IKEv2 IPsec VPN profile. This includes choosing the ESP Hash method, the hash or integrity algorithm to use in the Proposal for authentication, and the ESP Encryption method, the Encapsulating Security Protocol (ESP) encryption algorithm for this Proposal.

Additionally, it includes instructions for choosing the integrity algorithm (hash) used in the IKEv2 policy and the encryption algorithm used to establish the Phase 1 SA for protecting the Phase 2 negotiations.

Component Testing Assurance Activities: AES-CBC Known Answer Tests

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

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

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

Input: PT, IV, Key

for i = 1 to 1000:

if i == 1:

CT[1] = AES-CBC-Encrypt(Key, IV, PT)

PT = IV

else:

CT[i] = AES-CBC-Encrypt(Key, PT)

PT = CT[i-1]

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

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

AES-GCM Test

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

128 bit and 256 bit keys

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

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

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

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

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

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

AES-CTR Known Answer Tests

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

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

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

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

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

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

AES-CTR Multi-Block Message Test

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

AES-CTR Monte-Carlo Test

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

Input: PT, Key

for i = 1 to 1000:

CT[i] = AES-ECB-Encrypt(Key, PT) PT = CT[i]

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

There is no need to test the decryption engine.

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

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

2.3.6.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check that the association of the hash function with other TSF cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Section 7.1.4 states the TSF provides hashing capabilities in support of HMAC operations and digital signature functions:

- Using SHA-1, SHA-256, SHA-384, and SHA-512 as described in FIPS PUB 180-4.

Component Guidance Assurance Activities: The evaluator checks the AGD documents to determine that any configuration that is required to configure the required hash sizes is present.

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration.

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section “Configure IPsec” in the **Admin Guide** provides instructions for creating an IKEv2 IPsec VPN profile. This includes choosing the ESP Hash method, the hash or integrity algorithm to use in the Proposal for authentication, and the ESP Encryption method, the Encapsulating Security Protocol (ESP) encryption algorithm for this Proposal.

Additionally, it includes instructions for choosing the integrity algorithm (hash) used in the IKEv2 policy and the encryption algorithm used to establish the Phase 1 SA for protecting the Phase 2 negotiations.

Component Testing Assurance Activities: The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF only hashes messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs.

The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

Short Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Short Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Bit-oriented Mode

The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 99*i, where 1 <= i <= m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Selected Long Messages Test - Byte-oriented Mode

The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm (e.g. 512 bits for SHA-256). The length of the ith message is m + 8*99*i, where 1 <= i <= m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

Pseudorandomly Generated Messages Test

This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

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

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

2.3.7.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it specifies the following values used by the HMAC function: key length, hash function used, block size, and output MAC length used.

Section 7.1.4 of the ST states The TSF provides HMAC capabilities:

- Using 160, 256, 384, and 512 bits HMAC-SHA, described in FIPS PUB 198-1.
- HMAC-SHA-1 supports a key length of at least 160 bits with a MAC length of 160 and block size of 512 bits.
- HMAC-SHA-256 supports a key length of at least 256 bits with a MAC length of 256 bits and a block size of 512 bits.

- HMAC-SHA-384 supports a key length of at least 384 bits with a MAC length of 384 bits and a block size of 1024 bits.

- HMAC-SHA-512 supports a key length of at least 512 bits with a MAC length of 512 bits and a block size of 1024 bits

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the values used by the HMAC function: key length, hash function used, block size, and output MAC length used defined in the Security Target supported by the TOE for keyed hash function.

Section "Perform Manual Hardening for SSH" in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration.

Section "Generate Director X.509 Certificate and Configure TLS Ciphers" in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section "Configure IPsec" in the **Admin Guide** provides instructions for creating an IKEv2 IPsec VPN profile. This includes choosing the ESP Hash method, the hash or integrity algorithm to use in the Proposal for authentication, and the ESP Encryption method, the Encapsulating Security Protocol (ESP) encryption algorithm for this Proposal.

Additionally, it includes instructions for choosing the integrity algorithm (hash) used in the IKEv2 policy and the encryption algorithm used to establish the Phase 1 SA for protecting the Phase 2 negotiations.

Component Testing Assurance Activities: For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and message data using a known good implementation.

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

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

2.3.8.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it specifies the cryptographic algorithm and key size supported by the TOE for signature services.

Section 7.1.4 of the ST states The TSF provides digital signature capabilities:

- Using RSA PKCS-PSS or PKCS#1 with 2048 or 3072 bits as described in FIPS PUB 186-4.
- Using ECDSA with 256, 384, and 521 bits as described in FIPS PUB 186-4.

RSA and ECDSA signatures are used in the following functions:

- X.509 certificates used in TLS and IPsec (FCS_TLSS_EXT.1, FCS_TLSS_EXT.2, FCS_IPSEC_EXT.1)
- Verification of TSF binary integrity (FPT_TUD_EXT.3)

SSH host key identification and public key authentication (FCS_SSHS_EXT.1)

Component Guidance Assurance Activities: The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected cryptographic algorithm and key size defined in the Security Target supported by the TOE for signature services.

Sections “Configure X.509 Certificate”, “Generate a Private Key”, “Generate a CSR”, and “Import CA Chain” in the **Admin Guide** provide steps for enrolling a certificate and specifying the certificate contents including the Key Type: RSA or ECDSA and the Key Size. This key generation process on the Director results in private keys stored in binary format on each TOE component that will use the certificate associated with the generated keys.

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration.

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section “Configure IPsec” in the **Admin Guide** provides instructions for creating an IKEv2 IPsec VPN profile. This includes choosing the ESP Hash method, the hash or integrity algorithm to use in the Proposal for authentication, and the ESP Encryption method, the Encapsulating Security Protocol (ESP) encryption algorithm for this Proposal.

Additionally, it includes instructions for choosing the integrity algorithm (hash) used in the IKEv2 policy and the encryption algorithm used to establish the Phase 1 SA for protecting the Phase 2 negotiations.

Component Testing Assurance Activities: ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator generates or obtains 10 messages for each modulus size/SHA combination supported by the TOE. The TOE generates and returns the corresponding signatures.

The evaluator shall verify the correctness of the TOE's signature using a trusted reference implementation of the signature verification algorithm and the associated public keys to verify the signatures.

Signature Verification Test

For each modulus size/hash algorithm selected, the evaluator generates a modulus and three associated key pairs, (d, e). Each private key d is used to sign six pseudorandom messages each of 1024 bits using a trusted reference implementation of the signature generation algorithm. Some of the public keys, e, messages, or signatures are altered so that signature verification should fail. For both the set of original messages and the set of altered messages: the modulus, hash algorithm, public key e values, messages, and signatures are forwarded to the TOE, which then attempts to verify the signatures and returns the verification results.

The evaluator verifies that the TOE confirms correct signatures on the original messages and detects the errors introduced in the altered messages.

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

2.3.9 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.3.9.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.3 NDcPP22e:FCS_HTTPS_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 7.1.5 of the ST states The TSF implements a HTTPS server in compliance with RFC 2818 and implements X.509 certificates for server self-identification. The TSF implements TLSv1.2 as a server without mutual authentication for securing management connections to the Director UI. TLS v1.1, 1.0, SSL 3.0, SSL 2.0, and any other unsupported TLS versions will be rejected.

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

Section “Log In Remotely to Director” in the **Admin Guide** states that the TOE offers a Web UI that is protected with HTTPS. This section also explains the procedure to access the TOE's Web UI using a web browser.

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

Tests are performed in conjunction with the TLS evaluation activities.

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

The TOE’s HTTPS channel does not support mutual authentication and therefore FIA_X509_EXT.1 is not applicable to HTTPS in this evaluation. The testing of the HTTPS channel was demonstrated in FCS_TLSS_EXT.1.

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

2.3.10.1 NDcPP22e:FCS_IPSEC_EXT.1.1

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

RFC 4301.

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

Section 7.1.5 of the ST states the TSF supports the construction of a SPD consisting of BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) rules through a combination of IPsec SAs, IP forwarding rulesets and traffic filters. The packet processing algorithm is described as follows:

1. A packet is received on an interface and is placed into an outbound queue for processing.

2. If the packet does not violate any inspection policies or default firewall rules, the packet is matched against a set of rules in a top-down order until a match is found.

3. If the packet matches a rule which is marked as drop, the packet will be immediately discarded.

4. If the packet matches a rule which is marked as permit, the packet will be forwarded and transmitted from the destination interface. If the packet is not flagged by the IPsec VPN policy, or is not part of an existing SA, the packet will flow in plaintext.

5. If the packet matches an IPsec VPN policy, the packet will be forwarded encrypted according to the SA, if the packet matches an existing SA. If the packet does not match an existing SA, a new one will be established and upon completion of the SA, the packet will be forwarded encrypted.

6. If the packet does not match any of the configured rules, it will be dropped by a default deny rule.

Guidance Assurance Activities: The evaluator shall examine the guidance documentation to verify it instructs the Administrator how to construct entries into the SPD that specify a rule for processing a packet. The description includes all three cases â€“ a rule that ensures packets are encrypted/decrypted, dropped, and flow through the TOE without being encrypted. The evaluator shall determine that the description in the guidance documentation is

consistent with the description in the TSS, and that the level of detail in the guidance documentation is sufficient to allow the administrator to set up the SPD in an unambiguous fashion. This includes a discussion of how ordering of rules impacts the processing of an IP packet.

Section “Configure IPsec” in the **Admin Guide** states that the TSF supports the construction of an SPD consisting of BYPASS (e.g., no encryption), DISCARD (e.g., drop the packet), and PROTECT (e.g., encrypt the packet) rules through a combination of IPsec SAs, IP forwarding rulesets and traffic filters. The packet processing algorithm is described as follows:

1. A packet is received on an interface and is placed into an outbound queue for processing.

2. If the packet does not violate any inspection policies or default firewall rules, the packet is matched against a set of rules in a top-down order until a match is found.

3. If the packet matches a rule which is marked as drop, the packet will be immediately discarded.

4. If the packet matches a rule which is marked as permit, the packet will be forwarded and transmitted from the destination interface. If the packet is not flagged by the IPsec VPN policy, or is not part of an existing SA, the packet will flow in plaintext.

5. If the packet matches an IPsec VPN policy, the packet will be forwarded encrypted according to the SA, if the packet matches an existing SA. If the packet does not match an existing SA, a new one will be established and upon completion of the SA, the packet will be forwarded encrypted.

6. If the packet does not match any of the configured rules, it will be dropped by a default deny rule.

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

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

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

via the audit trail and packet captures, for each scenario that the expected behavior is exhibited, and is consistent with both the TSS and the guidance documentation.

Test 1: The TOE uses access-lists in order to control the flow of traffic. If traffic is destined for an IP address associated with the VPN tunnel, then the traffic will be protected. Otherwise, the access-lists are screened in order to determine whether the traffic is allowed through unprotected or dropped. The evaluator sent ICMP packets to the VPN interface of the TOE and confirmed via the packet capture that the traffic was encrypted with IPsec. The evaluator then sent ICMP packets from an IP address on the management subnet as this was not part of IPsec traffic selectors and confirmed that this resulted in the traffic being sent in plaintext (matching the bypass rule). Lastly, the evaluator attempted to SSH to the TOE. The packet capture shows that the TOE did not respond to the request and the SSH attempt was dropped (matching the discard rule). The evaluator also constructed a default deny rule entry in the access-list that would drop any traffic that doesn't match any of the previous rules for BYPASS and DISCARD. The evaluator then sent ICMP packets from an IP address on the subnet not defined in the SPD rules. The traffic was dropped as part of a default deny.

Test 2: The TOE uses access-lists in order to control the flow of traffic for SPDs. The results that demonstrate the TOE's SPDs are demonstrated in its firewall testing (FFW_RUL_EXT.1.8 and FFW_RUL_EXT.1.9).

2.3.10.2 NDcPP22e:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

This test was performed as part of NDcPP22e:FCS_IPSEC_EXT.1.1-t1, which demonstrates the TOE protecting a packet, discarding a packet, allowing the packet to bypass the tunnel, and dropping the packet if a packet field is modified and does not match any rule.

2.3.10.3 NDcPP22e:FCS_IPSEC_EXT.1.3

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

Section 7.1.5 of the ST states the TOE supports tunnel mode.

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

Section "Configure IPsec" in the **Admin Guide** as well as the ST both state that the TOE only supports Tunnel mode for IPsec. Transport mode is not supported.

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

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

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

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

Test 2: Not applicable, transport mode is not selected.

2.3.10.4 NDcPP22e:FCS_IPSEC_EXT.1.4

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

Section 7.1.5 of the ST states Each of the IPsec implementations within the TSF support the following encryption ciphers for IKEv2 SA and Child SAs. A configuration parser ensures that the strength of the IKEv2 SA cipher cannot be less than the IKEv2 Child SA cipher:

- AES-CBC-128,
- AES-CBC-256,
- AES-GCM-256

The TSF supports the following data integrity algorithms for IKEv2 SA and Child SAs:

- HMAC-SHA-1,
- HMAC-SHA-256,
- HMAC-SHA-384,
- HMAC-SHA-512

This aligns with the claims for FCS_COP.1(4)/KeyedHash.

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

Section "Configure IPsec" in the **Admin Guide** provides instructions for creating an IKEv2 IPsec VPN profile. This includes choosing the ESP Hash method, the hash or integrity algorithm to use in the Proposal for authentication, and the ESP Encryption method, the Encapsulating Security Protocol (ESP) encryption algorithm for this Proposal.

Additionally, it includes instructions for choosing the integrity algorithm (hash) used in the IKEv2 policy and the encryption algorithm used to establish the Phase 1 SA for protecting the Phase 2 negotiations.

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

The evaluator configured the TOE for AES-CBC-128, AES-CBC-256, and AES-GCM-256 and verified via logs and packet captures that the connection was successfully established for each algorithm.

2.3.10.5 NDcPP22e:FCS_IPSEC_EXT.1.5

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

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

Section 7.1.5 of the ST states that IKEv2 is supported for Phase 1 negotiation. The TSF supports IKEv2 with NAT Traversal.

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

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

As indicated in the ST, the TOE only supports IKEv2 and always attempts NAT traversal, so there is no further configuration needed.

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

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

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

Test 1: Not applicable. The TOE does not support IKEv1.

Test 2: The evaluator configured the TOE such that a VPN session from a test server traversed a NAT device. The evaluator initiated an IPsec connection and observed that the TOE correctly negotiated the NAT connection to establish a protected IPsec connection.

2.3.10.6 NDcPP22e:FCS_IPSEC_EXT.1.6

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

Section 7.1.5 of the ST states each of the IPsec implementations within the TSF support the following encryption ciphers for IKEv2 SA and Child SAs. A configuration parser ensures that the strength of the IKEv2 SA cipher cannot be less than the IKEv2 Child SA cipher:

- AES-CBC-128,
- AES-CBC-256,
- AES-GCM-256

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

Section “Configure IPsec” in the **Admin Guide** provides instructions for creating an IKEv2 IPsec VPN profile. This includes choosing the ESP Hash method, the hash or integrity algorithm to use in the Proposal for authentication, and the ESP Encryption method, the Encapsulating Security Protocol (ESP) encryption algorithm for this Proposal.

Additionally, it includes instructions for choosing the integrity algorithm (hash) used in the IKEv2 policy and the encryption algorithm used to establish the Phase 1 SA for protecting the Phase 2 negotiations.

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

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

2.3.10.7 NDcPP22e:FCS_IPSEC_EXT.1.7

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

Section 7.1.5 of the ST states the TSF enforces the following lifetime values for IKEv2 SAs:

- Between 2 minutes and 24 hours (configurable)

The TSF enforces the following lifetime values for IKEv2 Child SAs:

- Between 2 minutes and 24 hours (configurable),
- Number of bytes (configurable)

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 24 hours is exceeded (e.g. configure a time value of 23h 45min to ensure the actual rekey is performed no later than 24h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 1 SA value of 24 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 24 hours. It is not permitted to configure a value of 24 hours if that leads to an actual rekey after more than 24hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (TD0800 applied)

Section “Configure IPsec” in the **Admin Guide** provides further detail for configuring the lifetime of the SA in seconds. It also provides instructions for configuring the IKEv2 Child SA lifetime in seconds.

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

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

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

b) Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 24 hours for the Phase 1 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 1 SA lifetime that exceeds the lifetime Phase 1 SA on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 24 hours, and determine that a new Phase 1 SA is negotiated on or before 24 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 1 negotiation. (TD0800 applied)

Test 1: Not applicable. The TOE does not support data size based rekey limits for phase 1.

Test 2: The evaluator configured the TOE to have a 24-hour IKE limit and configured the test peer with a 25 hour limit. The evaluator then connected the IPsec VPN between the test peer and the TOE. The evaluator observed through logs and packet captures that the connection was successful and that the TOE rekeyed well before the configured time limit was reached ensuring that the keys were renegotiated successfully and there was no data loss during the rekey.

2.3.10.8 NDcPP22e:FCS_IPSEC_EXT.1.8

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

Section 7.1.5 of the ST states the TSF enforces the following lifetime values for IKEv2 SAs:

- Between 2 minutes and 24 hours (configurable)

The TSF enforces the following lifetime values for IKEv2 Child SAs:

- Between 2 minutes and 24 hours (configurable),
- Number of bytes (configurable)

Guidance Assurance Activities: The evaluator shall verify that the values for SA lifetimes can be configured and that the instructions for doing so are located in the guidance documentation. If time-based limits are supported, configuring the limit may lead to a rekey no later than the specified limit. For some implementations, it may be necessary, though, to configure the TOE with a lower time value to ensure a rekey is performed before the maximum SA lifetime of 8 hours is exceeded (e.g. configure a time value of 7h 45min to ensure the actual rekey is performed no later than 8h). The evaluator shall verify that the guidance documentation allows the Administrator to configure the Phase 2 SA value of 8 hours or provides sufficient instruction about the time value to configure to ensure the rekey is performed no later than the maximum SA lifetime of 8 hours. It is not permitted to configure a value of 8 hours if that leads to an actual rekey after more than 8 hours. Currently there are no values mandated for the number of bytes, the evaluator just ensures that this can be configured if selected in the requirement. (TD0800 applied)

Section “Configure IPsec” in the **Admin Guide** provides instructions for configuring a VPN connection including configuration of the lifetime in seconds and in kilobytes. It also provides further detail for configuring the lifetime of the SA in seconds.

Additionally, the same section provides instructions for configuring the IKEv2 Child SA lifetime in seconds and kilobytes and the IKEv2 SA lifetime in seconds.

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

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

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

Test 2: If 'length of time' is selected as the SA lifetime measure, the evaluator shall configure a maximum lifetime no later than 8 hours for the Phase 2 SA following the guidance documentation. The evaluator shall configure a test peer with a Phase 2 SA lifetime that exceeds the Phase 2 SA lifetime on the TOE. The evaluator shall establish a SA between the TOE and the test peer, maintain the Phase 1 SA for 8 hours, and determine that once a new Phase 2 SA is negotiated when or before 8 hours has elapsed. The evaluator shall verify that the TOE initiates a Phase 2 negotiation. (TD0800 applied)

Test 1: The evaluator configured a max lifetime of 512 MB and established a connection with a VPN peer. The IPsec SA timed out after the packet size was exceeded and the connection reset. A new Phase 2 SA negotiation was required.

Test 2: The evaluator configured a max lifetime of 8 hours on the TOE and a max lifetime of 9 hours on the test peer. The evaluator then connected the IPsec VPN between the test peer and the TOE. The evaluator observed through logs and packet captures that the connection was successful and that the IPsec SA timed out before the configured time limit was reached and a new SA was negotiated.

2.3.10.9 NDcPP22e:FCS_IPSEC_EXT.1.9

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

Section 7.1.5 states The TSF supports the following ECDH groups for key establishment. The “x” in g^x mod p is generated from the output of the DRBG and is twice the strength of the negotiated group (i.e., between 256 and 512 bits). Nonces are randomly generated according to the negotiated ECDH group and are at least 128 bits and at least half the size of the negotiated PRF. The negotiated ECDH group will be selected based on the strongest mutually accepted group configured on each IPsec endpoint:

- 19 (256-bit Random ECP),
- 20 (384-bit Random ECP),

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.10.10 NDcPP22e:FCS_IPSEC_EXT.1.10

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

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

Section 7.1.5 states The TSF supports the following ECDH groups for key establishment. The “x” in g^x mod p is generated from the output of the DRBG and is twice the strength of the negotiated group (i.e., between 256 and 512 bits). Nonces are randomly generated according to the negotiated ECDH group and are at least 128 bits and at least half the size of the negotiated PRF. The negotiated ECDH group will be selected based on the strongest mutually accepted group configured on each IPsec endpoint:

- 19 (256-bit Random ECP),
- 20 (384-bit Random ECP),

Guidance Assurance Activities: None Defined

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

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

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

See TSS assurance activity above.

2.3.10.11 NDcPP22e:FCS_IPSEC_EXT.1.11

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

Section 7.1.5 states The TSF supports the following ECDH groups for key establishment. The “x” in g^x mod p is generated from the output of the DRBG and is twice the strength of the negotiated group (i.e., between 256 and 512 bits). Nonces are randomly generated according to the negotiated ECDH group and are at least 128 bits and at least half the size of the negotiated PRF. The negotiated ECDH group will be selected based on the strongest mutually accepted group configured on each IPsec endpoint:

- 19 (256-bit Random ECP),
- 20 (384-bit Random ECP),

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

Section "Configure IPsec" in the **Admin Guide** outlines the supported DH groups and specifies that only DH groups 19 and 20 are supported when in CC mode.

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

The evaluator made a successful IPsec connection to an IPsec peer using each of the claimed DH groups. The evaluator was able to capture each DH group using a packet capture to ensure the correct DH group was used.

2.3.10.12 NDcPP22e:FCS_IPSEC_EXT.1.12

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

Section 7.1.5 of the ST states each of the IPsec implementations within the TSF support the following encryption ciphers for IKEv2 SA and Child SAs. A configuration parser ensures that the strength of the IKEv2 SA cipher cannot be less than the IKEv2 Child SA cipher:

- AES-CBC-128,
- AES-CBC-256,
- AES-GCM-256

Guidance Assurance Activities: None Defined

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

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

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

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

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

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

Test 2: The evaluator established the IKE SA with AES-CBC-128 and attempted to establish the ESP SA with AES-CBC-256. The evaluator observed that the connection failed.

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

Test 4: This test was performed in conjunction with Test 3 above.

2.3.10.13 NDcPP22e:FCS_IPSEC_EXT.1.13

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

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

Section 7.1.5 of the ST states The TSF supports the following algorithms for X.509 peer certificate authentication:

- RSA (2048 or 3072 bits),
- ECDSA (384 bits or greater)

This is consistent with the claims for FCS_COP.1/SigGen.

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

The evaluator shall check that the guidance documentation describes how pre-shared keys are to be generated and established. The description in the guidance documentation shall also indicate how pre-shared key

establishment is accomplished for TOEs that can generate a pre-shared key as well as TOEs that simply use a pre-shared key.

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

Sections “Configure X.509 Certificate”, “Generate a Private Key”, “Generate a CSR”, and “Import CA Chain” in the **Admin Guide** provide steps for enrolling a certificate and specifying the certificate contents including the Key Type: RSA or ECDSA and the Key Size. This key generation process on the Director results in private keys stored in binary format on each TOE component that will use the certificate associated with the generated keys.

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

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.

