www.GossamerSec.com

Assurance Activity Report for Cisco Firepower NGIPS/NGIPSv 7.0 with FMC/FMCv 7.0

Version 1.1
05/17/2023

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-VID11339

Revision History

Revision	Date	Authors	Summary
Version 0.1	04/17/23	Gossamer	Initial draft
Version 1.0	05/11/23	Cummins	ECR comments addressed
Version 1.0	05/17/23	Cummins	ECR responses addressed

The TOE Evaluation was Sponsored by:
Cisco Systems, Inc.
170 West Tasman Dr.
San Jose, CA 95134

Evaluation Personnel:
- Cody Cummins
- Douglas Kalmus

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 Cisco Firepower NGIPS/NGIPSv 7.0 with FMC/FMCv 7.0 NDcPP22e/IPS10 evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

1.1 Equivalence

1.1.1 Evaluated Platform Equivalence

The TOE is the Cisco Firepower NGIPS/NGIPSv 7.0 with FMC/FMCv7.0. It is a hardware and software solution comprised of a Firepower NGIPS and a Cisco FMC management console.

The TOE consists of the following hardware models running software versions: NGIPSv 7.0 and FMC/FMCv 7.0.

- Cisco Firepower Management Center (FMC) (FMC1000, FMC2500, FMC4500, FMC1600, FMC2600 and FMC4600)
- FMCv running on ESXi 6.7 or 7.0 on the Unified Computing System (UCS) UCSC-C220-M5, UCSC-C240-M5, UCSC-C480-M5, UCS-E160S-M3 and UCS-E180D-M3.
- NGIPSv running on ESXi 6.7 or 7.0 on the Unified Computing System (UCS) UCSC-C220-M5, UCSC-C240-M5, UCSC-C480-M5, UCS-E160S-M3 and UCS-E180D-M3.

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

- Cisco FMC1600 running FMC v7.0
- Cisco FMCv running on ESXi 7.0 with FMC v7.0 on UCSC-C220-M5
- Cisco NGIPSv 7.0 running on ESXi 7.0 on UCSC-C220-M5

There is one software image for each of the 3 hardware model types included in the evaluation. There is one image for all FMC devices, one image for all FMCv devices, and one image for ESXi FTDv devices. The only differences between the images among these devices is related to the different hardware characteristics and does not affect any of the security relevant functionality. Any differing hardware characteristics among the models in each group affect only non-security relevant functionality such as throughput, processing speed, number and type of network connections supported, number of concurrent connections supported, and amount of storage. All security functions provided by the TOE are implemented in software and the TOE security behavior is the same on all the devices for each of the SFRs defined by the Security Target. These SFRs are instantiated by the same version of the TOE software and in the same way on every platform.

Full testing was performed on both the FMC and FMCv. For FMCv on ESXi, the evaluators fully tested on the C220-M5. The same FMCv image is installed on the C-series, B-series and E-series UCS servers included in the evaluated configuration. The differences in hardware characteristics among the UCS servers similarly only affect non-security relevant functionality as described above. Those servers simply provide resources to the FMCv images and are not making security relevant decisions. The evaluators tested on the ESXi 7.0. This is sufficient to address 6.7 as well because the hypervisor does not provide any security function. The differences between 7.0 and 6.7 are in scalability and performance. The same image runs on either version 7.0 or 6.7. The same rationale applies to the NGIPSv image running on ESXi 7.0 or 6.7 on the UCS servers.

The same version of the FMC/FMCv is claimed in multiple subsequent evaluations as it was required to also manage the additional Firepower Threat Defense (FTD) devices from the various efforts. As a result, any evidence of the FMC/FMCv security functionality that was independent of the FTD/NGIPSv was performed once and shared across the subsequent evaluations. Additional devices being managed by the FMC/FMCv did not impact the ability for the FMC/FMCv to properly perform its security functionality nor did it affect the ability for the FMC/FMCv to manage the NGIPSv in this evaluation.

1.1.2 CAVP Equivalence

The TOE models and processors included in the evaluation are shown in the following table. The TOE includes multiple cryptographic modules across the range of TOE components. These modules are commonly referred to as FOM (FIPS Object Models). The CAVP-certified FOM of the TOE are listed in the table below along with the CPU for which they were certified, and the TOE component on which they're used. The table below this one lists the CAVP certificate numbers for each FOM for each applicable SFR.