2.3.10.14 NDcPP22e:FCS_IPSEC_EXT.1.14

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

Section 7.1.5 of the ST states that for IKE authentication using X.509 certificates, the peer identifier presented in the certificate is matched to the peer identifier configured in the IPsec VPN policy. The following are acceptable fields for use in certificate matching:

- Distinguished Name (DN)
- Subject Alternative Name
  - IP address
  - FQDN
  - User FQDN (email address)

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

Section “Configure IPsec” in the **Admin Guide** includes instructions for setting the field (id type and id string) for the matching rule to the Distinguished Name (DN) and Subject Alternative Name (SAN). As indicated by the ST and **Admin Guide**, the TOE supports Distinguished Name (DN) matching, and the following Subject Alternative Name (SAN) fields: IP address, FQDN, and user FQDN (email address).

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

The evaluator shall perform the following tests:

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

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

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

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

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

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

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

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

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

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

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

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

Test 1: Not applicable. The TOE does not support CN identifiers.

Test 2: These results are iterated for IPsec RSA and IPsec ECDSA. For part 1 of this test, the evaluator alternately configured a test peer to use an authentication certificate with the correct SAN: IP address, DNS address (FQDN), and user FQDN (depending on fields the TOE supports). The evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was successful. CN identifiers are not supported by the TOE so the second part of the test is not applicable.

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

Test 4: These results are iterated for IPsec RSA and IPsec ECDSA. For this test, the evaluator alternately configured the TOE to look for each of the supported SAN reference identifiers. The evaluator then configured the test peer to use a certificate that would present an incorrect SAN reference identifier and a correct CN reference identifier as CN checking is not prioritized over SAN (CN is not supported). In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE and confirmed that the connection was rejected. The evaluator then confirmed that the TOE rejected the connection.

Test 5: These results are iterated for IPsec RSA and IPsec ECDSA. The evaluator configured a test peer to send an authentication certificate with an authorized DN and confirmed that the connection succeeded.

Test 6: These results are iterated for IPsec RSA and IPsec ECDSA. (Part A) For this test, the evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate but with a DN containing a duplicate CN. In each case, the evaluator attempted to establish an IPsec connection,

and confirmed that the TOE rejected the certificate containing a duplicate CN. (Part B) For this test, the evaluator configured a test peer to first send an authentication certificate with an authorized DN, and then a nearly identical certificate in which the Organization field of the DN has a trailing null character (86) appended. In each case, the evaluator attempted to establish an IPsec connection, and confirmed that the TOE rejected the certificate containing a DN Organization containing an appended null character.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.11 IPsec Protocol (VPNGW13:FCS_IPSEC_EXT.1)

2.3.11.1 VPNGW13:FCS_IPSEC_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.2 VPNGW13:FCS_IPSEC_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.3 VPNGW13:FCS_IPSEC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.4 VPNGW13:FCS_IPSEC_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.5 VPNGW13:FCS_IPSEC_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.6 VPNGW13:FCS_IPSEC_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities[1]: None Defined

2.3.11.7 VPNGW13:FCS_IPSEC_EXT.1.7

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.8 VPNGW13:FCS_IPSEC_EXT.1.8

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

[1] TD0813 was not applied as the project received a waiver from NIAP since the TD was under discussion internally.

Testing Assurance Activities: None Defined

2.3.11.9 VPNGW13:FCS_IPSEC_EXT.1.9

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.10 VPNGW13:FCS_IPSEC_EXT.1.10

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.11 VPNGW13:FCS_IPSEC_EXT.1.11

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.12 VPNGW13:FCS_IPSEC_EXT.1.12

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.13 VPNGW13:FCS_IPSEC_EXT.1.13

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.11.14 VPNGW13:FCS_IPSEC_EXT.1.14

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Pre-shared keys are not selected.

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

Pre-shared keys are not selected.

Component Testing Assurance Activities: None Defined

2.3.12 NTP Protocol (NDcPP22e:FCS_NTP_EXT.1)

2.3.12.1 NDcPP22e:FCS_NTP_EXT.1.1

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

Section 7.1.5 of the ST states The TSF supports synchronization of its system clock with an external NTPv4 server. Authenticity and integrity of the time updates is protected using an IPsec tunnel.

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

Section “Configure NTP Servers” in the **Admin Guide** provides instructions to the administrator for configuring the TOE to synchronize its real-time clock with an external NTP time source.

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

The evaluator first started by configuring the test computer to act as an external NTP server and confirmed that the NTP daemon is running the NTP version specified in the ST (NTP v4). The evaluator then used the TOE's WebUI to configure 3 NTP servers for time synchronization. The evaluator then started a packet capture between the TOE and the NTP server and confirmed that the specified version of NTP (NTP v4) is used.

2.3.12.2 NDcPP22e:FCS_NTP_EXT.1.2

TSS Assurance Activities: None Defined

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

Assurance Activity Note:

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

Sections “Add a Static Route” and “Configure Static Routes” in the **Admin Guide** provide instructions to the administrator for configuring the TOE to protect its NTP traffic through IPsec using an IPsec profile.

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

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

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

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

The evaluator first started by setting up an IPsec profile on the TOE to be used for NTP traffic protection using the TOE's Web UI. The evaluator then configured the first NTP Time Server on the TOE to use IPsec to protect NTP traffic. The evaluator then started a packet capture between the TOE and the NTP server and confirmed that the NTP traffic was protected and sent through ESP packets confirming the use of IPsec. The evaluator then updated the tie on the NTP server and verified that the TOE synchronized its system time successfully.

2.3.12.3 NDcPP22e:FCS_NTP_EXT.1.3

TSS Assurance Activities: None Defined

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

As indicated in the ST, the TOE does not accept neither broadcast nor multicast NTP packets, so there is no further configuration needed.

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

The evaluator configured the NTP servers to support periodic time updates to broadcast and multicast addresses. The evaluator then confirmed that the TOE is configured to synchronize its' time through the NTP server. The evaluator also confirmed the time on the NTP server (which matches the time on the TOE indicating that the time between the TOE and the NTP server is synchronized) by running the 'date' command. At this point, the evaluator started a packet capture between the TOE and the NTP server to verify that the NTP server is supporting periodic time updates to broadcast and multicast addresses. The evaluator then updated the date and time on the NTP server. The evaluator then verified that the timestamp on the TOE did not change. This confirms that the TOE rejected to update its timestamp after receipt of the broadcast and multicast packets.

2.3.12.4 NDcPP22e:FCS_NTP_EXT.1.4

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

(TD0528 applied)

Test 1: Refer to the results of NDcPP22e:FCS_NTP_EXT.1.1-t1 where the evaluator showed that the TOE supports configuration of at least three (3) NTP time sources. Also, refer to the results of NDcPP22e:FPT_STM_EXT.1-t2 where the evaluator showed that the TOE successfully synchronizes its' time through the configured NTP servers.

Test 2: The evaluator first configured the same NTP server (172.16.16.254) that was used to test valid time updates successfully throughout FCS_NTP_EXT.1 testing to send broadcast packets directly to the TOE's IP address. The evaluator then configured the TOE to use NTP for time synchronization but provided an IP address (172.16.16.200) that is different from the one configured to provide time updates to the TOE. This prevents the TOE from reaching out to the configured NTP server (172.16.16.254) with client packets to ask for time updates. The evaluator also confirmed the time on the NTP server (which matches the time on the TOE indicating that the time between the TOE and the NTP server is synchronized) by running the 'date' command. At this point, the evaluator started a packet capture between the TOE and the NTP server to verify that the NTP server (172.16.16.254) is in fact sending NTP broadcast packets directly to the TOE's IP address. The evaluator then updated the date on the NTP servers

and verified that the timestamp on the TOE did not change. This confirms that the TOE rejected to update its' timestamp after receipt of the broadcast and multicast packets.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.13 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.3.13.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.13.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

The evaluator shall examine the TSS to determine that it specifies the DRBG type, identifies the entropy source(s) seeding the DRBG, and state the assumed or calculated min-entropy supplied either separately by each source or the min-entropy contained in the combined seed value.

The Entropy description is provided in a separate (non-ST) document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section 7.1.3 of the ST states The TOE headend (Director/Analytics) uses a HMAC_DRBG and CTR_DRBG for generation of seed values for private/public keypairs. All entropy is seeded from Linux RNG from CPU jitter as the primary entropy source. Both DRBGs are seeded with an estimated 256 bits of entropy.

VOS uses a CTR_DRBG with 256 bits of entropy seeded from RDRAND instruction in the compatible CPU.

Component Guidance Assurance Activities: Documentation shall be produced - and the evaluator shall perform the activities - in accordance with Appendix D of [NDcPP].

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

The Entropy description is provided in a separate (non-ST) Versa proprietary document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section “Enable FIPS Mode” in the **Admin Guide** outline what must be done in order to enable FIPS mode which configures the switch to use the proper DRBG methods.

Component Testing Assurance Activities: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration.

If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. 'generate one block of random bits' means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 - 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.

Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does not use a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one

string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

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

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

2.3.14.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.14.2 NDcPP22e:FCS_SSHS_EXT.1.2

TSS Assurance Activities: The evaluator shall check to ensure that the TSS contains a list of supported public key algorithms that are accepted for client authentication and that this list is consistent with signature verification algorithms selected in FCS_COP.1/SigGen (e.g., accepting EC keys requires corresponding Elliptic Curve Digital Signature algorithm claims).

The evaluator shall confirm that the TSS includes the description of how the TOE establishes a user identity when an SSH client presents a public key or X.509v3 certificate. For example, the TOE could verify that the SSH client's presented public key matches one that is stored within the SSH server's authorized_keys file.

If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, then the evaluator shall confirm its role in the authentication process is described in the TSS. (TD0631 applied)

Section 7.1.5 of the ST states The TSF supports password-based authentication in addition to the following public key algorithms for SSH client authentication and for the SSH host key:

- ecdsa-sha2-nistp256,
- ecdsa-sha2-nistp384,
- ecdsa-sha2-nistp521

The TSF accepts SSH clients presenting a public key found in the server’s authorized_keys file. If no match is found, the server will revert to password-based authentication.

The client authentication algorithms align with the claims for FCS_COP.1/Hash and FCS_COP.1/SigGen.

Guidance Assurance Activities: None Defined

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

Test 1: For each supported client public-key authentication algorithm, the evaluator shall configure a remote client to present a public key corresponding to that authentication method (e.g., 2048-bit RSA key when using ssh-rsa public key). The evaluator shall establish sufficient separate SSH connections with an appropriately configured remote non-TOE SSH client to demonstrate the use of all applicable public key algorithms. It is sufficient to observe the successful completion of the SSH Authentication Protocol to satisfy the intent of this test.

Test 2: The evaluator shall choose one client public key authentication algorithm supported by the TOE. The evaluator shall generate a new client key pair for that supported algorithm without configuring the TOE to recognize the associated public key for authentication. The evaluator shall use an SSH client to attempt to connect to the TOE with the new key pair and demonstrate that authentication fails.

Test 3: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication succeeds when the correct password is provided by the connecting SSH client.

Test 4: [Conditional] If password-based authentication method has been selected in the FCS_SSHS_EXT.1.2, the evaluator shall configure the TOE to accept password-based authentication and demonstrate that user authentication fails when the incorrect password is provided by the connecting SSH client.

(TD0631 applied)

Test 1: This test has been performed as part of FIA_UIA_EXT.1-t1 where the evaluator demonstrated a successful login using a valid public key.

Test 2: This was performed as part of FIA_UIA_EXT.1-t1 where the evaluator demonstrated an unsuccessful login using an unrecognized public key.

Test 3: The evaluator attempted to connect to the TOE using a SSH client alternately using the correct and incorrect password. The evaluator found that only the correct password would yield a successful SSH session.

Test 4: This was performed as part of Test 3.

2.3.14.3 NDcPP22e:FCS_SSHS_EXT.1.3

TSS Assurance Activities: The evaluator shall check that the TSS describes how 'large packets' in terms of RFC 4253 are detected and handled.

Section 7.1.5 of the ST states e TSF will reject large packets as defined by RFC 4253 if the payload size exceeds 262,131 bytes.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that if the TOE receives a packet larger than that specified in this component, that packet is dropped.

The evaluator created and sent a packet to the TOE that was larger than the maximum packet size. The TOE rejected the packet and the connection was closed.

2.3.14.4 NDcPP22e:FCS_SSHS_EXT.1.4

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

Section 7.1.5 of the ST states The TSF supports password-based authentication in addition to the following public key algorithms for SSH client authentication and for the SSH host key:

- ecdsa-sha2-nistp256,
- ecdsa-sha2-nistp384,
- ecdsa-sha2-nistp521

The TSF accepts SSH clients presenting a public key found in the server's authorized_keys file. If no match is found, the server will revert to password-based authentication.

The TSF supports the following encryption algorithms:

- aes128-ctr
- aes256-ctr
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com

The TSF supports the following message integrity algorithms:

- hmac-sha2-256
- hmac-sha2-512
- implicit

The TSF supports the following key exchange algorithms:

- ecdh-sha2-nistp256
- ecdh-sha2-nistp384
- ecdh-sha2-nistp521

This information matches the SFR claims made.

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

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

Testing Assurance Activities: The evaluator must ensure that only claimed ciphers and cryptographic primitives are used to establish an SSH connection. To verify this, the evaluator shall start session establishment for an SSH connection from a remote client (referred to as 'remote endpoint' below). The evaluator shall capture the traffic exchanged between the TOE and the remote endpoint during protocol negotiation (e.g. using a packet capture tool or information provided by the endpoint, respectively). The evaluator shall verify from the captured traffic that the TOE offers all the ciphers defined in the TSS for the TOE for SSH sessions, but no additional ones compared to the definition in the TSS. The evaluator shall perform one successful negotiation of an SSH session to verify that the TOE behaves as expected. It is sufficient to observe the successful negotiation of the session to satisfy the intent of the test. If the evaluator detects that not all ciphers defined in the TSS for SSH are supported by the TOE and/or the TOE supports one or more additional ciphers not defined in the TSS for SSH, the test shall be regarded as failed.

The evaluator attempted to connect to the TOE using a SSH client alternately using each of the ciphers that can be claimed to verify they are supported with successful connections. In each case the evaluator viewed the Server: Key Exchange Init packet and saw that no additional ciphers beyond those that were claimed were seen as supported.

2.3.14.5 NDcPP22e:FCS_SSHS_EXT.1.5

TSS Assurance Activities: The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the SSH server's host public key algorithms supported are specified and that they are identical to those listed for this component. (TD0631 applied)

Section 7.1.5 of the ST states the TSF supports password-based authentication in addition to the following public key algorithms for SSH client authentication and for the SSH host key:

- ecdsa-sha2-nistp256,
- ecdsa-sha2-nistp384

The TSF accepts SSH clients presenting a public key found in the server's authorized_keys file. If no match is found, the server will revert to password-based authentication.

The TSF supports the following encryption algorithms:

- aes128-ctr
- aes256-ctr
- aes128-gcm@openssh.com
- aes256-gcm@openssh.com

The TSF supports the following message integrity algorithms:

- hmac-sha2-256
- hmac-sha2-512
- implicit

The TSF supports the following key exchange algorithms:

- ecdh-sha2-nistp256
- ecdh-sha2-nistp384

This information matches the SFR claims made.

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

Section "Perform Manual Hardening for SSH" in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration. This section also provides instructions for generating the public/private key pairs and enabling public key authentication for SSH.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

Testing Assurance Activities: Test objective: This test case is meant to validate that the TOE server will support host public keys of the claimed algorithm types.

Test 1: The evaluator shall configure (only if required by the TOE) the TOE to use each of the claimed host public key algorithms. The evaluator will then use an SSH client to confirm that the client can authenticate the TOE server public key using the claimed algorithm. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Has effectively been moved to FCS_SSHS_EXT.1.2.

Test objective: This negative test case is meant to validate that the TOE server does not support host public key algorithms that are not claimed.

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

(TD0631 applied)

Test 1: The evaluator established an SSH connection with the TOE using each claimed host key algorithm. The connection was successful.

Test 2: The evaluator attempted to connect to the TOE using a host public key algorithm that is not included in the ST selection and observed that the connection failed.

2.3.14.6 NDcPP22e:FCS_SSHS_EXT.1.6

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

Section 7.1.5 of the ST states the TSF supports the following message integrity algorithms:

- hmac-sha2-256
- hmac-sha2-512
- implicit

This information matches the SFR claims made.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed data integrity algorithms are used in SSH connections with the TOE (specifically, that the 'none' MAC algorithm is not allowed).

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration. This section also provides instructions for generating the public/private key pairs and enabling public key authentication for SSH.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

Testing Assurance Activities: Test 1 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall establish an SSH connection using each of the algorithms, except 'implicit', specified by the requirement. It is sufficient to observe (on the wire) the successful negotiation of the algorithm to satisfy the intent of the test.

Note: To ensure the observed algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 2 [conditional, if an HMAC or AEAD_AES_*_GCM algorithm is selected in the ST]: The evaluator shall configure an SSH client to only allow a MAC algorithm that is not included the ST selection. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Note: To ensure the proposed MAC algorithm is used, the evaluator shall ensure a non-aes*- gcm@openssh.com encryption algorithm is negotiated while performing this test.

Test 1: The evaluator established an SSH connection with the TOE using each of the claimed integrity algorithms. The evaluator observed a successful connection using each claimed integrity algorithm.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using the HMAC-MD5 algorithm. The connection attempt failed.

2.3.14.7 NDcPP22e:FCS_SSHS_EXT.1.7

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

Section 7.1.5 of the ST states The TSF supports the following key exchange algorithms:

- ecdh-sha2-nistp256
- ecdh-sha2-nistp384

- ecdh-sha2-nistp521

This information matches the SFR claims made.

Guidance Assurance Activities: The evaluator shall also check the guidance documentation to ensure that it contains instructions to the Security Administrator on how to ensure that only the allowed key exchange algorithms are used in SSH connections with the TOE.

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration. This section also provides instructions for generating the public/private key pairs and enabling public key authentication for SSH.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

Testing Assurance Activities: Test 1: The evaluator shall configure an SSH client to only allow the diffie-hellman-group1-sha1 key exchange. The evaluator shall attempt to connect from the SSH client to the TOE and observe that the attempt fails.

Test 2: For each allowed key exchange method, the evaluator shall configure an SSH client to only allow that method for key exchange, attempt to connect from the client to the TOE, and observe that the attempt succeeds.

Test 1: The evaluator attempted to establish an SSH connection with the TOE using diffiehellman-group1-sha1 key exchange. The connection attempt failed.

Test 2: The evaluator attempted to establish an SSH connection with the TOE using each allowed key exchange method: ecdh-sha2-nistp256, ecdh-sha2-nistp384, and ecdh-sha2-nistp521. The connection succeeded in each case.

2.3.14.8 NDcPP22e:FCS_SSHS_EXT.1.8

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

a) Both thresholds are checked by the TOE.

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

Section 7.1.5 of the ST states The TSF will automatically trigger a rekey if either of the configured thresholds (one gigabyte and one hour) are reached.

Guidance Assurance Activities: If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, then the evaluator shall check that the guidance documentation describes how to configure those thresholds. Either the allowed values are specified in the guidance documentation and must not exceed the limits

specified in the SFR (one hour of session time, one gigabyte of transmitted traffic) or the TOE must not accept values beyond the limits specified in the SFR. The evaluator shall check that the guidance documentation describes that the TOE reacts to the first threshold reached.

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration. This section also provides instructions for generating the public/private key pairs and enabling public key authentication for SSH.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

The SSH rekeying thresholds are not configurable; they are hardcoded and will occur approximately at 1 hour of time or after 1 GB of data has been transmitted, whichever occurs first.

Testing Assurance Activities: The evaluator needs to perform testing that rekeying is performed according to the description in the TSS. The evaluator shall test both, the time-based threshold and the traffic-based threshold.

For testing of the time-based threshold the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one hour of session time but the value used for testing shall not exceed one hour. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

For testing of the traffic-based threshold the evaluator shall use the TOE to connect to an SSH client and shall transmit data to and/or receive data from the TOE within the active SSH session until the threshold for data protected by either encryption key is reached. It is acceptable if the rekey occurs before the threshold is reached (e.g. because the traffic is counted according to one of the alternatives given in the Application Note for FCS_SSHS_EXT.1.8).

The evaluator shall verify that more data has been transmitted within the SSH session than the threshold allows and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

Testing does not necessarily have to be performed with the threshold configured at the maximum allowed value of one gigabyte of transferred traffic, but the value used for testing shall not exceed one gigabyte. The evaluator needs to ensure that the rekeying has been initiated by the TOE and not by the SSH client that is connected to the TOE.

If one or more thresholds that are checked by the TOE to fulfil the SFR are configurable, the evaluator needs to verify that the threshold(s) can be configured as described in the guidance documentation and the evaluator needs to test that modification of the thresholds is restricted to Security Administrators (as required by FMT_MOF.1(3)/Functions).

In cases where data transfer threshold could not be reached due to hardware limitations it is acceptable to omit testing of this (SSH rekeying based on data transfer threshold) threshold if both the following conditions are met:

a) An argument is present in the TSS section describing this hardware-based limitation and

b) All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

The SSH data and time rekey thresholds are not configurable so the default of 1 GB and 1 hour were tested. The evaluator attempted to connect to the TOE using a SSH client generating 1 GB of data and verified that a rekey happened when the threshold was reached. The evaluator attempted to connect to the TOE using a SSH client waiting an hour and verified that a rekey happened right before the 1-hour threshold.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.3.15.1 NDcPP22e:FCS_TLSS_EXT.1.1

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

Section 7.1.5 of the ST states the TSF implements the following ciphersuites for each implementation of TLS in accordance with RFC 5289:

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_DHE_RSA_WITH_AES_128_GCM_SHA256

- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384

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

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

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

Test 2: The evaluator shall send a Client Hello to the server with a list of ciphersuites that does not contain any of the ciphersuites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the server denies the connection.

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

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

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

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

The evaluator shall verify that the Finished message (Content type hexadecimal 16 and handshake message type hexadecimal 14) is sent immediately after the server's ChangeCipherSpec (Content type hexadecimal 14) message. The evaluator shall examine the Finished message (encrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 11 22 33 44 55...) and confirm that it does not contain unencrypted data (unencrypted example in hexadecimal of a TLS record containing a Finished message, 16 03 03 00 40 14 00 00 0c...), by verifying that the first byte of the encrypted Finished message does not equal hexadecimal 14 for at least

one of three test messages. There is a chance that an encrypted Finished message contains a hexadecimal value of '14' at the position where a plaintext Finished message would contain the message type code '14'. If the observed Finished message contains a hexadecimal value of '14' at the position where the plaintext Finished message would contain the message type code, the test shall be repeated three times in total. In case the value of '14' can be observed in all three tests it can be assumed that the Finished message has indeed been sent in plaintext and the test has to be regarded as 'failed'. Otherwise it has to be assumed that the observation of the value '14' has been due to chance and that the Finished message has indeed been sent encrypted. In that latter case the test shall be regarded as 'passed'.

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

Test 2: The evaluator attempted to connect to the TOE using the TLS_NULL_WITH_NULL_NULL ciphersuite and observed that the connection was rejected. The evaluator then attempted to connect to the TOE using all ciphers except those defined in the PP and observed that the TOE rejected the connection attempt.

Test 3: (a,b): The evaluator made connection attempts from a client to the TOE. The client implementation of the TLS protocol was modified as stated in parts a and b of the assurance activity. In Scenario 3 b), the evaluator observed the packet capture and ensured that the first byte of the encrypted Finished message does not equal 0x14.

2.3.15.2 NDcPP22e:FCS_TLSS_EXT.1.2

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

Section 7.1.5 of the ST states the TSF implements TLSv1.2 as a server without mutual authentication for securing management connections to the Director UI. TLS v1.1, 1.0, SSL 3.0, SSL 2.0, and any other unsupported TLS versions will be rejected.

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

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

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

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

2.3.15.3 NDcPP22e:FCS_TLSS_EXT.1.3

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

Section 7.1.5 of the ST states The TSF supports ECDH with secp256r1, secp384r1, and secp521r1 curves and ffdhe2048 group for use in key establishment.

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

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

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

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

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

Test 2: [conditional] If DHE ciphersuites are supported, the evaluator shall repeat the following test for each supported parameter size. If any configuration is necessary, the evaluator shall configure the TOE to use a supported Diffie-Hellman parameter size. The evaluator shall attempt a connection using a supported DHE ciphersuite. The evaluator shall verify (through a packet capture or instrumented client) that the TOE sends a

Server Key Exchange Message where p Length is consistent with the message are the ones configured Diffie-Hellman parameter size(s).

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

Test 1: The evaluator attempted to establish a TLS session with the TOE and configure the remote peer (i.e., the client) to offer each supported ECDH key exchange. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is accepted by the TOE when using supported ECDH key exchange groups. The evaluator then configured the remote peer (i.e., the client) to offer up an unsupported ECDHE curve size (P-192) and verified that the TOE rejected the connection.

Test 2: The evaluator attempted to establish a TLS session with the TOE and configure the remote peer (i.e., the client) to offer each DHE key size supported by the TOE. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is accepted by the TOE when using supported DHE key sizes.

Test 3: Not applicable. The TOE does not support RSA ciphersuites.

2.3.15.4 NDcPP22e:FCS_TLSS_EXT.1.4

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

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

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

Section 7.1.5 of the ST states the TSF does not support session resumption or session tickets.

Guidance Assurance Activities: None Defined

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Test 1: The evaluator configured the remote peer (i.e., client) to send a Client Hello with a zero-length session ticket and a zero-length session ID. The TOE does not send a NewSessionTicket packet to the remote peer indicating that it does not support session resumption based on session tickets. The TOE also sent a different session ID than the client proposed one, indicating that it does not support session resumption based on session IDs.

Test 2: Not applicable. The TOE does not support session resumption based on session IDs.

Test 3: Not applicable. The TOE does not support session resumption based on session tickets.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4 User data protection (FDP)

2.4.1 Full Residual Information Protection (STFFW14E:FDP_RIP.2)

2.4.1.1 STFFW14E:FDP_RIP.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 7.1.13 of the ST states The TSF ensures that no data will be reused when processing network packets. When received on an interface, traffic is copied from the NIC into the userspace Versa services (bypassing Linux kernel) using mbufs. Each packet that is copied is associated with an mbuf. When the software is initialized, a pool of mbufs is instantiated and handed to the Versa applications for processing. The data plane management service will recycle mbufs for packets that enter and leave the device, in which case the mbuf is zeroized before being allocated for the next packet.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.5 Firewall (FFW)

2.5.1 Stateful Traffic Filtering (STFFW14E:FFW_RUL_EXT.1)

2.5.1.1 STFFW14E:FFW_RUL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.1.2 STFFW14E:FFW_RUL_EXT.1.2

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

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

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

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

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

Section 7.1.13 of the ST states The TSF supports a stateful packet filtering policy, and the following attributes are configurable within stateful traffic filtering rules for the associated protocols:

ICMPv4	Type

	Code
ICMPv6	Type
	Code
IPv4 (RFC 791)	Source address
	Destination Address
	Transport Layer Protocol
IPv6 (RFC 8200)	Source address
	Destination Address
	Transport Layer Protocol
TCP (RFC 793)	Source Port
	Destination Port
UDP (RFC 768)	Source Port
	Destination Port

For a stateful firewall policy, administrators may configure the following enforcement actions:

- Logging
  - Start
  - End
  - Both
  - Never
- Action
  - Allow—Allow sessions that match the configured rule to pass.
  - Deny—Drop sessions that match the rule.
  - Reject—Drop sessions that match the rule and sends a TCP reset (RST) or a UDP ICMP port unreachable message.

All interfaces of the TOE are subject to processing rules which can be applied to each distinct network interface or sub-interface as described above.

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

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

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

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

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

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

Section “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** states that traffic policies are defined in terms of the static routes which in turn are associated with branch interfaces. So, a policy may be defined to allow traffic from “WAN” to “LAN”. No branch interface will forward traffic until policies have been

configured and applied to that interface. Traffic will not be forwarded unless it's explicitly permitted by at least one policy rule, thus an implicit "deny-all" rule is applied to all interfaces to which any traffic filtering rule has been applied. The implicit deny-all rule is executed after all admin-defined rules have been executed, and will result in dropping all traffic that has not been explicitly permitted, or explicitly denied. If an administrator wants to log all denied traffic, a rule entry should be added that denies all traffic and logs it, e.g. by either adding a rule at the end of a policy to explicitly drop and log all traffic, or by setting the Default Action for the policy to block all traffic, and enabling logging for the default rule.

Additionally, the same section in the **Admin Guide** shows that for each rule, the administrator can specify a rule action, to allow, deny, reject or apply security profile to matching traffic with an intrusion policy.

It also outlines each protocol type, the required header field inspection and the specific keywords that can be used to configure a rule. For IPv4 and IPv6 the required header fields include the source and destination address and transport protocol. For ICMPv4 and ICMPv6 it includes the type and code, and for TCP and UDP it includes the source and destination port.

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

- ICMPv4

o Type

o Code

- ICMPv6

o Type

o Code

- IPv4

o Source address

o Destination Address

o Transport Layer Protocol

- IPv6

o Source address

o Destination Address

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

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

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

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

Test 1: This test was performed as part of STFFW14e:FFW_RUL_EXT.1.9 where a firewall rule was defined using each of the protocol attributes defined in this test and packets were sent either matching or not matching the rules. To create these rules the evaluator followed the administrative guidance and found all necessary instructions were provided accurately. In each case the TOE demonstrated the appropriate permit/deny behavior.

Test 2: This test was performed as part of STFFW14e:FFW_RUL_EXT.1.9 where a firewall rule was defined using each of the protocol attributes defined in this test and packets were sent either matching or not matching the rules. To create these rules the evaluator followed the administrative guidance and found all necessary instructions were provided accurately. The evaluator found that these rules can be applied to all types of supported network interface.

2.5.1.3 STFFW14E:FFW_RUL_EXT.1.3

TSS Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Guidance Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Testing Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

2.5.1.4 STFFW14E:FFW_RUL_EXT.1.4

TSS Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Guidance Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

Testing Assurance Activities: See FFW_RUL_EXT.1.2

See FFW_RUL_EXT.1.2

2.5.1.5 STFFW14E:FFW_RUL_EXT.1.5

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

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

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

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

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

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

Section 7.1.13 of the ST states The TSF is able to classify traffic according to stateful TCP and UDP sessions. To classify the traffic, stateful firewall verifies its destination port and then tracks the state of the traffic and monitors every interaction of each connection until it is closed. Stateful firewall grants or rejects access based not only on

port and protocol but also on the packet history in the state table. When stateful firewall receives a packet, it checks the state table for an established connection or for a request for the incoming packet from an internal host. If nothing is found, the packet's access is subject to the access policy rule.

For stateful firewall, Security Administrators configure a security access policy to classify traffic using a security access policy. A security access policy includes the stateful firewall rule that collates the defined objects and assigns an action to take based on the match conditions.

Stateful firewall focuses on examining the information in Layer 2 (link layer), Layer 3 (network), and Layer 4 (transport) packets. For these packets, their Layer 3 and 4 information (IP address and TCP/UDP port number) is verified against the information stored in the state table to confirm that they are part of the current exchange. This method increases overall firewall performance because only the initiating packets must be unencapsulated for these layers and all layers up to the application layer (Layer 7).

For more advanced inspection capabilities, stateful targets vital packets for Layer 7 (application) examination, such as the packet that initializes a connection. If the inspected packet matches an existing firewall rule that permits it, the packet is passed and an entry is added to the state table. From this point forward, because the packets in that communication session match an existing state table entry, they are allowed access without a call for further application layer inspection.

Each security access policy consists of one or more rules. Each rule consists of match criteria and enforcement actions. You can use one or more of these traffic attributes to specify the match criteria:

- IP headers
- Domain names
- Services, based on port and protocol
- Source and destination geographic location
- Source and destination IP addresses
- Source and destination zones
- Time-of-day scheduling

For TCP, the TSF uses the following attributes to determine if packets are associated with an existing session: source and destination addresses, source and destination ports, sequence number, and individual flags.

While UDP is a stateless protocol, the TSF uses the following attributes to determine if packets are associated with an existing session: source and destination addresses, source and destination ports.

ICMP is also a stateless protocol however the TSF uses the following attributes to determine if packets are associated with an existing session: source and destination addresses, type, and code.

Connections are removed after administrator-defined protocol timeout values and applied on the next received packet. If the session has been closed, the next received packet would be rejected.

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

Section “Stateful Session Behaviors” in the **Admin Guide** specifically outlines Stateful Session Behaviors and how they can be configured. It states that the TSF can classify traffic according to stateful TCP and UDP sessions. To classify traffic, stateful firewall verifies its destination port and then tracks the state of the traffic and monitors every interaction of each connection until it is closed. Stateful firewall grants or rejects access based not only on port and protocol but also on the packet history in the state table. When stateful firewall receives a packet, it checks the state table for an established connection or for a request for the incoming packet from an internal host. If nothing is found, the packet's access is subject to the access policy rule.

Connections are removed after administrator-defined protocol timeout values and applied on the next received packet. If the session has been closed, the next received packet would be rejected.

For stateful firewall, Security administrators configure a security access policy to classify traffic using a security access policy. A security access policy includes the stateful firewall rule that collates the defined objects and assigns an action to take based on the match conditions.

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

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

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

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

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

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

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

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

The evaluation team performed the tests specified in the assurance activity and confirmed that the traffic filter firewall rules operate as expected in each test scenario.

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

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

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

Test 4: The evaluator configured the TOE to allow and log network traffic for a valid UDP session. The evaluator started transmitting packets from a test server to establish a valid UDP session. The packets being sent were constructed such that they would be permitted to pass through the TOE. The evaluator also sent packets with incorrect source and destination addresses, incorrect source and destination ports, and incorrect flags and

sequence numbers. The evaluator confirmed (through logs and packet captures) that all nonmatching packets are not treated as part of the established session.

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

Test 6: Not applicable. ICMP was not selected.

Test 7: Not applicable. ICMP was not selected.

Test 8: Not applicable. ICMP was not selected.

2.5.1.6 STFFW14E:FFW_RUL_EXT.1.6

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

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

b) Fragments that cannot be completely re-assembled

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

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

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

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

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

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

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

Section 7.1.13 of the ST states the TSF may be configured to automatically drop the following packet types within an access policy (which in turn may associated with a LEF profile for logging):

- All fragmented packets (counters)
- Loose-source routing
- Strict-source routing

- Record route
- Broadcast source
- Multicast source
- Loopback source address
- Unspecified or reserved IP (RFC 5735, RFC 3513)

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

Section "Stateful Session Behaviors" in the **Admin Guide** provides instructions for configuring the traffic flow control as required for the CC-evaluated configuration. For each of the default stateful traffic filtering rules on all network traffic as specified in FFW_RUL_EXT.1.6 in the ST, this section states that the TSF can be configured to automatically drop the following packet types (specified in FFW_RUL_EXT.1.6 in the ST) within an access policy.

Section "Stateful Session Behaviors" in the **Admin Guide** states that all interfaces of the TOE are subject to processing rules which can be applied to each distinct network interface or sub- interface. Traffic will not be forwarded unless it's explicitly permitted by at least one policy rule, thus an implicit "deny-all" rule is applied to all interfaces to which any traffic filtering rule has been applied. The implicit deny-all rule is executed after all admin-defined rules have been executed, and will result in dropping all traffic that has not been explicitly permitted, or explicitly denied. If an administrator wants to log all denied traffic, a rule entry should be added that denies all traffic and logs it, e.g. by either adding a rule at the end of a policy to explicitly drop and log all traffic, or by setting the Default Action for the policy to block all traffic, and enabling logging for the default rule.

Section "Custom IPS" in the **Admin Guide** states that an enabled rule causes the system to generate intrusion events for (and optionally block) traffic matching the rule. It outlines the process for the administrator to create a new intrusion policy and includes the settings for specifying if rules should drop packets and log or simply log events that trigger the policy. It also provides instructions for enabling or disabling a rule within an individual intrusion policy as well as specifying the action the system takes if monitored conditions trigger the rule. The rule's state can be set to generate events or drop and generate events.

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

Test 1: The evaluator shall test each of the conditions for automatic packet rejection in turn. In each case, the TOE should be configured to allow all network traffic and the evaluator shall generate a packet or packet fragment that

is to be rejected. The evaluator shall use packet captures to ensure that the unallowable packet or packet fragment is not passed through the TOE.

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

Test 1 and test 2: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured firewall rules to allow network traffic and enabled rejected packet logging. The evaluator generated the following types of packets which the TOE rejected and logged. Note, that item “c” can’t apply to IPv6 as IPv6 does not implement broadcast addressing.

Test 1-1, Invalid Fragment IPv4
Test 1-2, Invalid Fragment IPv6
Test 2-1, Incomplete Fragment IPv4
Test 2-2, Incomplete Fragment IPv6
Test 5-1, Invalid Broadcast Source Address IPv4
Test 6-1, Invalid Multicast Source Address IPv4
Test 6-2, Invalid Multicast Source Address IPv6
Test 7-1, Invalid Loopback Source Address IPv4
Test 7-2, Invalid Loopback Source Address IPv6
Test 10-1, Invalid Future Source Address IPv4
Test 10-2, Invalid Future Destination Address IPv4
Test 11-1, Invalid Future Source Address IPv6
Test 11-2, Invalid Future Destination Address IPv6
Test 12-1, Invalid IP Options Loose Source Routing IPv4
Test 12-2, Invalid IP Options Strict Source Routing IPv4
Test 12-3, Invalid IP Options Record Route IPv4

2.5.1.7 STFFW14E:FFW_RUL_EXT.1.7

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

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

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

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

Section 7.1.13 of the ST states the TSF may be configured to automatically drop the following packet types within an access policy (which in turn may associated with a LEF profile for logging):

- Packets where the source address is equal to the address of the network interface where the network packet was received
- Packets where the source or destination address of the network packet is a link-local address
- Packets where the source address does not belong to the networks associated with the network interface where the network packet was received – the access policy will determine which zones the rules are associated with and therefore the networks associated with each zone.

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

Sections “Configure Intrusion Detection and Prevention” and “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** outline the process for the administrator to create a new intrusion policy and includes the settings for specifying if rules should drop packets and log or simply log events that trigger the policy. It also provides instructions for enabling or disabling a rule within an individual intrusion policy as well as specifying the action the system takes if monitored conditions trigger the rule. For the required rules in FFW_RUL_EXT.1.7, the traffic will be dropped by default, and auditing of these events can be enabled by enabling logging on the Default Action of the Access Control Policy.

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

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

These tests were performed for both IPv4 and IPv6 traffic.

Test 1: The evaluator configured the TOE to drop and log network traffic where the source address of the packet matches that of the TOE network interface upon which the traffic was received. The evaluator generated suitable traffic to match the configured rule and confirmed via packet capture and logs that the traffic is dropped and a log message is generated.

Test 2: The evaluator configured the TOE to drop and log network traffic where the source IP address of the packet fails to match the network reachability information of the interface to which it is targeted. The evaluator generated suitable traffic to match the configured rule and confirmed via packet capture and logs that the traffic is dropped and a log message is generated. The evaluator then configured the TOE to drop and log network traffic where the source or destination address of the network packet is a link-local address and confirmed that when traffic matching that rule was generated, the TOE dropped the traffic and generated a log message.

2.5.1.8 STFFW14E:FFW_RUL_EXT.1.8

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

Section 7.1.13 of the ST states all packets are serviced by the kernel hooks. Every packet that enters the networking system (incoming or outgoing) will trigger these hooks as it progresses through the stack, allowing programs that register with these hooks to interact with the traffic at key points. The kernel modules associated with network processing register at these hooks in order to ensure that the traffic conforms to the conditions laid out by the firewall rules. If the Versa networking services have not been initialized successfully, or have failed to load, no packets will be permitted to flow through the TOE.

The TSF also supports DoS protection to ensure that the traffic processed cannot exceed the capabilities of the networking stack, and will ensure that access policies are consistently enforced even during DoS attacks.

VOS devices support routed, or Layer 3, interfaces. The interface associated with each physical network interface (PNIC) or virtual network interface (VNIC) is configured with an IP address. Based on the routing configuration, the traffic from the tenant is forwarded to the interfaces on the VOS device. The VOS device supports several routing instances or virtual routing functions (VRFs). Each VRF is associated with one or more interfaces on the VOS device, and the VOS device supports static routing, BGP, and OSPF.

The traffic of a particular tenant enters a VOS device because the IP address of the routed interface is the next-hop address of the tenant traffic's final destination. Firewall policies can be applied on the traffic entering a VOS device, and the traffic is routed to the next hop (based on routing configuration) only if the security policy allows the traffic to be forwarded.

All interfaces of the TOE are subject to processing rules which can be applied to each distinct network interface or sub-interface as described above.

The TSF is able to classify traffic according to stateful TCP and UDP sessions. To classify the traffic, stateful firewall verifies its destination port and then tracks the state of the traffic and monitors every interaction of each connection until it is closed. Stateful firewall grants or rejects access based not only on port and protocol but also on the packet history in the state table. When stateful firewall receives a packet, it checks the state table for an established connection or for a request for the incoming packet from an internal host. If nothing is found, the packet's access is subject to the access policy rule.

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

Section “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** states that the TOE processes rules in a top-down order, meaning the top-most applies first, followed by the seconds, until it finds a match. The matching rule action is applied on the network traffic. It is recommended that you order rules from permissive to restrictive to prevent blocking of legitimate traffic. You can add a deny rule to the bottom to handle traffic that does not match any other rules.

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

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

Test 1: The evaluator configured the TOE with conflicting firewall rules and verified that the TOE rejected these conflicting rules. The evaluator first configured a permit rule to permit traffic to a specific destination and a second rule to deny traffic to the same destination. The evaluator confirmed that the connection was successful. Subsequently, the evaluator configured a deny rule first with the second rule (permit). The evaluator observed that the connection failed. The original firewall rules were implemented using IPv4. The test was then repeated using IPv6. Both test cases behaved as expected.

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

regardless of the specificity of the rule. The original firewall rules were implemented using IPv4. The test was then repeated using IPv6. Both test cases behaved as expected.

2.5.1.9 STFFW14E:FFW_RUL_EXT.1.9

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

Section 7.1.13 of the ST states A VOS firewall device can be installed as a bare metal or a virtual machine (VM). The security policies are applied to the traffic that enters the firewall through physical or virtual interfaces. The VOS firewall recognizes VLAN tags for incoming traffic and adds the appropriate VLAN tags to the outbound traffic.

The following are typical scenarios for configuring traffic on a PNIC:

- Non-VLAN Traffic—Traffic that is not tagged with VLAN and enters the firewall using PNIC is mapped to a single tenant.

- VLAN Traffic—Traffic tagged with VLAN is mapped to one or more tenant. The VOS device creates a unique subinterface for each VLAN. Use one or more VLAN to configure the traffic identification for each tenant hosted on the VOS device.

The following are typical scenarios for configuring traffic on a VNIC:

- VLAN-mapped VNIC— If the VNIC is mapped by the hypervisor to a specific VLAN for the traffic that enters through the PNIC, then when the traffic enters the firewall through the VNIC, the VLAN is already stripped by the hypervisor. Therefore, all the traffic that enters through the VNIC is mapped to a single tenant. In this scenario, a single VNIC cannot support traffic from multiple tenants.

- PNIC-mapped VNIC with non-VLAN traffic—When the hypervisor directly maps the VNIC to the PNIC without any VLAN stripping and if the traffic that enters the firewall through the VNIC is not VLAN tagged, all traffic that enters through the VNIC is mapped to a single tenant.

- PNIC-mapped VNIC with VLAN traffic—When the hypervisor directly maps the VNIC to the PNIC without any VLAN stripping and if the traffic that enters the firewall through the VNIC is VLAN tagged, traffic that belongs to different VLANs is mapped to one or more tenants.

- You create a unique subinterface for each VLAN. You can configure the traffic identification using one or more VLANs for each tenant hosted on the VOS device.

The TSF supports a stateful packet filtering policy, and the following attributes are configurable within stateful traffic filtering rules for the associated protocols:

ICMPv4	Type
	Code
ICMPv6	Type
	Code
IPv4 (RFC 791)	Source address
	Destination Address
	Transport Layer Protocol
IPv6 (RFC 8200)	Source address
	Destination Address
	Transport Layer Protocol
TCP (RFC 793)	Source Port
	Destination Port
UDP (RFC 768)	Source Port
	Destination Port

For stateful firewall, Security Administrators configure a security access policy to classify traffic using a security access policy. A security access policy includes the stateful firewall rule that collates the defined objects and assigns an action to take based on the match conditions.

Each security access policy consists of one or more rules. Each rule consists of match criteria and enforcement actions. You can use one or more of these traffic attributes to specify the match criteria:

- IP headers
- Domain names
- Services, based on port and protocol
- Source and destination geographic location
- Source and destination IP addresses
- Source and destination zones
- Time-of-day scheduling

It is recommended that in a security policy to configure more specific rules first and then configure generic rules, followed by a final deny-all rule. The TOE does not prevent administrators from applying conflicting rules.

Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes the behavior if no rules or special conditions apply to the network traffic. If the behavior is configurable, the evaluator shall

verify that the guidance documentation provides the appropriate instructions to configure the behavior to deny packets with no matching rules.

Sections “Configure Intrusion Detection and Prevention” and “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** outline the process for the administrator to create a new intrusion policy and includes the settings for specifying if rules should drop packets and log or simply log events that trigger the policy. It also provides instructions for enabling or disabling a rule within an individual intrusion policy as well as specifying the action the system takes if monitored conditions trigger the rule. For the required rules in FFW_RUL_EXT.1.7, the traffic will be dropped by default, and auditing of these events can be enabled by enabling logging on the Default Action of the Access Control Policy.

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

The evaluator defined several tests variations to exercise the attributes and rules from FFW_RUL_EXT.1.2. The evaluator generated traffic to match specific aspects of the configured firewall rule set and confirmed that all attributes demonstrated permit, deny and log for each test case. The following variations were tested.

Test Variations
Part 1-ICMP Type and code value matches Permit Rule
Part 2-ICMP Type PermitRule type mismatch
Part 3-ICMP Type PermitRule code mismatch
Part 4-ICMP Type DenyRule type match
Part 5-ICMP Type DenyRule type mismatch
Part 6-ICMP Type DenyRule code mismatch
Part 7-ICMP6 Type and code PermitRule match
Part 8-ICMP6 Type PermitRule type mismatch
Part 9-ICMP6 Type PermitRule code mismatch
Part 10-ICMP6 Type DenyRule match
Part 11-ICMP6 Type DenyRule type mismatch
Part 12-ICMP6 Type DenyRule code mismatch
Part 13-IPv4 TCP PermitRule Match
Part 14-IPv4 PermitRule src addr mismatch
Part 15-IPv4 PermitRule dest addr mismatch
Part 16-IPv4 PermitRule protocol mismatch
Part 17-IPv4 DenyRule Match
Part 18-IPv4 DenyRule src addr mismatch
Part 19-IPv4 DenyRule dest addr mismatch
Part 20-IPv4 DenyRule protocol mismatch
Part 21-IPv6 TCP PermitRule Match
Part 22-IPv6 PermitRule src addr mismatch
Part 23-IPv6 PermitRule dest addr mismatch
Part 24-IPv4 PermitRule protocol mismatch

Part 25-IPv6 DenyRule Match
Part 26-IPv6 DenyRule src addr mismatch
Part 27-IPv6 DenyRule dest addr mismatch
Part 28-IPv6 DenyRule protocol mismatch
Part 29-IPv4 TCP PermitRule Match
Part 30-IPv4 TCP PermitRule src port mismatch
Part 31-IPv4 TCP PermitRule dest port mismatch
Part 32-IPv4 TCP DenyRule Match
Part 33-IPv4 TCP DenyRule src port mismatch
Part 34-IPv4 TCP DenyRule dest port mismatch
Part 35-IPv6 TCP PermitRule match
Part 36-IPv6 TCP PermitRule src port mismatch
Part 37-IPv6 TCP PermitRule dest port mismatch
Part 38-IPv6 TCP DenyRule match
Part 39-IPv6 TCP DenyRule src port mismatch
Part 40-IPv6 TCP DenyRule dest port mismatch
Part 41-IPv4 UDP PermitRule Match
Part 42-IPv4 UDP PermitRule src port mismatch
Part 43-IPv4 UDP PermitRule dest port mismatch
Part 44-IPv4 UDP DenyRule Match
Part 45-IPv4 UDP DenyRule src port mismatch
Part 46-IPv4 UDP DenyRule dest port mismatch
Part 47-IPv6 UDP PermitRule match
Part 48-IPv6 UDP PermitRule src port mismatch
Part 49-IPv6 UDP PermitRule dest port mismatch
Part 50-IPv6 UDP DenyRule match
Part 51-IPv6 UDP DenyRule src port mismatch
Part 52-IPv6 UDP DenyRule dest port mismatch

2.5.1.10 STFFW14E:FFW_RUL_EXT.1.10

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

Section 7.1.13 of the ST states The TSF implements TCP SYN flood protection under DoS Protection or Zone Protection profiles. DoS protection is applied where the TOE is deployed at the perimeter of a network on which services are accessed externally through the VOS, where Zone Protection is applicable to an entire zone. Thresholds may be set for the number of TCP packets per second across three levels:

- Alarm rate – the rate at which a log will be generated
- Action rate – The rate at which the TOE will drop packets. By default, the firewall uses SYN Cookies to track valid connections, but may be configured to randomly drop packets.
- Maximum rate – The rate at which all packets would be dropped for a configurable duration.

Stale connections (including half-open connections) will be removed after the defined protocol timeout value.

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

Section “Configure Intrusion Detection and Prevention” in the **Admin Guide**, sub-section “configure DoS protection settings” describes the SYN attack prevention option which helps protect network hosts against SYN floods which are indicated by any traffic containing excessive incomplete connections to hosts on the network. Instructions are provided for configuring Syn Attack Prevention which prevents half-open ‘embryonic’ connections. The administrator can specify the number of SYN packets per number of seconds. After a timeout period elapses, if the rate condition has stopped, the event generation and packet dropping stops.

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

The evaluator configured a TCP half-open connection limit on the TOE. The evaluator then generated TCP SYN requests which would pass through the TOE to a target system using a randomized source IP address and common destination IP address. The number of SYN requests sent exceeded the TCP half-open threshold defined on the TOE. TCP SYN-ACK messages were not acknowledged. Using a packet capture, the evaluator verified that once the defined TCP half-open threshold had been reached, subsequent TCP SYN packets are not transmitted to the target system. The evaluator verified that when the configured threshold is reached the TOE behaved as claimed in the Security Target.

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

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

Section 7.1.5 of the ST states during the VOS boot process, first the root partition and filesystem is located, checked and mounted. Next the init process is started, which runs the initialization scripts. These scripts involve different startup events that eventually bring the Versa networking services online. All packets are serviced by the kernel hooks. Every packet that enters the networking system (incoming or outgoing) will trigger these hooks as it progresses through the stack, allowing programs that register with these hooks to interact with the traffic at key points. The kernel modules associated with network processing register at these hooks in order to ensure that the traffic conforms to the conditions laid out by the firewall rules. If the Versa networking services have not been initialized successfully, or have failed to load, no packets will be permitted to flow through the TOE.

The TSF also supports DoS protection to ensure that the traffic processed cannot exceed the capabilities of the networking stack, and will ensure that access policies are consistently enforced even during DoS attacks.

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

See subsequent test evaluation activities.

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

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

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

Test 1: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to block network traffic directed at a specific destination. The evaluator started transmitting packets from a test server. The packets being sent were constructed such that they should be blocked by the configure rule. The evaluator rebooted the TOE and monitored traffic on both sides of the TOE during the reboot process. The evaluator observed that while packets were being delivered to the TOE ingress side, no matching packets were being passed from the egress side.

Test 2: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit network traffic directed at a specific destination. The evaluator started transmitting packets from a test server. The packets being sent were constructed such that they should be accepted by the configured rule. The evaluator rebooted the TOE and monitored traffic on both sides of the TOE during the reboot process. The evaluator observed that there is a gap during which no traffic is passed shortly after the reboot is started until just prior to the login prompt being presented by the TOE. This

demonstrates that none of the generated network traffic is permitted through the firewall during initialization and is only permitted once initialization is complete.

2.6 IDENTIFICATION AND AUTHENTICATION (FIA)

2.6.1 AUTHENTICATION FAILURE MANAGEMENT (NDCPP22E:FIA_AFL.1)

2.6.1.1 NDCPP22E:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.1.2 NDCPP22E:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 7.1.7 of the ST states the Director enforces account lockouts for the GUI and CLI after the configured number of unsuccessful authentication attempts has been reached. On each attempt, a counter is incremented until the value is reached, upon which the user account will be locked and will no longer be able to login until:

- The administrator-configured unlock time period elapses
- Another administrator manually unlocks the account

The time-based unlock feature ensures that administrators will be automatically re-enabled, to prevent situations where all administrative accounts are permanently locked out. The local console is not subject to lockout.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the remote administrator to once again successfully log on is described for each 'action' specified (if that option is chosen). If different actions or mechanisms are implemented depending on the secure protocol employed (e.g., TLS vs. SSH), all must be described.

The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

Sections “Set User Account Unlock Time” and “Set Director User Timeout and Authentication Lockout” in the **Admin Guide** provide instructions to set the number of successive unsuccessful authentication attempts before an account becomes locked. The instructions state that the administrator can define the policy to keep the account locked until the locked account can be automatically unlocked after a specified period. This section provides instructions to define the unlock period.

These sections also states that the TOE can be configured to block remote login requests from a user account for a configurable period of time after a configurable number of failed remote login attempts. Admin accounts are never locked out from using the local console. Additionally, these sections provide instructions for manually unlocking a locked user account.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which remote administrators access the TOE (e.g. any passwords entered as part of establishing the connection protocol or the remote administrator application):

a) Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.

b) Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows.

If the administrator action selection in FIA_AFL.1.2 is included in the ST then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote administrator's access results in successful access (when using valid credentials for that administrator).

If the time period selection in FIA_AFL.1.2 is included in the ST then the evaluator shall wait for just less than the time period configured in Test 1 and show that an authorisation attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorisation attempt using valid credentials results in successful access.

Test 1: The evaluator configured the limit for the maximum number of failed login attempts at 2. The evaluator then logged in to the TOE using incorrect credentials two times. The evaluator then attempted to use correct credentials and found that the account had been locked out in each case.

Test 2: Continuing Test 1, the evaluator then successfully unlocked the account in each case using the procedures found in the guidance.

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

2.6.2.1 NDcPP22e:FIA_PMG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

(TD0792 applied)

Section 7.1.7 of the ST states Administrative passwords may be composed of any combination of upper and lower case letters, numbers, and the following special characters:

! @ # $ % ^ & * ( ) ” ’ + , - . / : ; < = > ? @ [\ ] _ ` { | } ~

Minimum password length is configurable to between 8 and 25 characters.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it:

a) identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and

b) provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

Sections “Configure Director and System User Authentication” and “Set Password Complexity” in the **Admin Guide** provide instructions for configuring the minimum password length and also provide guidance on strong password composition using the identified characters consistent with the ST. Minimum password length can be set to be 8-25 characters with the default being 8 characters. Passwords must be at least 8 alphanumeric characters of mixed case and must include at least one numeric character and one special character. It cannot be a word that appears in a dictionary or include consecutive repeating characters.

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

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

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

Test 1: The evaluator successfully set valid user passwords on the TOE with compositions including the following:

-Min configurable length
-demonstrating special character set
-demonstrating number set
-demonstrating uppercase set
-demonstrating lowercase set
-Max length

Test 2: The evaluator attempted to set invalid user passwords on the TOE with compositions including the following. All invalid password attempts were rejected.

-short password
-long password
-no numbers
-no special character

-no uppercase
-no lowercase

2.6.3 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.6.3.1 NDcPP22e:FIA_UAU.7.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

There are no preparatory steps needed to ensure authentication data is not revealed. The ST states that when logging in, the TOE will not echo passwords such that passwords are not inadvertently displayed to the user and any other users that might be able to view the login display. The TOE replaces the entered password character with a "*" character or does not show any characters at all.

Sections "Log In Remotely to Director" and "Log In from a Local Console" in the **Admin Guide** describe and provide instructions for logging in remotely via Web UI and CLI as well as logging in locally via the serial console.

Component Testing Assurance Activities: The evaluator shall perform the following test for each method of local login allowed:

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

Test 1- This test was performed as part of the tests for FIA_UIA_EXT.1 where the evaluator observed that passwords are obscured on the console logins.

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

2.6.4.1 NDcPP22e:FIA_UAU_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

This is covered under FIA_UIA_EXT.1.

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

This is covered under FIA_UIA_EXT.1.

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

This is covered under FIA_UIA_EXT.1.

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

2.6.5.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.5.2 NDcPP22e:FIA_UIA_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a 'successful logon'.

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

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

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

Section 7.1.7 of the ST states the TOE can identify administrators by a unique ID and enforces their authentication before granting them access to any TSF management interfaces. The TOE supports two levels of administrators, which are the roles Admin and Operator.

The TOE requires each administrator to be successfully identified and authenticated before access is granted to any management functions except viewing the login banner.

Administrative access to the TOE is facilitated through the Director CLI (console or and management port via SSH), and through the Director GUI (management port via HTTPS/TLS. The TOE mediates all administrative actions through the CLI and GUI. Once a potential administrative user attempts to access an administrative interface either locally or remotely, the TOE prompts the user for a username and password. Only after the administrative user presents the correct authentication credentials will access to the TOE administrative functionality be granted. No access can the administrative functionality of the TOE until an administrator is successfully identified and authenticated.

Security Administrators are identified through their login session to the Director GUI and CLI. From this session, the administrator may access CLIs on each component from within existing authenticated IPsec tunnels between TOE components using usernames and passwords configured on each device. With the exception of the Director management interfaces, all TOE components will only be accessible via the Director through the IPsec channel. Local console interfaces may be enabled on Controller and Branch devices for emergency access situations.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps (e.g., establishing credential material such as pre-shared keys, tunnels, certificates, etc.) to logging in are described. For each supported the login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the allowed services.

All TOE components are entirely managed through the Director. The Director provides the ability for security administrators to securely configure and manage the TOE via a locally connected terminal (console), a remote SSH connection or an HTTPS/TLS Web-based GUI connection. The majority of the administrative functionality is available only through the Web GUI.

Section “Generate Director X.509 Certificate and Configure TLS Ciphers” in the **Admin Guide** provides instructions for configuring the TLS versions and TLS cipher suites to the approved ones claimed in the Security Target.

Section “Perform Manual Hardening for SSH” in the **Admin Guide** provides instructions for enabling CC mode on the Director. This section states that enabling CC mode will restrict the SSH algorithms to the approved ones claimed in the Security Target. CC Mode must be enabled in the evaluated configuration.

Section “Log In Remotely to Director” in the **Admin Guide** states that the system by default only supports SSH and HTTPS security protocols for management. The system is required to support only the cipher suites, version, and protocols claimed in the Security Target.

Component Testing Assurance Activities: The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

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

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

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

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

Test 1: The evaluator performed an unsuccessful and successful logon of each type (SSH pubkey, SSH password, Web UI and Local Console) using bad and good credentials respectively, and observed that when providing correct credentials, the connection was successful, but providing incorrect credentials resulted in a denial of access.

Test 2: There are no services available prior to login. The evaluator was able to observe the TOE displayed a banner to the user before login as identified in the tests performed in FTA_TAB.1.

Test 3: No functions were available to the administrator accessing the console with the exception of displaying the banner.

Test 4: The evaluator determined that each of the TOE's components (Director and Branches) require authentication before TSF actions can be performed. Each component of the TOE is managed by a Director and requires the administrator to successfully log into the Director using valid credentials to gain access to any TOE services. The Director has SSH, Web UI, and a local console, all of which have an authentication mechanism. The results for FIA_UIA_EXT.1, test 1 show that the operator has no access to any commands via the login screens (except viewing the TOE banner) before authenticating via each authentication method.

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

2.6.6.1 NDcPP22e:FIA_X509_EXT.1.1/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

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

way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store).

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

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

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

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

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

f) Test 6: The evaluator shall modify any byte in the last byte of the certificate and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

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

h) The following tests are run when a minimum certificate path length of three certificates is implemented.

Test 8: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall conduct the following tests:

Test 8a: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a valid chain of EC certificates (terminating in a trusted CA certificate), where

the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

Test 8b: (Conditional on TOE ability to process CA certificates presented in certificate message) The test shall be designed in a way such that only the EC root certificate is designated as a trust anchor, and by setting up the trust store in a way that the EC Intermediate CA certificate needs to be provided, together with the leaf certificate, from outside the TOE to complete the chain (e.g. by storing only the EC root CA certificate in the trust store). The evaluator shall present the TOE with a chain of EC certificates (terminating in a trusted CA certificate), where the intermediate certificate in the certificate chain uses an explicit format version of the Elliptic Curve parameters in the public key information field, and is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

Test 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

(TD0527 12/2020 update applied)

Test 1a and 1b: For this test, the evaluator alternately configured the TOE to have and then not have the trusted root CA used by the test peer to anchor all of its certificates. In each case, the evaluator then attempted a connection between the test peer and the TOE and confirmed that the connection only succeeded when the trusted root CA was properly configured forming a valid certificate chain.

Test 2: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid and 2) that is expired and 3) issued by an intermediate CA that is expired. In each case, the evaluator then attempted to connect the TOE to the test server. The connection succeeded only if there were no expired certificates.

Test 3: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that is revoked, and 3) issued by an intermediate CA that is revoked. In each case, the evaluator then attempted to connect the TOE to the test server and confirmed that the connection only succeeded if there were no revoked certificates.

Test 4: For this test, the evaluator alternately configured Strongswan on a test peer to send an authentication certificate 1) that is valid, 2) that has a root that refers to an OCSP revocation server where the signer lacks OCSPSigning, 3) issued by an intermediate CA whose issuer CA refers to an OCSP revocation server where the signer lacks OCSPSigning, and 4) issued by an intermediate CA referring to an OCSP revocation server where the signer lacks OCSPSigning. In each case, the evaluator then attempted to connect the IPsec VPN between the test peer and the TOE expecting the connection to succeed only if all retreived OCSP responses are signed using certificates with OCSPSigning.

Test 5: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) that has one byte in the ASN1 field changed, 3) that has one byte in the certificate signature changed, and 4) that has one byte in the certificate public key changed. In each case, the evaluator attempted to connect the TOE to the test server and verified that the connection only succeeded if the certificate was not modified/corrupted.

Test 6: This test was performed as part of Test 5.

Test 7: This test was performed as part of Test 5.

Test 8a and b: The evaluator executed a control test case with a valid ECDSA certificate chain and verified that the connection was successful. The evaluator then replaced an intermediate CA with one that has explicit curves defined. The TOE successfully rejected the invalid certificate chain.

Test 8c: The evaluator first attempted to import a certificate where the elliptic curve parameters are specified as a named curve, and signed by a trusted EC rootCA. The evaluator observed that the TOE accepted this certificate into the truststore. The evaluator then attempted to import a certificate that utilizes an explicit format version of the elliptic curve parameters, and signed by a trusted EC rootCA. The evaluator observed the TOE rejected the certificate import attempt and did not save the certificate to the truststore.

2.6.6.2 NDcPP22e:FIA_X509_EXT.1.2/Rev

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation). For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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

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

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

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

Test 1: The evaluator configured a test syslog server to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then used the TOE VPN client (IPsec) to attempt to connect to the test server. The evaluator observed that the TOE rejected the connection.

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

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

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

Section 7.1.8 of the ST states Validation checks occur on the following certificates and under the following conditions:

- Importing CA certificates into the trust store
  - BasicConstraints flag is present and set to TRUE
  - Intermediate CA certificates are properly chained
- Importing end-entity certificates into the trust store

- Valid CA chain is installed and terminated at the root CA
- CA certificates contain BasicConstraints flag and is set to TRUE

- During IPsec session establishment
  - Peer certificate signatures can be successfully verified up to the root of trust
  - Peer certificates are not expired
  - OCSP response indicates a non-revoked peer certificate and the OCSP responder certificate is valid and contains the OCSP signing extended key usage
  - Peer certificate chains containing EC certificates use named elliptic curves
  - Peer reference identifier matches the expected identifier

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

Section “” in the **Admin Guide** states that X509 certificates validation checks occur on the following certificates and under the following conditions:

Importing CA certificates into the trust store:

1. BasicConstraints flag is present and set to TRUE

2. Intermediate CA certificates are properly chained

Importing end-entity certificates into the trust store:

1. Valid CA chain is installed and terminated at the root CA

2. CA certificates contain BasicConstraints flag and is set to TRUE

During IPsec session establishment:

1. Peer certificate signatures can be successfully verified up to the root of trust

2. Peer certificates are not expired

3. OCSP response indicates a non-revoked peer certificate and the OCSP responder certificate is valid and contains the OCSP signing extended key usage

4. Peer certificate chains containing EC certificates use named elliptic curves

5. Peer reference identifier matches the expected identifier

Component Testing Assurance Activities: None Defined

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

2.6.7.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.7.2 NDcPP22e:FIA_X509_EXT.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 7.1.8 of the ST states The TSF supports X.509 certificates for authentication of IPsec peers as well as identification of the Director GUI to browser clients. Both manual and automated certificate enrollment methods are supported.

Certificates used in Director TLS server functions are stored separately in protected keystores used only by the web application server. The server configuration file specifies which certificate and trust anchor to use for server

authentication. This certificate is installed during the initial configuration as specified in the administrative guidance.

To configure a VOS device to use certificates for VPN authentication, a certificate server is configured via the Director that hosts the certificates. When the Branch or Controller device requires a certificate, it sends a certificate request to the server. The supported automated methods are: Certificate Management Protocol (CMP) and Simple Certificate Enrollment Protocol (SCEP).

Within the IPSec VPN profile, the trust anchor and certificate must be specified in order to determine which certificate to use for connections.

It also states OCSP is the supported protocol for revocation status validation. The VOS device can also be configured to enforce signature validation of the OCSP responses and terminate the IPsec session if the OCSP responder is unreachable or the response is unknown.

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

See FIA_X509_EXT.1.2/Rev where all the X509 certificates validation checks are mentioned.

Sections “Configure X.509 Certificate”, “Generate a Private Key”, “Generate a CSR”, and “Import CA Chain” in the **Admin Guide** provide steps for enrolling a certificate and specifying the certificate contents including the Key Type: RSA or ECDSA and the Key Size. This key generation process on the Director results in private keys stored in binary format on each TOE component that will use the certificate associated with the generated keys.

Section “Configure IPsec” in the **Admin Guide** provides instructions for creating VPN profiles that utilizes these X509 certificates for authentication.

Component Testing Assurance Activities: The evaluator shall perform the following test for each trusted channel:

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

The evaluator alternately configured a test peer to send an authentication certificate with valid/accessible revocation servers and an authentication certificate with revocation information referring to an inaccessible revocation server. In each case, the evaluator then attempted to make a connection between the test peer and the

TOE expecting the connection to be successful when the revocation server is accessible and when the revocation server is not accessible only if that behavior is claimed for the TOE. The evaluator observed the certificate validation checking behavior in each case and confirmed that it was consistent with the actions selected in FIA_X509_EXT.2.2(1) and FIA_X509_EXT.2.2(2) in the ST.

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

2.6.8.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.8.2 NDcPP22e:FIA_X509_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Device-specific information is not claimed.

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

Section “Generate a CSR” in the **Admin Guide** provides instructions for creating a certificate request which include establishing the fields: Country, Organization, Organization Unit and Common Name.

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

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

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

Test 1: The evaluator followed operational guidance to log into the TOE and then generate the CSR. The evaluator captured the generated request and ensured that it contains the information claimed in the ST.

Test 2: The evaluator attempted to import a certificate without a valid certification path and the import failed. The evaluator then attempted to import a certificate with a valid certification path and the import succeeded.

2.7 Security management (FMT)

2.7.1 Management of Security Functions Behaviour (ndcPP22e:FMT_MOF.1/Functions)

2.7.1.1 ndcPP22e:FMT_MOF.1.1/Functions

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see chapter 2.4.1.1.

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

Section 7.1.9 of the ST states all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

The TOE provides the ability for Security Administrators to access and modify TSF data, such as audit data, configuration data, certificates and private keys, security policies (IPS, firewall and VPN), and all other security configuration data.

Abilities to disable, enable, determine, and modify configuration settings is determined by the roles (and the privileges therein) assigned to each account. These privileges apply only to the accounts with Admin role, whereas accounts with Operator role do not have these privileges.

The TOE restricts the ability to perform administrative functions such as starting and stopping services (IPsec tunnels, etc.) and configuration of the log export functions to the Security Administrator. The configuration of the IPsec tunnels for securing distributed TOE traffic is also restricted to Security Administrators.

Management of the X.509 certificate trust store is restricted to Security Administrators. Refer to FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, and FIA_X509_EXT.3 for additional information.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE) are performed to include required configuration settings.

As indicated by the ST, all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console). The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Testing Assurance Activities: Test 1 (if 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity without prior authentication as security administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to modify the security related parameters can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

Test 2 (if 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as Security Administrator. The effects of the modifications should be confirmed.

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

Test 1 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

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

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

Test 1 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify the behaviour when Local Audit Storage Space is full without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This attempt should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

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

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

determine the behaviour of the selected functions without administrator authentication shall fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

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

For test cases 1 and 3, the evaluator (as a security administrator), attempted to modify the audit data behavior of the TOE prior to authentication. This attempt failed.

For test cases 2 and 4, the evaluator (as a security administrator also) logged onto the TOE, and then attempted to modify the audit data behavior of the TOE. This attempt succeeded.

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

2.7.2.1 NDcPP22e:FMT_MOF.1.1/ManualUpdate

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs it is required to verify the TSS to ensure that it describes how every function related to security management is realized for every TOE component and shared between different TOE components. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

There are no specific requirements for non-distributed TOEs.

Section 7.1.9 of the ST states all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

The TOE provides the ability for Security Administrators to access and modify TSF data, such as audit data, configuration data, certificates and private keys, security policies (IPS, firewall and VPN), and all other security configuration data.

Abilities to disable, enable, determine, and modify configuration settings is determined by the roles (and the privileges therein) assigned to each account. These privileges apply only to the accounts with Admin role, whereas accounts with Operator role do not have these privileges.

The TOE restricts the ability to perform administrative functions such as starting and stopping services (IPsec tunnels, etc.) and configuration of the log export functions to the Security Administrator. The configuration of the IPsec tunnels for securing distributed TOE traffic is also restricted to Security Administrators.

Management of the X.509 certificate trust store is restricted to Security Administrators. Refer to FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, and FIA_X509_EXT.3 for additional information.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that any necessary steps to perform manual update are described. The guidance documentation shall also provide warnings regarding functions that may cease to operate during the update (if applicable).

For distributed TOEs the guidance documentation shall describe all steps how to update all TOE components. This shall contain description of the order in which components need to be updated if the order is relevant to the update process. The guidance documentation shall also provide warnings regarding functions of TOE components and the overall TOE that may cease to operate during the update (if applicable).

Section “Upgrade Director and VOS” in the **Admin Guide** provides instructions for manually updating all TOE components (Director and Branches).

Component Testing Assurance Activities: The evaluator shall try to perform the update using a legitimate update image without prior authentication as Security Administrator (either by authentication as a user with no administrator privileges or without user authentication at all - depending on the configuration of the TOE). The attempt to update the TOE should fail.

The evaluator shall try to perform the update with prior authentication as Security Administrator using a legitimate update image. This attempt should be successful. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

Test 1: The TOE restricts the ability to perform manual updates to Security Administrators with prior authentication. This has been tested in NDcPP22e:FIA_UIA_EXT.1 where the evaluator showed that the TOE does not offer any services through the administrative interfaces prior to authenticating the connecting user other than the display of the TOE warning banner.

Test 2: A successful update was performed as part of FPT_TUD_EXT.1.

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

2.7.3.1 NDcPP22e:FMT_MOF.1.1/Services

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see chapter 2.4.1.1.

For non-distributed TOEs, the evaluator shall ensure the TSS lists the services the Security Administrator is able to start and stop and how that how that operation is performed.

Section 7.1.9 of the ST states all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

The TOE provides the ability for Security Administrators to access and modify TSF data, such as audit data, configuration data, certificates and private keys, security policies (IPS, firewall and VPN), and all other security configuration data.

Abilities to disable, enable, determine, and modify configuration settings is determined by the roles (and the privileges therein) assigned to each account. These privileges apply only to the accounts with Admin role, whereas accounts with Operator role do not have these privileges.

The TOE restricts the ability to perform administrative functions such as starting and stopping services (IPsec tunnels, etc.) and configuration of the log export functions to the Security Administrator. The configuration of the IPsec tunnels for securing distributed TOE traffic is also restricted to Security Administrators.

Management of the X.509 certificate trust store is restricted to Security Administrators. Refer to FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, and FIA_X509_EXT.3 for additional information.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the TSS lists the services the Security Administrator is able to start and stop and how that how that operation is performed.

As indicated by the ST, all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console). The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Testing Assurance Activities: The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) without prior authentication as Security Administrator (either by authenticating as a user with no administrator privileges, if possible, or without prior authentication at all). The attempt to enable/disable this service/these services should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to enable/disable this service/these services can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

The evaluator shall try to enable and disable at least one of the services as defined in the Application Notes for FAU_GEN.1.1 (whichever is supported by the TOE) with prior authentication as security administrator. The attempt to enable/disable this service/these services should be successful.

See Test Case FMT_MOF.1/Functions-t3 where the evaluator shows that without prior authentication as security administrator, the TOE represents its' login screen to the user to provide correct credentials for login and that no functions (modifying audit data behavior) can be executed prior to authentication.

See Test Case FMT_MOF.1/Functions-t4 where the evaluator shows that with prior authentication as security administrator, the evaluator is able to perform functions (modifying audit data behavior) and execute them.

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

2.7.4.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are accessible through an interface prior to administrator log-in are identified. For each of these functions, the evaluator shall also confirm that the

TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

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

Section 7.1.9 of the ST states all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

The TOE provides the ability for Security Administrators to access and modify TSF data, such as audit data, configuration data, certificates and private keys, security policies (IPS, firewall and VPN), and all other security configuration data.

Management of the X.509 certificate trust store is restricted to Security Administrators. Refer to FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, and FIA_X509_EXT.3 for additional information.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Guidance Assurance Activities: The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

If the TOE supports handling of X.509v3 certificates and provides a trust store, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to configure and maintain the trust store in a secure way. If the TOE supports loading of CA certificates, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to securely load CA certificates into the trust store. The evaluator shall also review the guidance documentation to determine that it explains how to designate a CA certificate a trust anchor.

The TSF data manipulating functions and the corresponding configuration information are identified or referenced throughout this AAR with the requirement to which they apply.

Section “Configure Director and System User Authentication” in the **Admin Guide** provides instructions for creating admin accounts and states that all accounts created are administrators. User accounts without the Administrator role are restricted from accessing user management functions. This is consistent with the ST which indicates that the TOE supports the predefined Administrator role. Users with the Administrator role have access to all TOE security functions including importing X.509v3 certificates to the TOE’s trust store.

See NDcPP22e:FIA_X509_EXT.2.2 which identifies the sections in the Guide which describe how the administrator role can configure and maintain the trust store including loading of CA certificates.

Component Testing Assurance Activities: No separate testing for FMT_MTD.1/CoreData is required unless one of the management functions has not already been exercised under any other SFR.

All management functions are exercised under other SFRs.

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

2.7.5.1 NDcPP22e:FMT_MTD.1.1/CryptoKeys

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: For distributed TOEs see chapter 2.4.1.1.

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

Section 7.1.9 of the ST states all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

The TOE provides the ability for Security Administrators to access and modify TSF data, such as audit data, configuration data, certificates and private keys, security policies (IPS, firewall and VPN), and all other security configuration data.

Abilities to disable, enable, determine, and modify configuration settings is determined by the roles (and the privileges therein) assigned to each account. These privileges apply only to the accounts with Admin role, whereas accounts with Operator role do not have these privileges.

The TOE restricts the ability to perform administrative functions such as starting and stopping services (IPsec tunnels, etc.) and configuration of the log export functions to the Security Administrator. The configuration of the IPsec tunnels for securing distributed TOE traffic is also restricted to Security Administrators.

Management of the X.509 certificate trust store is restricted to Security Administrators. Refer to FIA_X509_EXT.1/Rev, FIA_X509_EXT.2, and FIA_X509_EXT.3 for additional information.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

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

As indicated by the ST, all security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console). The TOE supports a Security Administrator role which is mapped to “Admin”, as well as a lower-privilege “Operator” role. All administrative accounts are assigned to only one role.

No administrative functions are available prior to authentication as a Security Administrator or to any non-privileged user.

Component Testing Assurance Activities: The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

The evaluator first confirmed that there is an admin account configured on the TOE. An admin account is the only account able to perform any cryptographic keys changes (modify, delete, generate/import) on the TOE. The evaluator then logged out of the admin account and verified that without prior authentication, no cryptographic keys changes can be performed on the TOE. The evaluator then provided the correct credentials to the admin account and logged onto the TOE using that account. The evaluator then verified that with prior authentication, cryptographic keys changes can be performed on the TOE.

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

2.7.6.1 NDcPP22e:FMT_SMF.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_SSL_EXT.1, FTA_SSL.3, FTA_TAB.1, FMT_MOF.1(1)/ManualUpdate, FMT_MOF.1(4)/AutoUpdate (if included in the ST), FIA_AFL.1, FIA_X509_EXT.2.2 (if included in the ST), FPT_TUD_EXT.1.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(2)/Services, and FMT_MOF.1(3)/Functions (for all of these SFRs that are included in the ST), FMT_MTD, FPT_TST_EXT, and any cryptographic management functions specified in the reference standards. Compliance to these requirements satisfies compliance with FMT_SMF.1.

(containing also requirements on Guidance Documentation and Tests)

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

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

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

All of the claimed management functionality has been analyzed and addresses as part of the corresponding SFRs through this document.

Section 7.1.9 of the ST states All security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

A subset of these functions is also available via the remote or local CLI. Refer to the *Versa Operating System (VOS), Versa Director and Versa Analytics Version 22.1 Common Criteria Hardening Guide* for a complete description of all security management functions available through each interface.

Section 7.1.12 of the ST states VOS devices are added to the distributed TOE using Zero-Touch Provisioning.

Two enablement methods are supported in the evaluated configuration:

- URL-based ZTP—URL-based ZTP allows an onsite administrator to activate a VOS device. The administrator connects a laptop to the VOS device and clicks an email link to a staging Controller node, which completes the activation process.

- From the CLI—A site administrator can connect to the CLI on a VOS device and run a staging script that activates the device.

Both methods require a Security Administrator with direct control of the device and requires positive enablement on each component prior to joining the distributed TOE. No registration channel is used. Once activated, TOE components will communicate via the IPsec trusted channel defined in FPT_ITC.1. Without performing the enablement steps, the TOE components will not have the parameters necessary to establish a connection. Only Security Administrators may disable and remove TOE components via the Director.

Component Guidance Assurance Activities: See TSS Assurance Activities

See TSS Assurance Activities.

Component Testing Assurance Activities: The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

All of the management functions were demonstrated throughout the course of testing.

2.7.7 Specification of Management Functions (STFFW14E:FMT_SMF.1/FFW)

2.7.7.1 STFFW14E:FMT_SMF.1.1/FFW

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

All of the claimed management functionality has been analyzed and addresses as part of the corresponding SFRs through this document.

Section 7.1.9 of the ST states All security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

A subset of these functions is also available via the remote or local CLI. Refer to the *Versa Operating System (VOS), Versa Director and Versa Analytics Version 22.1 Common Criteria Hardening Guide* for a complete description of all security management functions available through each interface.

Section 7.1.12 of the ST states VOS devices are added to the distributed TOE using Zero-Touch Provisioning.

Two enablement methods are supported in the evaluated configuration:

- URL-based ZTP—URL-based ZTP allows an onsite administrator to activate a VOS device. The administrator connects a laptop to the VOS device and clicks an email link to a staging Controller node, which completes the activation process.

- From the CLI—A site administrator can connect to the CLI on a VOS device and run a staging script that activates the device.

Both methods require a Security Administrator with direct control of the device and requires positive enablement on each component prior to joining the distributed TOE. No registration channel is used. Once activated, TOE components will communicate via the IPsec trusted channel defined in FPT_ITC.1. Without performing the enablement steps, the TOE components will not have the parameters necessary to establish a connection. Only Security Administrators may disable and remove TOE components via the Director.

Component Guidance Assurance Activities: See TSS Activity.

See TSS Assurance Activities.

Component Testing Assurance Activities: The evaluator tests management functions as part of testing the SFRs identified in section 2.4.4. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

All of the management functions were demonstrated throughout the course of testing. The management functions identified by STFFW14e:FMT_SMF.1 were used (at least) during the identified tests.

- Ability to configure firewall rules (STFFW14e:FFW_RUL_EXT.1 and VPNGW13:FPF_RUL_EXT.1)

2.7.8 Specification of Management Functions (IPS) (IPS10:FMT_SMF.1/IPS)

2.7.8.1 IPS10:FMT_SMF.1.1/IPS

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes how the IPS data analysis and reactions can be configured. This may be performed in conjunction with the evaluation of IPS_ABD_EXT.1, IPS_IPB_EXT.1, and IPS_SBD_EXT.1.

Refer to the assurance activities for IPS_ABD_EXT.1, IPS_IPB_EXT.1, and IPS_SBD_EXT.1 which describe how the IPS data analysis and reactions can be configured.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance describes the instructions for each function defined in the SFR, describes how to configure the IPS data analysis and reactions, including how to set any configurable defaults and how to configure each of the applicable analysis pattern matching methods and reaction modes

The requirement defines the following functions for which instructions can be found in the sections identified in the **Admin Guide**:

- Enable, disable signatures applied to sensor interfaces, and determine the behavior of IPS functionality – section “Configure Intrusion Detection and Prevention”.
- Modify these parameters that define the network traffic to be collected and analyzed: Sections “Configure Intrusion Detection and Prevention” and “Custom IPS”.
  - Source IP addresses (host address and network address)
  - Destination IP addresses (host address and network address)
  - Source port (TCP and UDP)
  - Destination port (TCP and UDP)
  - Protocol (IPv4 and IPv6)
  - ICMP type and code
- Update (import) signatures – section “Custom IPS”.
- Create custom signatures – section “Custom IPS”.
- Configure anomaly detection – section “Custom IPS”.
- Enable and disable actions to be taken when signature or anomaly matches are detected – section “Configure Intrusion Detection and Prevention”.
- Modify thresholds that trigger IPS reactions – section “Configure Intrusion Detection and Prevention”.
- Modify the duration of traffic blocking actions – section “Configure Intrusion Detection and Prevention”.
- Modify the known-good and known-bad lists (of IP addresses or address ranges) – Section “Configure Intrusion Detection and Prevention”.
- Configure the known-good and known-bad lists to override signature-based IPS policies – Section “Configure Intrusion Detection and Prevention”.

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

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

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

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

Other testing for this SFR is performed in conjunction with the EAs for IPS_ABD_EXT.1 and IPS_SBD_EXT.1.

Test 1: This test was performed in conjunction with part of IPSEP211:IPS_SBD_EXT.1.2-t1 testing. The evaluator created a custom rule duplicating an imported rule and observed that matching traffic triggered both of the rules, with the TOE enforcing the configured reaction to the rule (ie. generating an event and blocking the traffic).

Test 2: The evaluator disabled the "String" rules used during testing of IPS_SBD_EXT.1.2-t1 and re-ran IPS_SBD_EXT.1.2-t1 Part 2 (UDP string). Test results show IPS_SBD_EXT.1.2-t1 Part 2 failing because the blocking of traffic did not occur.

Test 3: This test was performed as part of Test 1.

2.7.9 Specification of Management Functions (VPNGW13:FMT_SMF.1/VPN)

2.7.9.1 VPNGW13:FMT_SMF.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

All of the claimed management functionality has been analyzed and addresses as part of the corresponding SFRs through this document.

Section 7.1.9 of the ST states All security management functions are performed via the Director component through the GUI or CLI (either remotely or via local console).

A subset of these functions are also available via the remote or local CLI. Refer to the *Versa Operating System (VOS), Versa Director and Versa Analytics Version 22.1 Common Criteria Hardening Guide* for a complete description of all security management functions available through each interface.

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

This activity has been performed in VPNGW13:FPF_RUL_EXT.1.4 , VPNGW13:FPF_RUL_EXT.1.5 and NDcPP2e:FMT_SMF.1 where the evaluator verified that the guidance describes and provides instructions for configuring all management functions specified in FMT_SMF.1/VPN.

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

This requirement is met throughout the testing of all other VPNGW13 requirements. All of the management functions have been exercised in other SFRs.

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

2.7.10.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.10.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.7.10.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Section 7.1.9 states The TOE is configured restrict the ability to perform privileged managing functions to authorized administrators with Admin role. Privileged managing functions are where the commands are available to configure TOE data (including saving configuration), configure administrator, restore factory default, delete configuration file, roll back configuration, and modify current admin password.

The TOE is configured restrict the ability to perform non-privileged managing functions to authorized administrators with roles Admin and Operator. Non-privileged managing functions are where the commands are available to perform a reboot, view configuration information, view log information, modify own admin password, and perform ping and traceroute tests.

Component Guidance Assurance Activities: The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration.

See FIA_UIA_EXT.1 which identifies the instructions in the **Admin Guide** for administering the TOE both locally and remotely.

Component Testing Assurance Activities: In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

All administrative interfaces were used throughout the course of testing including console for all devices, SSH and HTTPS for the Director, and SSH for the VOS branches. The evaluator also tested these communications channel by applying the FCS_SSHS and FCS_TLSS tests to the relevant channels on the Director. The evaluator applied the FCS_IPSEC testing to the VOS branches.

2.8 Packet Filtering (FPF)

2.8.1 Packet Filtering Rules (VPNGW13:FPF_RUL_EXT.1)

2.8.1.1 VPNGW13:FPF_RUL_EXT.1.1

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

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

Section 7.1.5 of the ST states during the VOS boot process, first the root partition and filesystem is located, checked and mounted. Next the init process is started, which runs the initialization scripts. These scripts involve different startup events that eventually bring the Versa networking services online. All packets are serviced by the kernel hooks. Every packet that enters the networking system (incoming or outgoing) will trigger these hooks as it progresses through the stack, allowing programs that register with these hooks to interact with the traffic at key points. The kernel modules associated with network processing register at these hooks in order to ensure that the traffic conforms to the conditions laid out by the firewall rules. If the Versa networking services have not been initialized successfully, or have failed to load, no packets will be permitted to flow through the TOE.

The TSF also supports DoS protection to ensure that the traffic processed cannot exceed the capabilities of the networking stack, and will ensure that access policies are consistently enforced even during DoS attacks.

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

See subsequent test evaluation activities.

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

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

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

Test 1: Because the TOE acts as a router, packets are sent to a TOE interface and forwarded. Thus, traffic used during STFFW14e:FFW_RUL_EXT.1-t1 testing is directed at TOE interfaces (per VPNGW13:FPF_RUL_EXT.1.1-t1) to be forwarded as appropriate to the ultimate destination. Therefore, the tests for Packet Filtering from the VPNGW

module are a subset of those identified by the FW module and thus are covered by tests described under STFFW14e:FFW_RUL_EXT.1.

Test 2: This was performed as part of testing for STFFW14e:FFW_RUL_EXT.1-t1.

2.8.1.2 VPNGW13:FPF_RUL_EXT.1.2

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet piltering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

Refer to FPF_RUL_EXT.1.4.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.3 VPNGW13:FPF_RUL_EXT.1.3

TSS Assurance Activities: There are no EAs specified for this element. Definition of packet filtering policy, association of operations with packet filtering rules, and association of these rules to network interfaces is described collectively under FPF_RUL_EXT.1.4.

Refer to FPF_RUL_EXT.1.4.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.4 VPNGW13:FPF_RUL_EXT.1.4

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

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

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

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

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

Section 7.1.13 of the ST states The TSF supports a stateful packet filtering policy, and the following attributes are configurable within stateful traffic filtering rules for the associated protocols:

ICMPv4	Type
	Code
ICMPv6	Type
	Code
IPv4 (RFC 791)	Source address
	Destination Address
	Transport Layer Protocol
IPv6 (RFC 8200)	Source address
	Destination Address
	Transport Layer Protocol

TCP (RFC 793)	Source Port
	Destination Port
UDP (RFC 768)	Source Port
	Destination Port

Versa uses industry-standard network traffic generators to perform interoperability testing to ensure RFC compliance with the above standards.

For a stateful firewall policy, administrators may configure the following enforcement actions:

- Logging
  - Start
  - End
  - Both
  - Never
- Action
  - Allow—Allow sessions that match the configured rule to pass.
  - Deny—Drop sessions that match the rule.
  - Reject—Drop sessions that match the rule and sends a TCP reset (RST) or a UDP ICMP port unreachable message.

All interfaces of the TOE are subject to processing rules which can be applied to each distinct network interface or sub-interface as described above.

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

- IPv4 (RFC 791)

o source address

o destination address

o Protocol

- IPv6 (RFC 8200)

o source address

o destination address

o next header (protocol)

- TCP (RFC 793)

o source port

o destination port

- UDP (RFC 768)

o source port

o destination port

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

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

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

Section “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** states that traffic policies are defined in terms of the static routes which in turn are associated with branch interfaces. So, a policy may be defined to allow traffic from “WAN” to “LAN”. No branch interface will forward traffic until policies have been configured and applied to that interface. Traffic will not be forwarded unless it’s explicitly permitted by at least one policy rule, thus an implicit “deny-all” rule is applied to all interfaces to which any traffic filtering rule has been applied. The implicit deny-all rule is executed after all admin-defined rules have been executed, and will result in dropping all traffic that has not been explicitly permitted, or explicitly denied. If an administrator wants to log all denied traffic, a rule entry should be added that denies all traffic and logs it, e.g. by either adding a rule at the end of a policy to explicitly drop and log all traffic, or by setting the Default Action for the policy to block all traffic, and enabling logging for the default rule.

Additionally, the same section in the **Admin Guide** shows that for each rule, the administrator can specify a rule action, to allow, deny, reject or apply security profile to matching traffic with an intrusion policy.

It also outlines each protocol type, the required header field inspection and the specific keywords that can be used to configure a rule. For IPv4 and IPv6 the required header fields include the source and destination address and

transport protocol. For ICMPv4 and ICMPv6 it includes the type and code, and for TCP and UDP it includes the source and destination port.

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

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

- IPv4

o Source address

o Destination Address

o Protocol

- IPv6

o Source Address

o Destination Address

o Next Header (Protocol)

- TCP

o Source Port

o Destination Port

- UDP

o Source Port

o Destination Port

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

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

The VPNGW13:FPF_RUL_EXT.1 testing is a subset of testing required for the STFFW14e:FFW_RUL_EXT.1 testing. Refer to the STFFW14e:FFW_RUL_EXT.1.2-t1 & STFFW14e:FFW_RUL_EXT.1.2-t2 for actual results. Also, refer to VPNGW13:FPF_RUL_EXT.1.6 for test results on various packet filtering scenarios.

2.8.1.5 VPNGW13:FPF_RUL_EXT.1.5

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

Section 7.1.13 of the ST states that a rule matches when all match criteria defined in the rule matches. All rules in the security access policy are evaluated starting with the first rule in the policy. The first rule that matches is selected and the corresponding security actions are enforced. No other rules are evaluated once a match is found.

It is recommended that in a security policy to configure more specific rules first and then configure generic rules, followed by a final deny-all rule. The TOE does not prevent administrators from applying conflicting rules.

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

Section “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** states that the TOE processes rules in a top-down order, meaning the top-most applies first, followed by the seconds, until it finds a match. The matching rule action is applied on the network traffic. It is recommended that you order rules from permissive to restrictive to prevent blocking of legitimate traffic. You can add a deny rule to the bottom to handle traffic that does not match any other rules.

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

Test 1: The evaluator shall devise two equal Packet Filtering rules with alternate operations â€“ permit and discard. The rules should then be deployed in two distinct orders and in each case the evaluator shall ensure that the first rule is enforced in both cases by generating applicable packets and using packet capture and logs for confirmation.

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

The VPNGW13:FPF_RUL_EXT.1 testing is a subset of testing required for the STFFW14e:FFW_RUL_EXT.1 testing. Refer to the STFFW14e:FFW_RUL_EXT.1.8-t1 & STFFW14e:FFW_RUL_EXT.1.8-t2 for actual results.

2.8.1.6 VPNGW13:FPF_RUL_EXT.1.6

TSS Assurance Activities: The evaluator shall verify that the TSS describes the process for applying Packet Filtering rules and also that the behavior (either by default, or as configured by the administrator) is to discard packets

when there is no rule match. The evaluator shall verify the TSS describes when the IPv4/IPv6 protocols supported by the TOE differ from the full list provided in the RFC Values for IPv4 and IPv6 table.

Section 7.1.13 of the ST states A VOS firewall device can be installed as a bare metal or a virtual machine (VM). The security policies are applied to the traffic that enters the firewall through physical or virtual interfaces. The VOS firewall recognizes VLAN tags for incoming traffic and adds the appropriate VLAN tags to the outbound traffic.

The following are typical scenarios for configuring traffic on a PNIC:

- Non-VLAN Traffic—Traffic that is not tagged with VLAN and enters the firewall using PNIC is mapped to a single tenant.

- VLAN Traffic—Traffic tagged with VLAN is mapped to one or more tenant. The VOS device creates a unique subinterface for each VLAN. Use one or more VLAN to configure the traffic identification for each tenant hosted on the VOS device.

The following are typical scenarios for configuring traffic on a VNIC:

- VLAN-mapped VNIC— If the VNIC is mapped by the hypervisor to a specific VLAN for the traffic that enters through the PNIC, then when the traffic enters the firewall through the VNIC, the VLAN is already stripped by the hypervisor. Therefore, all the traffic that enters through the VNIC is mapped to a single tenant. In this scenario, a single VNIC cannot support traffic from multiple tenants.

- PNIC-mapped VNIC with non-VLAN traffic—When the hypervisor directly maps the VNIC to the PNIC without any VLAN stripping and if the traffic that enters the firewall through the VNIC is not VLAN tagged, all traffic that enters through the VNIC is mapped to a single tenant.

- PNIC-mapped VNIC with VLAN traffic—When the hypervisor directly maps the VNIC to the PNIC without any VLAN stripping and if the traffic that enters the firewall through the VNIC is VLAN tagged, traffic that belongs to different VLANs is mapped to one or more tenants.

- You create a unique subinterface for each VLAN. You can configure the traffic identification using one or more VLANs for each tenant hosted on the VOS device.

The TSF supports a stateful packet filtering policy, and the following attributes are configurable within stateful traffic filtering rules for the associated protocols:

ICMPv4	Type
	Code
ICMPv6	Type
	Code
IPv4 (RFC 791)	Source address

	Destination Address
	Transport Layer Protocol
IPv6 (RFC 8200)	Source address
	Destination Address
	Transport Layer Protocol
TCP (RFC 793)	Source Port
	Destination Port
UDP (RFC 768)	Source Port
	Destination Port

For stateful firewall, Security Administrators configure a security access policy to classify traffic using a security access policy. A security access policy includes the stateful firewall rule that collates the defined objects and assigns an action to take based on the match conditions.

Each security access policy consists of one or more rules. Each rule consists of match criteria and enforcement actions. You can use one or more of these traffic attributes to specify the match criteria:

- IP headers
- Domain names
- Services, based on port and protocol
- Source and destination geographic location
- Source and destination IP addresses
- Source and destination zones
- Time-of-day scheduling

It is recommended that in a security policy to configure more specific rules first and then configure generic rules, followed by a final deny-all rule. The TOE does not prevent administrators from applying conflicting rules.

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

Section “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** states that traffic policies are defined in terms of the static routes which in turn are associated with branch interfaces. So, a policy may be

defined to allow traffic from “WAN” to “LAN”. No branch interface will forward traffic until policies have been configured and applied to that interface. Traffic will not be forwarded unless it’s explicitly permitted by at least one policy rule, thus an implicit “deny-all” rule is applied to all interfaces to which any traffic filtering rule has been applied. The implicit deny-all rule is executed after all admin-defined rules have been executed, and will result in dropping all traffic that has not been explicitly permitted, or explicitly denied. If an administrator wants to log all denied traffic, a rule entry should be added that denies all traffic and logs it, e.g. by either adding a rule at the end of a policy to explicitly drop and log all traffic, or by setting the Default Action for the policy to block all traffic, and enabling logging for the default rule.

Additionally, the same section in the **Admin Guide** shows that for each rule, the administrator can specify a rule action, to allow, deny, reject or apply security profile to matching traffic with an intrusion policy.

It also outlines each protocol type, the required header field inspection and the specific keywords that can be used to configure a rule. For IPv4 and IPv6 the required header fields include the source and destination address and transport protocol. For ICMPv4 and ICMPv6 it includes the type and code, and for TCP and UDP it includes the source and destination port.

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

Test 1: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

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

Test 3: The evaluator shall configure the TOE to permit and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv4 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a

specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each supported IPv4 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 4: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are permitted (i.e., by capturing the packets after passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

Test 5: The evaluator shall configure the TOE to permit all traffic except to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and within the configured source and destination addresses in order to ensure that the supported protocols are denied (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must also be denied but are not required to be logged.

Test 6: The evaluator shall configure the TOE to permit and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. Additionally, the evaluator shall configure the TOE to discard and log each supported IPv6 Transport Layer Protocol (see RFC Values for IPv4 and IPv6 table for full possible list) in conjunction with different (than those permitted above) combinations of a specific source address and specific destination address, specific source address and wildcard destination address, wildcard source address and specific destination address, and wildcard source address and wildcard destination address. The evaluator shall generate packets matching each defined IPv6 Transport Layer Protocol and outside the scope of all source and destination addresses configured above in order to ensure that the supported protocols are dropped (i.e., by capturing no applicable packets passing through the TOE) and logged. Any protocols not supported by the TOE must be denied.

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

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

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

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

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

Test 1: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit and log each defined IPv4 Transport Layer Protocol in multiple parts as described below:

- Test 1-1: IPv4 Protocol permitted and logged based on specific source and destination addresses.
- Test 1-2: IPv4 Protocol permitted and logged based on specific destination addresses.
- Test 1-3: IPv4 Protocol permitted and logged based on specific source addresses.
- Test 1-4: IPv4 Protocol permitted and logged based on wildcard addresses.

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 2: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit all traffic except to deny and log each defined IPv4 Transport Layer Protocol in multiple parts as described below:

- Test 2-1: IPv4 Protocol all permitted with some denied and logged based on specific source and destination addresses.
- Test 2-2: IPv4 Protocol all permitted with some denied and logged based on specific destination addresses.
- Test 2-3: IPv4 Protocol all permitted with some denied and logged based on specific source addresses.
- Test 2-4: IPv4 Protocol all permitted with some denied and logged based on wildcard addresses.

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 3: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit and log and to deny and log each defined IPv4 Transport Layer Protocol. Then the evaluator generated packets using these protocols which do not match any configured rule (ie., which have source and destination address outside the scope of the configured rules) and thus match the default deny rule.

- Test 3-1: IPv4 Protocol some permitted and logged and some denied and logged based on addresses outside the scope of the configured rules (matching the default deny rule)

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 4: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit and log and to deny and log each defined IPv6 Transport Layer Protocol in multiple parts as described below:

- Test 4-1: IPv6 Protocol permitted and logged based on specific source and destination addresses.
- Test 4-2: IPv6 Protocol permitted and logged based on specific destination addresses.
- Test 4-3: IPv6 Protocol permitted and logged based on specific source addresses.
- Test 4-4: IPv6 Protocol permitted and logged based on wildcard addresses.

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 5: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit all traffic except to deny and log each defined IPv6 Transport Layer Protocol in multiple parts as described below:

- Test 5-1: IPv6 Protocol all permitted with some denied and logged based on specific source and destination addresses.
- Test 5-2: IPv6 Protocol all permitted with some denied and logged based on specific destination addresses.
- Test 5-3: IPv6 Protocol all permitted with some denied and logged based on specific source addresses.
- Test 5-4: IPv6 Protocol all permitted with some denied and logged based on wildcard addresses.

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 6: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit and log and to deny and log each defined IPv6 Transport Layer Protocol. Then the evaluator generated packets using these protocols which do not match any configured

rule (ie., which have source and destination address outside the scope of the configured rules) and thus match the default deny rule.

- Test 6-1: IPv6 Protocol some permitted and logged and some denied and logged based on addresses outside the scope of the configured rules (matching the default deny rule)

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 7: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit and log traffic in multiple parts as described below:

- Test 7-1: TCP (IPv4) permitted and logged based on source port.
- Test 7-2: TCP (IPv4) permitted and logged based on destination port.
- Test 7-3: TCP (IPv4) permitted and logged based on source and destination port.
- Test 7-4: TCP (IPv6) permitted and logged based on source port.
- Test 7-5: TCP (IPv6) permitted and logged based on destination port.
- Test 7-6: TCP (IPv6) permitted and logged based on source and destination port.

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 8: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to deny and log traffic in multiple parts as described below:

- Test 8-1: TCP (IPv4) denied and logged based on source port.
- Test 8-2: TCP (IPv4) denied and logged based on destination port.
- Test 8-3: TCP (IPv4) denied and logged based on source and destination port.
- Test 8-4: TCP (IPv6) denied and logged based on source port.
- Test 8-5: TCP (IPv6) denied and logged based on destination port.
- Test 8-6: TCP (IPv6) denied and logged based on source and destination port.

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 9: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to permit and log traffic in multiple parts as described below:

- Test 9-1: UDP (IPv4) permitted and logged based on source port.
- Test 9-2: UDP (IPv4) permitted and logged based on destination port.
- Test 9-3: UDP (IPv4) permitted and logged based on source and destination port (includes UDP port 500)
- Test 9-4: UDP (IPv6) permitted and logged based on source port.
- Test 9-5: UDP (IPv6) permitted and logged based on destination port.

- Test 9-6: UDP (IPv6) permitted and logged based on source and destination port (includes UDP port 500)

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Test 10: The evaluator set up a test configuration allowing packets to be captured on both (ingress and egress) sides of the TOE. The evaluator configured the TOE to deny and log traffic in multiple parts as described below:

- Test 10-1: UDP (IPv4) denied and logged based on source port.
- Test 10-2: UDP (IPv4) denied and logged based on destination port.
- Test 10-3: UDP (IPv4) denied and logged based on source and destination port (includes UDP port 500)
- Test 10-4: UDP (IPv6) denied and logged based on source port.
- Test 10-5: UDP (IPv6) denied and logged based on destination port.
- Test 10-6: UDP (IPv6) denied and logged based on source and destination port (includes UDP port 500)

The evaluator verified that the TOE was able to filter the packets as expected and described by each of the specified tests.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.9 Protection of the TSF (FPT)

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

2.9.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Section 7.1.7 of the ST states Operator passwords are stored as SHA-512 hashes. UIs involving password manipulation (e.g. change own password, user creation, set another user's password by Admin) are designed such that there is no display of the stored credential in plaintext. During entry, passwords are obfuscated in accordance with FIA_UAU.7.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.9.2 Failure with Preservation of Secure State (Self-Test Failures) (VPNGW13:FPT_FLS.1/SelfTest)

2.9.2.1 VPNGW13:FPT_FLS.1.1/SelfTest

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 7.1.6 of the ST states The TSF runs a suite of self-tests during initial start-up and periodically during normal operation to verify its correct operation. The complete list of self-tests implemented by each TOE component are described in section 6.1.9.5. Power-on self-tests are automatically invoked by start-up scripts. Periodic self- tests are run automatically by the daemons/commands, as applicable.

The following binaries are critical to the TSF and are verified during POST using RSA 2048 SHA-256 signatures against an image verification public key preinstalled on the TOE:

- Versa-dnsd
- Versa-ntpd
- Versa-vsmd
- Versa-vmod
- Versa-dhcpd
- Versa-certd
- Versa-rtd
- Versa-l2cd
- Libcrypto.so.1.1
- Libssl.so.1.1.
- Libfipstest.so

The following output to versa-service.log demonstrates a successful verification of the cryptographic self-tests:

AES Selftest PASSED

AES-CCM Encrypt selftest PASSED

AES-CCM Decrypt selftest PASSED

AES-GCM Encrypt Selftest PASSED

AES-GCM Decrypt selftest PASSED

AES-XTS-128 selftest PASSED

AES-XTS-256 selftest PASSED

AES-CMAC selftest PASSED

HMAC SHA1

HMAC SHA224

HMAC SHA256

HMAC SHA384

HMAC SHA512

HMAC Self test PASSED

Generated RSA Sign and Verified. RSA Selftest PASSED

ECDH Self test Passed

ECDSA selftest for Signature generation and Signature verification Passed

16 DRBG selftest PASSED

KDF SSH selftest PASSED

KDF TLS selftest PASSED

SHA1 Self test PASSED

The implementation of self-tests by the TSF provides coverage of each TSF-relevant cryptographic function employed by each TOE component and covers the integrity of binaries critical to the operation of the TSF, along with other critical functions such as entropy noise source health testing. Therefore, the tests are sufficient to demonstrate that the TSF is operating correctly.

In the event of a failed self-test, the TSF will shut down its interfaces and prevent traffic from flowing through the TOE.

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

Section “Self-Test Errors” in the **Admin Guide** outlines possible self-test errors that could take place and what steps to follow to attempt to fix them. It states that Versa products perform a suite of FIPS 140-2 self-tests during power-up and re-boot. If any of the self-test fails, the product does not enter operational state and the serial console CLI and system logs display an error message indicating a self-test failure. If this occurs, you must reboot the device. If the product still does not enter operational state, contact Versa Support.

The following errors that can occur during self-tests:

• Failure to verify integrity of Versa application binaries.

• Failure to satisfy cryptographic self-test condition (for example, known-answer test failure).

• Failure of entropy source.

• Failure to instantiate a system service.

• Hardware or virtualization platform related errors.

Component Testing Assurance Activities: None Defined

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

2.9.3.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined **Guidance Assurance Activities**: None Defined **Testing Assurance Activities**: None Defined
Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

Section 7.1.3 of the ST states the list of all relevant keys (including the origin and storage of each), are described in section 7.3 of the ST. All keys are stored either in non-persistent RAM or persistent Flash. No interfaces exist which enable inspection of plaintext key components. All persistent keys are stored in the Linux filesystem and are restricted via adequate file permissions.

Section 7.3 of the ST provides a table that identifies all relevant keys and describes type, how they are generated, where they are stored, and how they are zeroized.

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

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

2.9.4.1 NDcPP22e:FPT_STM_EXT.1.1

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

Testing Assurance Activities: None Defined

2.9.4.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time, and that it provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

If 'obtain time from the underlying virtualization system' is selected, the evaluator shall examine the TSS to ensure that it identifies the VS interface the TOE uses to obtain time. If there is a delay between updates to the time on the VS and updating the time on the TOE, the TSS shall identify the maximum possible delay.

Section 7.1.1 of the ST states the TOE is configured to provide a source of date and time information used in audit event timestamps, certificate validity checking, time-based rekeying, session timeouts, user account lockout periods, time-of-day access, etc. The default time zone of the TOE components is UTC. The TOE must be set to receive clock updates from an NTP server which is secured via IPsec.

Obtain time from the underlying virtualization system is not selected.

Component Guidance Assurance Activities: The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the guidance documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

If the TOE supports obtaining time from the underlying VS, the evaluator shall verify the Guidance Documentation specifies any configuration steps necessary. If no configuration is necessary, no statement is necessary in the Guidance Documentation. If there is a delay between updates to the time on the VS and updating the time on the TOE, the evaluator shall ensure the Guidance Documentation informs the administrator of the maximum possible delay.

Section “Configure NTP Servers” in the **Admin Guide** provides instructions on how the admin can set the system’s time.

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

a) Test 1: If the TOE supports direct setting of the time by the Security Administrator then the evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

b) Test 2: If the TOE supports the use of an NTP server; the evaluator shall use the guidance documentation to configure the NTP client on the TOE, and set up a communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

c) Test 3: [conditional] If the TOE obtains time from the underlying VS, the evaluator shall record the time on the TOE, modify the time on the underlying VS, and verify the modified time is reflected by the TOE. If there is a delay between the setting the time on the VS and when the time is reflected on the TOE, the evaluator shall ensure this delay is consistent with the TSS and Guidance.

Test 1: The evaluator manually changed the time and observed that the TOE's time changed.

Test 2: The evaluator set up a local NTP server, then configured the TOE to utilize the NTP server, and observed that the TOE's time changed.

Test 3: Not applicable. The TOE does not obtain time from an underlying VS.

2.9.5 TSF TESTING (NDCPP22E:FPT_TST_EXT.1)

2.9.5.1 NDCPP22E:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

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

Section 7.1.6 of the ST states The TSF runs a suite of self-tests during initial start-up and periodically during normal operation to verify its correct operation. The complete list of self-tests implemented by each TOE component are described in section 6.1.9.5. Power-on self-tests are automatically invoked by start-up scripts. Periodic self- tests are run automatically by the daemons/commands, as applicable.

The following binaries are critical to the TSF and are verified during POST using RSA 2048 SHA-256 signatures against an image verification public key preinstalled on the TOE:

- Versa-dnsd
- Versa-ntpd
- Versa-vsmd
- Versa-vmod
- Versa-dhcpd
- Versa-certd
- Versa-rtd
- Versa-l2cd
- Libcrypto.so.1.1
- Libssl.so.1.1.
- Libfipstest.so

The following output to versa-service.log demonstrates a successful verification of the cryptographic self-tests:

AES Selftest PASSED

AES-CCM Encrypt selftest PASSED

AES-CCM Decrypt selftest PASSED

AES-GCM Encrypt Selftest PASSED

AES-GCM Decrypt selftest PASSED

AES-XTS-128 selftest PASSED

AES-XTS-256 selftest PASSED

AES-CMAC selftest PASSED

HMAC SHA1

HMAC SHA224

HMAC SHA256

HMAC SHA384

HMAC SHA512

HMAC Self test PASSED

Generated RSA Sign and Verified. RSA Selftest PASSED

ECDH Self test Passed

ECDSA selftest for Signature generation and Signature verification Passed

16 DRBG selftest PASSED

KDF SSH selftest PASSED

KDF TLS selftest PASSED

SHA1 Self test PASSED

The implementation of self-tests by the TSF provides coverage of each TSF-relevant cryptographic function employed by each TOE component and covers the integrity of binaries critical to the operation of the TSF, along with other critical functions such as entropy noise source health testing. Therefore, the tests are sufficient to demonstrate that the TSF is operating correctly.

In the event of a failed self-test, the TSF will shut down its interfaces and prevent traffic from flowing through the TOE.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

For distributed TOEs the evaluator shall ensure that the guidance documentation describes how to determine from an error message returned which TOE component has failed the self-test.

Section “Self-Test Errors” in the **Admin Guide** outlines possible self-test errors that could take place and what steps to follow to attempt to fix them. It states that Versa products perform a suite of FIPS 140-2 self-tests during power-up and re-boot. If any of the self-test fails, the product does not enter operational state and the serial console CLI and system logs display an error message indicating a self-test failure. If this occurs, you must reboot the device. If the product still does not enter operational state, contact Versa Support.

The following errors that can occur during self-tests:

• Failure to verify integrity of Versa application binaries.

• Failure to satisfy cryptographic self-test condition (for example, known-answer test failure).

• Failure of entropy source.

• Failure to instantiate a system service.

• Hardware or virtualization platform related errors.

Component Testing Assurance Activities: It is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE

b) Verification of the correct operation of the cryptographic functions necessary to fulfill any of the SFRs.

Although formal compliance is not mandated, the self-tests performed should aim for a level of confidence comparable to:

a) FIPS 140-2, chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

b) FIPS 140-2, chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions. Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

The evaluator shall either verify that the self tests described above are carried out during initial start-up or that the developer has justified any deviation from this.

For distributed TOEs the evaluator shall perform testing of self-tests on all TOE components according to the description in the TSS about which self-test are performed by which component.

The evaluator initiated a reboot from each device and confirmed that the status messages show that the FIPS self-tests were executed successfully.

2.9.6 Self-Test with Defined Methods (VPNGW13:FPT_TST_EXT.3)

2.9.6.1 VPNGW13:FPT_TST_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.6.2 VPNGW13:FPT_TST_EXT.3.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator verifies that the TSS describes the method used to perform self-testing on the TSF executable code, and that this method is consistent with what is described in the SFR.

Section 7.1.6 of the ST states The TSF runs a suite of self-tests during initial start-up and periodically during normal operation to verify its correct operation. The complete list of self-tests implemented by each TOE component are described in section 6.1.9.5. Power-on self-tests are automatically invoked by start-up scripts. Periodic self- tests are run automatically by the daemons/commands, as applicable.

The following binaries are critical to the TSF and are verified during POST using RSA 2048 SHA-256 signatures against an image verification public key preinstalled on the TOE:

- Versa-dnsd
- Versa-ntpd
- Versa-vsmd
- Versa-vmod
- Versa-dhcpd
- Versa-certd
- Versa-rtd
- Versa-l2cd
- Libcrypto.so.1.1
- Libssl.so.1.1.
- Libfipstest.so

The following output to versa-service.log demonstrates a successful verification of the cryptographic self-tests:

AES Selftest PASSED

AES-CCM Encrypt selftest PASSED

AES-CCM Decrypt selftest PASSED

AES-GCM Encrypt Selftest PASSED

AES-GCM Decrypt selftest PASSED

AES-XTS-128 selftest PASSED

AES-XTS-256 selftest PASSED

AES-CMAC selftest PASSED

HMAC SHA1

HMAC SHA224

HMAC SHA256

HMAC SHA384

HMAC SHA512

HMAC Self test PASSED

Generated RSA Sign and Verified. RSA Selftest PASSED

ECDH Self test Passed

ECDSA selftest for Signature generation and Signature verification Passed

16 DRBG selftest PASSED

KDF SSH selftest PASSED

KDF TLS selftest PASSED

SHA1 Self test PASSED

The implementation of self-tests by the TSF provides coverage of each TSF-relevant cryptographic function employed by each TOE component and covers the integrity of binaries critical to the operation of the TSF, along

with other critical functions such as entropy noise source health testing. Therefore, the tests are sufficient to demonstrate that the TSF is operating correctly.

In the event of a failed self-test, the TSF will shut down its interfaces and prevent traffic from flowing through the TOE.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.9.7 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.9.7.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.7.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.7.3 NDcPP22e:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describe how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the TSS needs to describe how and when the inactive version becomes active. The evaluator shall verify this description.

The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system firmware and software (for simplicity the term 'software' will be used in the following although the requirements apply to firmware and software). The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

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

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

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

Section 7.1.10 of the ST states that each TOE component has a specific version that can be queried by an administrator. When updates are made available by Versa, an administrator can obtain and install those updates from downloads.versa.com.

Cryptographic checksums (i.e., digital signatures) are used to verify software update files (to ensure they have not been modified from the originals distributed by Versa) before they are used to update the applicable TOE components.

Versa packages are signed with RSA-SHA256. At the package update or package installation time, the preinstalled RSA public key is used to do signature verification. If the signature is not matched, the package upgrade will fail.

The Security Administrator may also verify the integrity of the installation images manually by calculating a hash of the image and comparing it against the SHA256 checksums published along with the update packages.

The currently running TOE version may be queried by running the show system package-info command or by viewing the ‘About Versa Director’ page in the GUI. The Branch and Controller devices may be queried by clicking on the ‘Appliances’ view.

Each TOE component may be upgraded manually using the Director CLI by the Security Administrator. Each TOE component is updated individually.

The automatic checking of updates is not claimed.

Component Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the guidance documentation needs to describe how to query the loaded but inactive version.

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

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

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

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

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

Sections “Verify Software Version” and “Upgrade Director and VOS” in the **Admin Guide** explain that security updates are signed by the vendor using Versa Security Keys and verified during the update installation. This section also states that if the installation fails, an error with details is logged and the update process is terminated. This section also provides instructions to determine the currently running version on each TOE component.

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

a) Test 1: The evaluator performs the version verification activity to determine the current version of the product. If a trusted update can be installed on the TOE with a delayed activation, the evaluator shall also query the most recently installed version (for this test the TOE shall be in a state where these two versions match). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

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

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

2) An image that has not been signed

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

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

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

verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

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

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

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

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

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

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

Test 1: The branches are updated via the Director. First, the evaluator checked the currently installed version of the branch through the Director's Web UI, then updated the branch using the Director. The evaluator then checked the new version and verified that the installation was successful.

Test 2: For each device, the evaluator used legitimate updates that were modified in three ways:

1. A few bytes in the update file is modified via a hex editor
2. Update is missing a signature
3. Update's signature is corrupted

All TOE components, including the branches are updated via the Director. The update image is verified at the time of upload to the Director component. Attempts to update with each of these modified update files failed.

Test 3: While the Security Administrator may verify the integrity of the installation images manually by calculating a hash of the image and comparing it against the SHA256 checksums published along with the update packages, the TOE itself does not verify the hash value and therefore this test is omitted.

2.10 TOE ACCESS (FTA)

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

2.10.1.1 NDCPP22E:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the administrative remote session termination and the related inactivity time period.

Section 7.1.11 of the ST states an administrator can configure maximum inactivity times for both local and remote user sessions. When a session is inactive (i.e., no session input) for a configured period of time, the TOE will terminate the session, requiring the administrator to log in again to establish a new session when needed. The default is set to 15 minutes. The CLI interface must be configured and can support values between 15 and 43200 seconds.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

Section “Set Director GUI and CLI Idle Timeout” in the **Admin Guide** explains that the administrator can configure the number of minutes of inactivity after which the Director logs the user out and terminates the session.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

For each method of remote administration on the TOE, the evaluator configured two different idle timeout values. The evaluator then logged into the device and observed that after the configured amount of time, the TOE terminated the evaluator's session.

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

2.10.2.1 NDcPP22e:FTA_SSL.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how the local and remote administrative sessions are terminated.

Section 7.1.11 of the ST states administrators may terminate their own session by selecting the ‘Logout’ operation from the GUI or by typing ‘exit’ or ‘logout’ at the CLI.

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

Section “Log Out Manually from an Administrator Session” in the **Admin Guide** explains how an administrator can log out of an active session using the logout or exit commands.

Component Testing Assurance Activities: For each method of remote administration, the evaluator shall perform the following tests:

a) Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

b) Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

Test 1: This test was performed as part of FIA_UIA_EXT.1-t1 where the evaluator performed a logout for each interactive console session after a successful authentication attempt.

Test 2: This test was performed as part of FIA_UIA_EXT.1-t1 where the evaluator performed a logout for each remote interactive session after a successful authentication attempt.

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

2.10.3.1 NDcPP22e:FTA_SSL_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details whether local administrative session locking or termination is supported and the related inactivity time period settings.

Section 7.1.11 of the ST states an administrator can configure maximum inactivity times for both local and remote user sessions. When a session is inactive (i.e., no session input) for a configured period of time, the TOE will terminate the session, requiring the administrator to log in again to establish a new session when needed. The default is set to 15 minutes. The CLI interface must be configured and can support values between 15 and 43200 seconds.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation states whether local administrative session locking or termination is supported and instructions for configuring the inactivity time period.

Section “Set Director User Timeout and Authentication Lockout” in the **Admin Guide** contains instructions to configure the inactivity timeout period for console sessions. This section explains that on the expiry of the idle timeout, the administrator is logged out of sessions automatically.

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

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that reauthentication is needed when trying to unlock the session.

Test 1: The evaluator followed operational guidance to set the console timeout on the TOE to both 3 and 6 minutes and observed that the user was logged out automatically in 3 and 6 minutes, respectively.

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

2.10.4.1 NDcPP22e:FTA_TAB.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access, and might be configured during initial configuration (e.g. via configuration file).

Section 7.1.7 of the ST states that administrative access to the TOE is facilitated through the Director CLI (console or and management port via SSH), and through the Director GUI (management port via HTTPS/TLS).

Section 7.1.11 of the ST states the TOE has the functionality to present warning information to an administrator when attempting to login. There is no limit to the number of characters that can be entered in the login banner field through the GUI, but the banner is truncated to a 64K character limit for display.

The administrator may manually edit the /opt/versa/etc/banner.net file to present the login message to CLI users prior to authenticating.

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Section “Configure Login Banners” in the **Admin Guide** provides instructions to configure the login banner on all authentication interfaces (Web UI, SSH, and local console). It explains that the banner is displayed on all authentication interfaces (Web UI, SSH, and local console) once it is configured through the Director’s Web UI.

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

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

Test 1: The evaluator followed the steps outlined in the guidance to configure a login banner, then logged into the TOE via each possible login method to see the login banner updated and displayed properly.

2.11 Trusted path/channels (FTP)

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

2.11.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.11.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.11.1.3 NDcPP22e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Section 7.1.12 of the ST states that during startup of the Branch appliance, an IPsec tunnel is established with the SD-WAN Controller. All communication between Versa VOS (Branch appliance) and the Versa headend is sent via the IPsec tunnel.

The TOE supports establishing trusted channels between internal TOE components using the protocols described in the table below. All internal communications between the Versa headend and remote VOS Branch devices will be routed through this tunnel.

Source	Destination	Protocols / Data
VOS Branch	Headend - VOS Controller	IPsec / Distributed TOE internal communications Audit logs TSF data

The TOE supports establishing trusted channels between TOE components and external entities:

Component	Destination	Protocol / Format
All	NTP server	Time synchronization via NTP within IPsec.
All	Syslog server	Syslog forwarding to an external receiver within IPsec.
VOS Branch	Peer VPN Gateway	Site-to-site IPsec tunnels
VOS Branch	VPN Client	IPsec VPN client connections.

All IPsec endpoints are authenticated using X.509 certificates.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

As indicated by the ST, all communications between any TOE component and an authorized IT entity (Syslog server, NTP server) are protected via IPsec.

Section “Configure IPsec” in the **Admin Guide** describes how to configure an IPsec Tunnel profile on the TOE and use it to protect and route any traffic between the TOE component and the authorized IT entity. For example, protecting audit log records being transmitted to a remote auditing server or NTP traffic.

Component Testing Assurance Activities: The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each protocol with each authorized IT entity is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: For each protocol that the TOE can initiate as defined in the requirement, the evaluator shall follow the guidance documentation to ensure that in fact the communication channel can be initiated from the TOE.

c) Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

d) Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public- facing document or report.

Test 1: Refer to NDcPP22e:FCS_IPSEC_EXT.1 where the Syslog IPsec Channel was fully tested.

Test 2: Refer to NDcPP22e:FTP_ITC.1-t4 where the TOE initiating the trusted channel has been demonstrated.

Test 3: Refer to NDcPP22e:FTP_ITC.1-t4 where the TOE using an encrypted trusted channel has been demonstrated.

Test 4:

MAC Layer: The evaluator connected the TOE to the test peer/server and began to send some traffic between the devices. The connection was physically disrupted and after about 30-60 seconds the connection was restored. After the restoration, the evaluator observed the IPsec connection remained active and data remained protected.

APP Layer: The evaluator connected the TOE to the test peer/server and began to send some traffic between the devices. The connection was physically disrupted and after about 2-3 minutes the connection was restored. After the restoration, the evaluator observed that the IPsec connection had to be reestablished and data remained protected.

2.11.2 Inter-TSF Trusted Channel (VPN Communications) (VPNGW13:FTP_ITC.1/VPN)

2.11.2.1 VPNGW13:FTP_ITC.1.1/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.11.2.2 VPNGW13:FTP_ITC.1.2/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.11.2.3 VPNGW13:FTP_ITC.1.3/VPN

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

Refer to NDCPP22E:FTP_ITC.1.1

Component Guidance Assurance Activities: The EAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications.

Refer to NDCPP22E:FTP_ITC.1.1

Component Testing Assurance Activities: TheEAs specified for FTP_ITC.1 in the Supporting Document for the Base-PP shall be applied for IPsec VPN communications. Additional evaluation testing for IPsec is covered in FCS_IPSEC_EXT.1.

Refer to NDCPP22E:FTP_ITC.1.1

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

2.11.3.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.11.3.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.11.3.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Section 7.1.12 of the ST states remote administration of the TOE is performed via SSHv2 access to the Versa Director CLI and HTTPS/TLS access to the Director GUI.

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

See FIA_UIA_EXT.1 which documents the areas in the guide which provide instructions for establishing the remote administrative sessions for each supported method.

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

a) Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

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

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

Test 1 and test 2: The different remote administration methods were tested throughout the course of the evaluation. The successful testing of these channels and the demonstration of their encryption can be found in FCS_SSHS_EXT.1 for SSH and FCS_TLSS_EXT.1 for HTTPS on the Director.

2.12 Intrusion Prevention (IPS)

2.12.1 Anomaly-Based IPS Functionality (IPS10:IPS_ABD_EXT.1)

2.12.1.1 IPS10:IPS_ABD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.12.1.2 IPS10:IPS_ABD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.12.1.3 IPS10:IPS_ABD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describes the composition, construction, and application of baselines or anomaly-based attributes specified in IPS_ABD_EXT.1.1.

The evaluator shall verify that the TSS provides a description of how baselines are defined and implemented by the TOE, or a description of how anomaly-based rules are defined and configured by the administrator.

If 'frequency' is selected in IPS_ABD_EXT.1.1, the TSS shall include an explanation of how frequencies can be defined on the TOE.

If 'thresholds' is selected in IPS_ABD_EXT.1.1, the TSS shall include an explanation of how the thresholds can be defined on the TOE.

The evaluator shall verify that each baseline or anomaly-based rule can be associated with a reaction specified in IPS_ABD_EXT.1.3.

The evaluator shall verify that the TSS identifies all interface types capable of applying baseline or anomaly-based rules and explains how they are associated with distinct network interfaces. Where interfaces can be grouped into

a common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used) they can be treated collectively as a distinct network interface.

Section 7.1.0 of the ST states the TSF also supports anomaly detection, which monitors a network for unusual events or trends. Vulnerability profiles may be configured to compare an observed event with the baseline of the normal traffic. Anomaly detection detects patterns that are normally not present in the traffic, so it is useful for detecting new attacks.

The following actions may be configured in a vulnerability profile which are taken on matching anomaly rules:

- Allow
- Alert
- Drop packet
- Drop session
- Reject
- Reset client
- Reset server

Throughput rates are configured within a DoS protection profile. Time of day settings are configured via Schedule Objects which are applied to NGFW security policies. Frequency options are set under Thresholds within a vulnerability profile. Thresholds may be defined for each Security Scanner, or protocol parser.

Rules may be associated with distinct network interfaces, or groups of interfaces, referred to as zones. A zone profile defines flood protection, scan protection, and traffic anomaly protection information, and it is applied to all traffic flows that enter the zone through the interfaces associated with the zone.

Component Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides instructions to manually create baselines or anomaly-based rules according to the selections made in IPS_ABD_EXT.1.1. Note that dynamic 'profiling' of a network to establish a baseline is outside the scope of the PP-Module.

The evaluator shall verify that the operational guidance provides instructions to associate reactions specified in IPS_ABD_EXT.1.3 with baselines or anomaly-based rules.

The evaluator shall verify that the operational guidance provides instructions to associate the different policies with distinct network interfaces.

Section “Configure Stateful and Next-Generation Firewall” in the **Admin Guide** states that traffic policies are defined in terms of the static routes which in turn are associated with branch interfaces. So, a policy may be

defined to allow traffic from “WAN” to “LAN”. No branch interface will forward traffic until policies have been configured and applied to that interface. Traffic will not be forwarded unless it’s explicitly permitted by at least one policy rule, thus an implicit “deny-all” rule is applied to all interfaces to which any traffic filtering rule has been applied. The implicit deny-all rule is executed after all admin-defined rules have been executed, and will result in dropping all traffic that has not been explicitly permitted, or explicitly denied. If an administrator wants to log all denied traffic, a rule entry should be added that denies all traffic and logs it, e.g. by either adding a rule at the end of a policy to explicitly drop and log all traffic, or by setting the Default Action for the policy to block all traffic, and enabling logging for the default rule.

Additionally, the same section in the **Admin Guide** shows that for each rule, the administrator can specify a rule action, to allow, deny, reject or apply security profile to matching traffic with an intrusion policy.

It also outlines each protocol type, the required header field inspection and the specific keywords that can be used to configure a rule. For IPv4 and IPv6 the required header fields include the source and destination address and transport protocol. For ICMPv4 and ICMPv6 it includes the type and code, and for TCP and UDP it includes the source and destination port.

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

Test 1: The evaluator shall use the instructions in the operational guidance to configure baselines or anomaly-based rules for each attributes specified in IPS_ABD_EXT.1.1. The evaluator shall send traffic that does not match the baseline or matches the anomaly-based rule and verify the TOE applies the configured reaction. This shall be performed for each attribute in IPS_ABD_EXT.1.1.

Test 2: The evaluator shall repeat the test above to ensure that baselines or anomaly- based rules can be defined for each distinct network interface type supported by the TOE.

Test 1: The preprocessor detection rules for anomaly detected in headers and protocols can be seen where noted throughout IPS_SBD_EXT.1.1. For threshold (ie. frequency), the evaluator configured the TOE with an anomaly-based rule (and policy configuration) which utilized a combination of fields that were otherwise tested in IPS_SBD_EXT.1 (e.g. TCP destination port, IP destination address, and FTP commands). The evaluator then attempted to send traffic where the number of packets exceeds the configured maximum frequency for the anomalous packets and verified that the traffic exceeding the configured anomaly-based rule was detected and the TOE generated an intrusion event and dropped the traffic.

Test 2: The TOE supports only the Ethernet based networking that was tested during IPS_SBD_EXT.1.1-t1.

2.12.2 IP Blocking (IPS10:IPS_IPB_EXT.1)

2.12.2.1 IPS10:IPS_IPB_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.12.2.2 IPS10:IPS_IPB_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

Section 7.1.14 of the ST states IP address filters are based on the following IP address attributes:

- IP reputation—Administrators can create IP-filtering profiles with the following predefined IP reputations:
  - BotNets
  - Denial of service
  - Phishing
  - Proxy
  - Reputation
  - Scanners
  - Spam sources
  - Web attacks
  - Windows exploits

- Geolocation—Versa Networks provides a list of predefined regions that you can use to create IP-filtering profiles based on geolocation.

Administrators define IP-filtering profiles to filter traffic based on the IP address attributes. Each IP-filtering profile object can specify the following:

- Allow lists for IP addresses
- Deny lists for IP addresses
- DNS reverse lookup configurations
- Rules for geolocation-based actions
- Rules for IP reputation–based actions

Administrators can configure rules to match the IP address based on the following match criteria:

- Destination IP address
- Source IP address
- Source and destination IP address
- Source or destination IP address

Administrators can enforce the following actions when a session's IP address matches the conditions in an IP-filtering profile:

- Allow
- Alert
- Drop packet
- Drop session
- Reset

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

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

Section “Configure Intrusion Detection and Prevention” in the **Admin Guide** describes how to configure whitelists and blacklists. It is noted that only the Administrator can configure these attributes which are part of access control.

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

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

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

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

Test 1: While configured in in-line mode, the evaluator attempted the following types of traffics with the corresponding results:

Variation
CONTROL: Attempt to send traffic using addresses not in a Known-Good or a Known-Bad list: Pass -- packet allowed as expected.
Attempt to send traffic using a SRC address blocked by a Known-Bad list rule specifying only a single SRC address: Pass -- packet not allowed as expected.
Attempt to send traffic using a DST address blocked by a Known-Bad list rule specifying only a single DST address: Pass -- packet not allowed as expected.
Attempt to send traffic using a DST address blocked by a Known-Bad list rule specifying multiple SRC addresses: Pass -- packet not allowed as expected.
Attempt to send traffic using a SRC address blocked by a Known-Bad list rule specifying multiple DST addresses: Pass -- packet not allowed as expected.

Test 2: Because the TOE defaults to passing packets not otherwise blocked, the only way to “send traffic that would otherwise be denied by the TOE” is to create overlapping Good and Bad list rules. Since that is the focus of test 3 below, no additional tests were performed here. The tests which demonstrate that “Good List” entries are allowed to pass through the TOE are parts 2, 3, 4, 6, 8, 9, 10, 11, 13, and 15 of IPS_IPB_EXT.1-t3.

Test 3: For this product, the policy hierarchy order is not configurable. The following table organizes the results from the individual test variations, which comprise this test case. Details can be found in the log file and packet captures for each variation shown in the following table.

Variation
1. CONTROL: Attempt to send traffic using addresses not in a Known-Good or a Know-Bad list: Pass -- packet allowed as expected.
2. Attempt to send traffic using a SRC address that is in a Known-Good list rule and in a Known-Bad list rule both specifying a single SRC address: Pass -- packet allowed as expected.
3. Attempt to send traffic using a DST address that is in a Known-Good list rule and in a Known-Bad list rule both specifying a single DST address: Pass -- packet allowed as expected.
4. Attempt to send traffic using a SRC address in a Known-Good list rule and in a multiple address Known-Bad list (e.g., list or range): Pass -- packet allowed as expected.
5. Attempt to send traffic using a SRC address contained by a multiple address Known-Bad list (e.g., list or range) that is overlapped by a Known-Good list rule, where the SRC address is not part of the Known-Good list: Pass -- packet not allowed as expected.
6. Attempt to send traffic using a DST address in a Known-Good list rule and in a multiple address Known-Bad list (e.g., list or range): Pass -- packet allowed as expected.
7. Attempt to send traffic using a DST address contained by a multiple address Known-Bad list (e.g., list or range) that is overlapped by a Known-Good list rule, where the DST address is not part of the Known-Good list: Pass -- packet not allowed as expected.
8. Attempt to send traffic using a SRC address in a Known-Good list rule with multiple addresses and a single overlapping Known-Bad list address: Pass -- packet allowed as expected.
9. Attempt to send traffic using a SRC address matching a Known-Bad list rule with a single address overlapped by a Known-Good list rule with multiple addresses: Pass -- packet allowed as expected.
10. Attempt to send traffic using a DST address in a Known-Good list rule with multiple addresses and a single overlapping Known-Bad list address: Pass -- packet allowed as expected.
11. Attempt to send traffic using a DST address matching a Known-Bad list rule with a single address overlapped by a Known-Good list rule with multiple addresses: Pass -- packet allowed as expected.
12. Attempt to send traffic using an SRC address in a Known-Bad list rule, with multiple addresses when the rule is overlapped by a Known-Good list rule with multiple addresses which does not include the SRC address:

Pass -- packet not allowed as expected.
13. Attempt to send traffic using an SRC address in a Known-Bad list rule, with multiple addresses when the rule is overlapped by a Known-Good list rule with multiple addresses which does include the SRC address: Pass -- packet allowed as expected.
14. Attempt to send traffic using an DST address in a Known-Bad list rule, with multiple addresses when the rule is overlapped by a Known-Good list rule with multiple addresses which does not include the DST address: Pass -- packet not allowed as expected.
15. Attempt to send traffic using an DST address in a Known-Bad list rule, with multiple addresses when the rule is overlapped by a Known-Good list rule with multiple addresses which does include the DST address: Pass -- packet allowed as expected.

2.12.3 Network Traffic Analysis (IPS10:IPS_NTA_EXT.1)

2.12.3.1 IPS10:IPS_NTA_EXT.1.1

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

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

Section 7.1.14 in the ST states that IP filtering and vulnerability policies are applied to NGFW security access policies. In the policy, administrators define the traffic to match based on various parameters, such as zones and applications, and configure the policy to enforce the action defined in vulnerability profile.

It is recommended to use the predefined vulnerability profiles, however administrators may create custom vulnerability profiles. Then, the administrator must associate the vulnerability profiles with a next-generation firewall (NGFW) security profile (also called an access policy profile) in an NGFW policy. An NGFW security profile comprises an ordered set of one or more policy rules. Each policy rule comprises a set of match criteria and enforcement actions.

As with firewall access policies, a rule is triggered when all match criteria defined in the rule matches the payload. All rules in the security access policy are evaluated starting with the first rule in the policy. The first rule that matches is selected and the corresponding security actions are enforced. No other rules are evaluated once a match is found.

It is recommended that in a security policy to configure more specific rules first and then configure generic rules, followed by a final deny-all rule. The TOE does not prevent administrators from applying conflicting rules. Since IP

filtering, vulnerability, and override settings may all be associated with an access rule, it is recommended to create the rules in the order in which the desired filtering needs to occur.

Guidance Assurance Activities: The evaluator shall verify that the guidance describes the default precedence.

If the precedence is configurable, the evaluator shall verify that the guidance explains how to configure the precedence.

Section "Configure Intrusion Detection and Prevention" in the **Admin Guide** describes how to configure whitelist, blacklist rules which are applied to traffic before it is inspected by access control rules and intrusion policies. This section states that the policy hierarchy order is not configurable and follows this order: Security Intelligence (whitelist takes precedence over blacklist), anomaly-based rules, then signature-based rules.

Other sections in the **Admin Guide**, further describes the default precedence as follows:

Section "Configure Stateful and Next-Generation Firewall" in the **Admin Guide** states that VOS processes rules in a top-down order, meaning the top-most applies first, followed by the seconds, until it finds a match. The matching rule action is applied on the network traffic. It is recommended that you order rules from permissive to restrictive to prevent blocking of legitimate traffic. You can add a deny rule to the bottom to handle traffic that does not match any other rules.

Section "Configure Intrusion Detection and Prevention" in the **Admin Guide** provides instructions for creating an Access Control policy. The Intrusion and Network Analysis policies are associated with the Access Control policy which is then assigned to one or more sensors.

Section "Custom IPS" in the **Admin Guide** states that Intrusion policies are invoked by your access control policy and are the system's last line of defense before traffic is allowed to its destination. It states that IP filtering and vulnerability policies are applied to NGFW security access policies. In the policy, administrators define the traffic to match based on various parameters, such as zones and applications, and configure the policy to enforce the action defined in vulnerability profile.

Testing Assurance Activities: None Defined

2.12.3.2 IPS10:IPS_NTA_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS indicates that the following protocols are supported:

- IPv4

- IPv6

- ICMPv4

- ICMPv6

- TCP

- UDP

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

Section 7.1.14 of the ST states the following protocols are supported for NGFW and IDP inspection policies:

- IPv4
- IPv6
- ICMPv4
- ICMPv6
- TCP
- UDP

It also states that Versa uses industry-standard network traffic generators to perform interoperability testing to ensure compliance with the above protocols.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.12.3.3 IPS10:IPS_NTA_EXT.1.3

TSS Assurance Activities: The evaluator shall verify that the TSS identifies all interface types capable of being deployed in the modes of promiscuous, and or inline mode as well as the interfaces necessary to facilitate each deployment mode (at a minimum, the interfaces need to support inline mode). The evaluator shall also check that the TSS provides a description for how the management interface is logically distinct from any sensor interfaces.

Section 7.1.14 of the ST states Management Ethernet interfaces are logically separated from the Ethernet data ports and cannot be used as a sensor interface. The TSF supports both in-line and promiscuous modes on any available Ethernet data port. A minimum of two Ethernet data ports is required for in-line mode.

Rules may be associated with distinct network interfaces, or groups of interfaces, referred to as zones.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides instructions on how to deploy each of the deployment methods outlined in the TSS. The evaluator shall also verify that the operational guidance provides instructions of applying IPS policies to interfaces for each deployment mode. If the management interface is configurable, the evaluator shall verify that the operational guidance explains how to configure the interface as a management interface.

The evaluator shall verify that the operational guidance explains how the TOE sends commands to remote traffic filtering devices if this functionality is supported.

Section "Custom IPS" in the **Admin Guide** states that signature-based detection applies a set of pre-defined rules or custom rules. Rules are based on the snort rule format. Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the **rule's action**, protocol, source and destination IP addresses and netmasks, and the source and destination ports information, and a series of customizable rule options which cover all the fields described in IPS_SBD_EXT.1.5. It also lists the different actions that can be configured and depending on which action is configured, the TSF will behave accordingly.

Testing Assurance Activities: Testing for this element is performed in conjunction with testing where promiscuous and inline interfaces are tested.

The tests associated with this requirement have been completed in subsequent assurance activities in which promiscuous and inline interfaces are tested (e.g. tests for IPS_SBD_EXT.1) and in the requirement of FTP_ITC.1 (if the ST author selects other interface types) and/or FTP_TRP.1 (for interfaces in management mode) in the base PP.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.12.4 Signature-Based IPS Functionality - per TD0722 (IPS10:IPS_SBD_EXT.1)

2.12.4.1 IPS10:IPS_SBD_EXT.1.1

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

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

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

common interface type (e.g., where the same internal logical path is used, perhaps where a common device driver is used) they can be treated collectively as a distinct network interface.

Section 7.1.14 states the following protocols are supported for NGFW and IDP inspection policies:

- IPv4
- IPv6
- ICMPv4
- ICMPv6
- TCP
- UDP

Management Ethernet interfaces are logically separated from the Ethernet data ports and cannot be used as a sensor interface. The TSF supports both in-line and promiscuous modes on any available Ethernet data port. A minimum of two Ethernet data ports is required for in-line mode.

Rules may be associated with distinct network interfaces, or groups of interfaces, referred to as zones. A zone profile defines flood protection, scan protection, and traffic anomaly protection information, and it is applied to all traffic flows that enter the zone through the interfaces associated with the zone.

The zone protection profile can detect and prevent the following types of traffic from entering the networks in the zone:

- Traffic floods of various protocols, such as TCP, UDP, and ICMP
- Port scans, host sweeps, and other types of reconnaissance traffic
- Malicious or spoofed packets

Signature based detection applies a set of pre-defined rules or custom rules. Rules are based on the snort rule format. Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information, and a series of customizable rule options which cover all the fields described in IPS_SBD_EXT.1.5.

The TSF also supports anomaly detection, which monitors a network for unusual events or trends. Vulnerability profiles may be configured to compare an observed event with the baseline of the normal traffic. Anomaly detection detects patterns that are normally not present in the traffic, so it is useful for detecting new attacks.

The following actions may be configured in a vulnerability profile which are taken on matching anomaly rules:

- Allow
- Alert
- Drop packet
- Drop session
- Reject
- Reset client
- Reset server

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

- IPv4: version; header length; packet length; ID; IP flags; fragment offset; time to live (TTL); protocol; header checksum; source address; destination address; IP options; and, if selected, type of service (ToS).

- IPv6: Version; payload length; next header; hop limit; source address; destination address; routing header; and, if selected, traffic class and/or flow label.

- ICMP: type; code; header checksum; and, if selected, other header fields (varies based on the ICMP type and code).

- ICMPv6: type; code; and header checksum.

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

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

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

(TD0722 applied)

Section “Custom IPS” in the **Admin Guide** specifies all the supported protocols and filtering options available to an administrator. This includes all of the protocols and fields specified in the assurance activity. It also provides the instructions for how to select and configure the rule’s action (allow, alert, drop packet, etc.).

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

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet header signatures can be created and/or configured with the selected and/or configured reactions specified in IPS_SBD_EXT.1.5 for each of the attributes listed below. Each attribute shall be individually assigned to its own unique signature:

- IPv4: Version; Header Length; Packet Length; ID; IP Flags; Fragment Offset; Time to Live (TTL); Protocol; Header Checksum; Source Address; Destination Address; IP Options; and, if selected, type of service (ToS).

- IPv6: Version; payload length; next header; hop limit; source address; destination address; routing header; and, if selected, traffic class and/or flow label.

- ICMP: Type; Code; Header Checksum; and, if selected, other Header fields (varies based on the ICMP type and code).

- ICMPv6: Type; Code; and Header Checksum.

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

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

The evaluator shall generate traffic to trigger a signature and shall then use a packet sniffer to capture traffic that ensures the reactions of each rule are performed as expected.

Test 2: The evaluator shall repeat the test above to ensure that signature-based IPS policies can be defined for each distinct network interface type capable of applying signatures as supported by the TOE.

(TD0722 applied)

Test 1: The evaluator created new local rules and identified existing rules that inspected each field in the protocols as listed in the test above. The evaluator then set up a packet capture to capture traffic from each rule and verified that the appropriate signatures are present and that the reactions of each rule are performed as expected.

Test 2: The TOE supports only the Ethernet based networking that was tested during IPS_SBD_EXT.1.1-t1.

2.12.4.2 IPS10:IPS_SBD_EXT.1.2

TSS Assurance Activities: The evaluator shall verify that the TSS describes what is comprised within a string-based detection signature.

The evaluator shall verify that each packet payload string-based detection signature can be associated with a reaction specified in IPS_SBD_EXT.1.5.

Section 7.1.14 in the ST states that signature-based detection applies a set of pre-defined rules or custom rules. Rules are based on the snort rule format. Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and

netmasks, and the source and destination ports information, and a series of customizable rule options which cover all the fields described in IPS_SBD_EXT.1.5.

Using the «content» keyword allows for a string in quotes to be used for detection. Contents match on bytes. There are 256 different values of a byte (0-255). You can match on all characters; from a to z, upper case and lower case, and special characters. Not all of the bytes are printable characters, and others have sig. For these bytes, hexadecimal notations are used.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides instructions with how to configure rules using the packet payload string-based detection fields defined in IPS_SBD_EXT.1.2.

The evaluator shall verify that the operational guidance provides instructions with how to configure reactions specified in IPS_SBD_EXT.1.5 for each string-based detection signature.

The evaluator shall verify that the operational guidance provides instructions with how rules are associated with distinct network interfaces that are capable of being associated with signatures.

Section “Custom IPS” in the **Admin Guide** outlines the ability for the administrator to specify patterns that a payload within a packet must match to trigger the rule. The rule can then be configured to drop and log, allow, or log the packet. It also provides the instructions for how to select and configure the rule’s action (allow, alert, drop packet, etc.).

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

Test 1: The evaluator shall use the instructions in the operational guidance to test that packet payload string-based detection rules can be assigned to the reactions specified in IPS_SBD_EXT.1.5 using the attributes specified in IPS_SBD_EXT.1.2. However it is not required (nor is it feasible) to test all possible strings of protocol data, the evaluator shall ensure that a selection of strings in the requirement is selected to be tested. At a minimum at least one string using each of the following attributes from IPS_SBD_EXT.1.2 should be tested for each protocol. The evaluator shall generate packets that match the string in the rule and observe the corresponding reaction is as configured.

- Test at least one string of characters for ICMPv4 data: beyond the first 4 bytes of the ICMP header.

- Test at least one string of characters for ICMPv6 data: beyond the first 4 bytes of the ICMP header.

- TCP data (characters beyond the 20 byte TCP header):

i) Test at least one FTP (file transfer) command: help, noop, stat, syst, user, abort, acct, allo, appe, cdup, cwd, dele, list, mkd, mode, nlst, pass, pasv, port, pass, quit, rein, rest, retr, rmd, rnfr, rnto, site, smnt, stor, stou, stru, and type.

ii) HTTP (web) commands and content:

(1) Test both GET and POST commands

(2) Test at least one administrator-defined strings to match URLs/URIs, and web page content.

iii) Test at least one SMTP (email) state: start state, SMTP commands state, mail header state, mail body state, abort state.

iv) Test at least one string in any additional attribute type defined within the 'other types of TCP payload inspection' assignment, if any other types are specified.

- Test at least one string of UDP data: characters beyond the first 8 bytes of the UDP header;

- Test at least one string for each additional attribute type defined in the 'other types of packet payload inspection' assignment, if any other types are specified.

Test 2: The evaluator shall repeat Test 1 above to ensure that signature-based IPS policies can be defined for each distinct network interface type capable of applying signatures as supported by the TOE.

Test 1: The evaluator created new local rules and identified existing rules that inspected each string pattern as listed in the test above. The TOE was able to match the intrusion policy with each of those fields in each packet and act according to the policy. The evaluator repeated the tests with the TOE configured in promiscuous mode. The evaluator observed that all attacks generated IPS intrusion events regardless of inline or Promiscuous. The evaluator also observed that when configured for inline mode of operation, the TOE blocked packets.

Test 2: The TOE supports only the Ethernet based networking that was tested during IPS_SBD_EXT.1.1-t1.

2.12.4.3 IPS10:IPS_SBD_EXT.1.3

TSS Assurance Activities: The evaluator shall verify that the TSS describes how the attacks defined in IPS_SBD_EXT.1.3 are processed by the TOE and what reaction is triggered when these attacks are identified.

Section 7.1.14 of the ST states within a zone protection profile, the following options may be selected in order to drop packets involved in attacks as defined in IPS_SBD_EXT.1. These include:

- Fragmented IP packets
- Spoofed IP packets
- Fragmented ICMP packets
- Large ICMP packets
- Packets with improper TCP flags
- Malformed UDP packets

Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides instructions with configuring rules to identify the attacks defined in IPS_SBD_EXT.1.3 as well as the reactions to these attacks as specified in IPS_SBD_EXT.1.5.

Section “Configure Intrusion Detection and Prevention” in the **Admin Guide** states that it is recommended to use the Versa-Recommended vulnerability profile. It contains pre-configured rules that can be used to identify and prevent all of the attacks defined in IPS_SBD_EXT.1.3.

Testing Assurance Activities: The evaluator shall create and/or configure rules for each attack signature in IPS_SBD_EXT.1.3. For each attack, the TOE should apply its corresponding signature and enable it to each distinct network interface type capable of applying the signatures. The evaluator shall use packet captures to ensure that the attack traffic is detected by the TOE and a reaction specified in IPS_SBD_EXT.1.5 is triggered and stops the attack. Each attack should be performed one after another so as to ensure that its corresponding signature successfully identified and appropriately reacted to a particular attack.

Test 1: The evaluator created rules for each type of attack and verified via packet capture and syslog that each attack was detected and dropped or logged depending on which mode the TOE was configured in.

2.12.4.4 IPS10:IPS_SBD_EXT.1.4

TSS Assurance Activities: The evaluator shall verify that the TSS describes how the attacks defined in IPS_SBD_EXT.1.4 are processed by the TOE and what reaction is triggered when these attacks are identified.

Section 7.1.14 of the ST states that the zone protection profile can detect and prevent the following types of traffic from entering the networks in the zone:

- Traffic floods of various protocols, such as TCP, UDP, and ICMP
- Port scans, host sweeps, and other types of reconnaissance traffic
- Malicious or spoofed packets

DoS Flood thresholds may be defined for ICMP, IP, TCP, and UDP. For TCP, packets may be randomly dropped or SYN cookies may be used to ensure that valid connections are not dropped during a SYN flood attack. SYN cookies are the default behavior.

Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides instructions with configuring rules to identify the attacks defined in IPS_SBD_EXT.1.4 as well as the reactions to these attacks as specified in IPS_SBD_EXT.1.5.

Section “Custom IPS” in the **Admin Guide** specifically outlines IP, TCP, UDP, and ICMP port scan detection methods and provides instructions for the administrator to configure these methods.

Section “Configure Intrusion Detection and Prevention” in the **Admin Guide** provides instructions for the administrator to configure DoS rules to limit excess activity by users. The administrator then has the option to configure how such detected events are handled (log, drop and log, allow).

Testing Assurance Activities: The evaluator shall configure individual signatures for each attack in IPS_SBD_EXT.1.4. For each attack, the TOE should apply its corresponding signature and enable it to each distinct network interface type capable of applying signatures. The evaluator shall use packet captures to ensure that the attack traffic is detected by the TOE and a reaction specified in IPS_SBD_EXT.1.5 is triggered and stops the attack. Each attack should be performed one after another so as to ensure that its corresponding signature successfully identified and appropriately reacted to a particular attack.

Test 1: The evaluator configured the ‘signature’ for the attack traffic identified in IPS_SBD_EXT.1.4, then followed the abstract procedures when the TOE was configured as an inline mode device. The evaluator repeated the tests with the TOE configured as a Promiscuous device. The evaluator observed that all attack traffic generated IPS intrusion events regardless of inline or Promiscuous. The evaluator also observed that when configured for inline mode of operation, the TOE blocked packets.

2.12.4.5 IPS10:IPS_SBD_EXT.1.5

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The guidance EAs for this element are performed in conjunction with IPS_SBD_EXT.1.1, IPS_SBD_EXT.1.3, and IPS_SBD_EXT.1.4.

The guidance EAs for this element are performed in conjunction with IPS_SBD_EXT.1.1, IPS_SBD_EXT.1.3, and IPS_SBD_EXT.1.4.

Testing Assurance Activities: The test EAs for this element are performed in conjunction with those for IPS_SBD_EXT.1.1, IPS_SBD_EXT.1.2, IPS_SBD_EXT.1.3, and IPS_SBD_EXT.1.4.

The test EAs for this element are performed in conjunction with those for IPS_SBD_EXT.1.1, IPS_SBD_EXT.1.2, IPS_SBD_EXT.1.3, and IPS_SBD_EXT.1.4.

2.12.4.6 IPS10:IPS_SBD_EXT.1.6

TSS Assurance Activities: None Defined

Guidance Assurance Activities: The evaluator shall verify that the operational guidance provides configuration instructions, if needed, to detect payload across multiple packets

Section “Custom IPS” in the **Admin Guide** outlines the ability for the administrator to specify patterns that a payload within a packet must match to trigger the rule. The rule can then be configured to drop and log, allow, or log the packet. It also provides the instructions for how to select and configure the rule’s action (allow, alert, drop packet, etc.).

Section “Configure Intrusion Detection and Prevention” in the **Admin Guide** describes the creation of IPS events and what options are available to a system administrator to monitor or react to rules being triggered. It also describes how to enable or disable a rule and specifies that when an event is enabled (either to log or log and drop), a log will be generated each time the policy is triggered. If an event is disabled, no log will be generated. Subsection “configure DoS protection settings” describes how to modify intrusion event thresholds including a count of events and timeframe of events.

Testing Assurance Activities: The evaluator shall repeat one of the tests in IPS_SBD_EXT.1.2 Test 1 but generate multiple non-fragmented packets that contain the string in the rule defined. The evaluator shall verify that the malicious traffic is still detected when split across multiple non-fragmented packets.

The evaluator repeated the UDP string matching test sending 40 packets instead of just one. The TOE detected and blocked all 40 packets.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

3. Protection Profile SAR Assurance Activities

The following sections address assurance activities specifically defined in the claimed Protection Profile that correspond with Security Assurance Requirements.

3.1 Development (ADV)

3.1.1 Basic Functional Specification (ADV_FSP.1)

Assurance Activities: The EAs for this assurance component focus on understanding the interfaces (e.g., application programing interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV_FSP.1-1, ADV_FSP.1-2, ADV_FSP.1-3, and ADV_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV_FSP.1.2D (work units ADV_FSP.1-4, ADV_FSP.1-6 and ADV_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent is that these interfaces will be adequately tested, and

having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

It should be noted that there may be some SFRs that do not have an interface that is explicitly 'mapped' to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

For this PP, the Evaluation Activities for this family focus on understanding the interfaces presented in the TSS in response to the functional requirements and the interfaces presented in the AGD documentation. No additional 'functional specification' documentation is necessary to satisfy the Evaluation Activities specified in the SD.

3.2 Guidance documents (AGD)

3.2.1 Operational User Guidance (AGD_OPE.1)

Assurance Activities: The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps (per TD0536):

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

In addition, the evaluator performs the EAs specified below.

The evaluator shall ensure the Operational guidance documentation is distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

In addition the evaluator shall ensure that the following requirements are also met.

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) The documentation must describe the process for verifying updates to the TOE by verifying a digital signature. The evaluator shall verify that this process includes the following steps:

1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes instructions that describe at least one method of validating the hash/digital signature.

c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

The **Admin Guide** provides instructions for configuring CC and FIPS mode for each TOE component such that only the cryptographic algorithms and parameters used for the evaluated configuration are available and that it is clear that no other cryptographic engines have been evaluated or tested. There are warnings and notes throughout the **Admin Guide** regarding use of functions that are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP22e:FPT_TUD_EXT.1.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

Preparative procedures are distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

In addition, the evaluator performs the EAs specified below.

The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

In addition the evaluator shall ensure that the following requirements are also met.

The preparative procedures must

a) include instructions to provide a protected administrative capability; and

b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

The evaluation team had the following documents to use when configuring the TOE:

- Configure for NIAP Common Criteria **(Admin Guide)**

In some instances, the document referenced general Versa manuals which the evaluation could find on the Versa web site. The completeness of the documentation is addressed by their use in the AA's carried out in the evaluation.

3.3 Life-cycle support (ALC)

3.3.1 Labelling of the TOE (ALC_CMC.1)

Assurance Activities: This component is targeted at identifying the TOE such that it can be distinguished from other products or versions from the same vendor and can be easily specified when being procured by an end user. A label could consist of a 'hard label' (e.g., stamped into the metal, paper label) or a 'soft label' (e.g., electronically presented when queried).

The evaluator performs the CEM work units associated with ALC_CMC.1.

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

The evaluator verified that the ST, TOE and Guidance are all labeled with the same hardware versions and software. The information is specific enough to procure the TOE and it includes hardware models and software versions. The evaluator checked the TOE software version and hardware identifiers during testing by examining the actual machines used for testing.

3.3.2 TOE CM Coverage (ALC_CMS.1)

Assurance Activities: Given the scope of the TOE and its associated evaluation evidence requirements, the evaluator performs the CEM work units associated with ALC_CMS.1.

When evaluating the developer's coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

See section 3.3.1 above for an explanation of how all CM items are addressed.

3.4 Tests (ATE)

3.4.1 Independent Testing - Conformance (ATE_IND.1)

Assurance Activities: Testing is performed to confirm the functionality described in the TSS as well as the guidance documentation (includes 'evaluated configuration' instructions). The focus of the testing is to confirm that the requirements specified in Section 5.1.7 are being met. The Evaluation Activities in [SD] identify the specific testing activities necessary to verify compliance with the SFRs. The evaluator produces a test report documenting the plan for and results of testing, as well as coverage arguments focused on the platform/TOE combinations that are claiming conformance to this cPP.

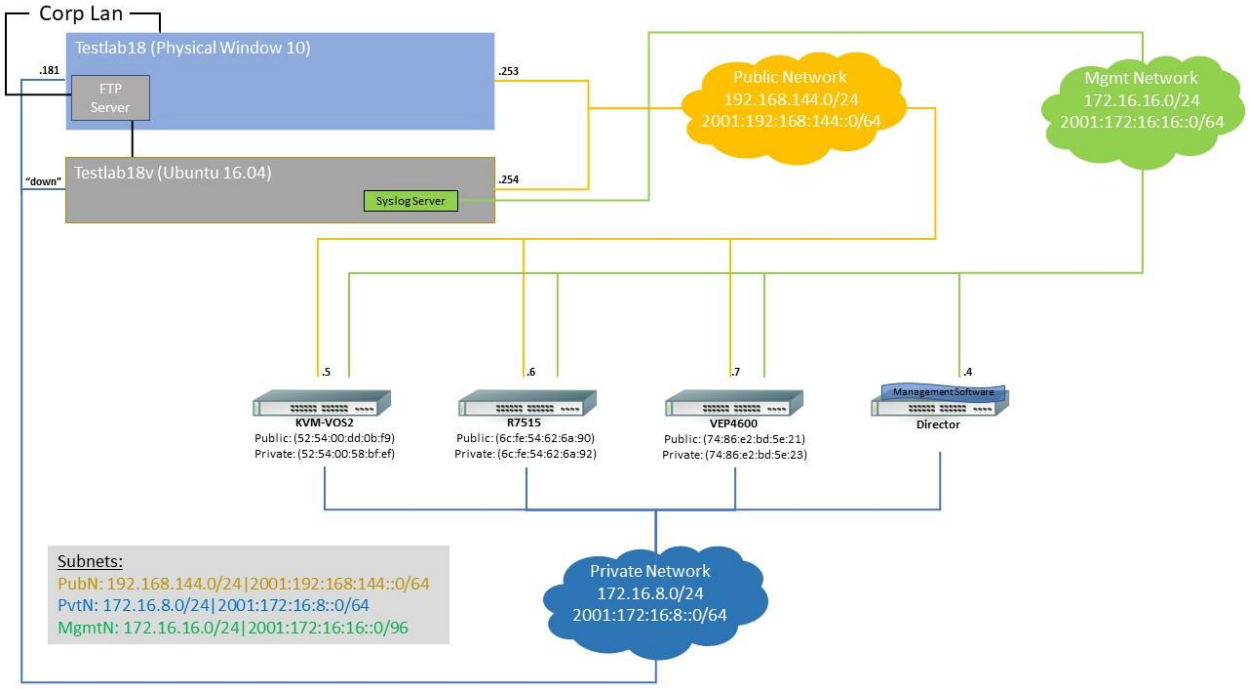
The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2, 3 and 4.

The evaluator should consult Appendix B when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section B.4.3.1.

The evaluator created a Detailed Test Report (DTR) to address all aspects of this requirement. The DTR discusses the test configuration, test cases, expected results, and test results. The test configuration consisted of the following TOE platforms along with supporting products.



TOE Platforms:

- Versa Headend (consists of Director, Analytics, and Controller VMs)
- Versa VOS VM on Ubuntu 18.04 with KVM
- Dell PowerEdge R7515
- Dell Virtual Edge Platform (VEP) 4600

Supporting Software:

- Windows 10 Pro
- Wireshark version 2.6.6
- Windows SSH Client – Putty version 0.68 (used to connect to device console and SSH)

The Gossamer Test Server with an Ubuntu environment acted as a platform to initiate testing. The test Server also acted as a syslog server.

- Openssh-client version 7.2p2
- Big Packet Putty version 6.2

- Nmap version 7.01
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Stunnel version 5.30
- OpenSSL version 1.0.2g
- Strongswan version 5.3.5
- Rsyslog version 8.16.0

3.5 Vulnerability assessment (AVA)

3.5.1 Vulnerability Survey (AVA_VAN.1)

Assurance Activities: While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities, and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis, and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A, while an 'outline' of the assurance activity is provided below.

In addition to the activities specified by the CEM in accordance with Table 2, the evaluator shall perform the following activities.

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

The developer shall provide documentation identifying the list of software and hardware components7 that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE) such as a web server and protocol or cryptographic libraries. This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

If the TOE is a distributed TOE then the developer shall provide:

a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]

b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.5.1.2 and 3.6.1.2.

The evaluator formulates hypotheses in accordance with process defined in Appendix A. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2. The results of the analysis shall be documented in the report according to Appendix A.3.

The vulnerability analysis is in the Detailed Test Report (DTR) prepared by the evaluator. The vulnerability analysis includes a public search for vulnerabilities and fuzz testing. None of the public search for vulnerabilities, or the fuzz testing uncovered any residual vulnerability.

The evaluation team performed a public search for vulnerabilities in order to ensure there are no publicly known and exploitable vulnerabilities in the TOE from the following sources:

- National Vulnerability Database (https://web.nvd.nist.gov/vuln/search)
- Vulnerability Notes Database (http://www.kb.cert.org/vuls/)
- Rapid7 Vulnerability Database (https://www.rapid7.com/db/vulnerabilities)
- Tipping Point Zero Day Initiative (http://www.zerodayinitiative.com/advisories)
- Exploit / Vulnerability Search Engine (http://www.exploitsearch.net)
- SecurITeam Exploit Search (http://www.securiteam.com)
- Tenable Network Security (http://nessus.org/plugins/index.php?view=search)
- Offensive Security Exploit Database (https://www.exploit-db.com/)

The search was performed on 03/28/2024 with the following search terms: "Versa", "Versa Networks", "Versa Director 22.1", "Versa Operating System 22.1", "VOS 22.1", "Versa SD-WAN Controller 22.1", "Versa SD-WAN Branch 22.1", "Versa Analytics 22.1", "Versa CSG", "Ubuntu 18.0.4.6 LTS", "Bouncy Castle FIPS Java API 1.0.2.3", "Bouncy Castle 1.0.2.3", "Rambus Quicksec 6.1", "Quicksec 6.1", "OpenSSL 1.1.1-1ubuntu2.1~18.04.23", "OpenSSL 1.1.1-1ubuntu2.1", "OpenSSH 8.4p1-2", "Linux kernel 5.4.0", "Linux kernel 4.15.0-1117-fips", "Linux kernel 4.15.0", "Tomcat 9.0.82", "rsyslog 8.32.0-1ubuntu4.2", "rsyslog 8.32.0", "ntp 1:4.2.8p10+dfsg-5ubuntu7.3+esm1", "ntp 1:4.2.8p10", "Intel Xeon", "Intel Xeon CPU E5-2683 v4", "Intel Xeon D-2187NT", "Intel Xeon Gold 6252N", "AMD EPYC 7713P", "AMD EPYC", "VMware ESXi 7.0U2", "ESXi 7.0U2", "Dell VEP 4600", "Dell Poweredge R7515", "MetaSwitch", "Intel DPDK 16.04TCPTCP".