Table: Processors and Implementations

CPU Family	CPU Model (Microarchitecture)	FOM	Physical Appliances	CAVP Cert#
NGIPSv				
Intel Xeon Scalable w/ Linux 4 on ESXi 6.7/7.0	Intel Xeon Bronze 3104 (Skylake) w/ Linux 4 on ESXi 6.7/7.0	CiscoSSL FOM 7.3sp	UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	A2952
	Intel Xeon Silver 4110 (Skylake) w/ Linux 4 on ESXi 6.7/7.0		UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	
	Intel Xeon® Gold 6128 (Skylake) w/ Linux 4 on ESXi 6.7/7.0		UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	
	Intel Xeon Platinum 8153 (Skylake) w/ Linux 4 on ESXi 6.7/7.0		UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	
Intel Xeon D w/ Linux 4 on ESXi 6.7/7.0	Intel Xeon D-1528 (Broadwell) w/ Linux 4 on ESXi 6.7/7.0		UCS-E160S-M3	
	Intel Xeon D-1548 (Broadwell) w/ Linux 4 on ESXi 6.7/7.0		UCS-E180D-M3	
FMC				
Intel Xeon E5-2600 v4	Intel Xeon E5-2620 v4 (Broadwell)	CiscoSSL FOM 7.3sp	FMC4500	A2585
	Intel Xeon E5 2640 v4 (Broadwell)		FMC1000 and FMC2500	
Intel Xeon Skylake	Intel Xeon Silver 4110 (Skylake)		FMC1600 and FMC2600	
	Intel Xeon Silver 4116 (Skylake)		FMC4600	

CPU Family	CPU Model (Microarchitecture)	FOM	Physical Appliances	CAVP Cert#
FMCv				
Intel Xeon Scalable w/ Linux 4 on ESXi 6.7/7.0	Intel Xeon Bronze 3104 (Skylake) w/ Linux 4 on ESXi 6.7/7.0	CiscoSSL FOM 7.3sp	UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	A2952
	Intel Xeon Silver 4110 (Skylake) w/ Linux 4 on ESXi 6.7/7.0		UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	
	Intel Xeon® Gold 6128 (Skylake) w/ Linux 4 on ESXi 6.7/7.0		UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	
	Intel Xeon Platinum 8153 (Skylake) w/ Linux 4 on ESXi 6.7/7.0		UCSC-C220-M5, UCSC-C240-M5 and UCSC-C480-M5	
Intel Xeon D w/ Linux 4 on ESXi 6.7/7.0	Intel Xeon D-1528 (Broadwell) w/ Linux 4 on ESXi 6.7/7.0		UCS-E160S-M3	
	Intel Xeon D-1548 (Broadwell) w/ Linux 4 on ESXi 6.7/7.0		UCS-E180D-M3	

Table: Algorithm Certificate Numbers

Algorithm	SFR	CiscoSSL FOM 7.3sp (for FMC)	CiscoSSL FOM – Virtual 7.3sp (for NGIPSv/FMCv)
AES CBC 128/256 GCM 128/256	FCS_COP.1/DataEncryption	A2585	A2952
RSA 2048/3072 bits Signature Gen & Verify Key Gen	FCS_COP.1/SigGen FCS_CKM.1	A2585	A2952
DSA 2048/3072 bits	FCS_CKM.1	A2585	A2952
ECDSA curves P-256, P-384 and P-521 Key Sizes – 256, 384 and 521 bits Signature Gen & Verify Key Gen and Verify	FCS_COP.1/SigGen FCS_CKM.1	A2585	A2952
Hashing SHA-1, SHA-256, SHA-384, SHA-512	FCS_COP.1/Hash	A2585	A2952

Keyed Hash HMAC-SHA-1, HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	FCS_COP.1/KeyedHash	A2585	A2952
DRBG (key size 256) CTR_DRBG(AES)	FCS_RBG_EXT.1	A2585	A2952
KAS ECC KAS FFC CVL	FCS_CKM.2	A2585	A2952

1.2 References

The following evidence was used to complete the Assurance Activities:

The ST:

Cisco NGIPSv 7.0 with FMC/FMCv 7.0 Security Target, Version 1.0, May 16, 2023.

The Admin Guide:

Common Criteria Supplemental User Guide for Cisco NGIPSv 7.0 with FMC/FMCv 7.0, Version 1.0, May 16, 2023.

2. Protection Profile SFR Assurance Activities

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

2.1 Security audit (FAU)

2.1.1 Audit Data Generation (NDcPP22e:FAU_GEN.1)

2.1.1.1 NDcPP22e:FAU_GEN.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.1.2 NDcPP22e:FAU_GEN.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1, FAU_GEN.1 the TOE can record activity on the system in two ways. The system can generate an audit record for each user interaction with the web interface and each command in the CLI interface in the audit log, and can also record system status messages in the system log (i.e., syslog). For example, when an administrator imports or deletes a certificate (with associated keys), an audit message is generated that indicates which administrator performed, the action, and includes a unique identifier for the certificate (keys cannot be imported or deleted independently of their associated certificates). Section 6.1, FAU_GEN.1 of the ST includes a table that shows which auditable events are generated, stored, and transmitted by each TOE component. Each component that generates a message also transmits that message to a remote audit server via syslog over TLS; IPS events are generated by NGIPS and are sent to FMC for storage in addition to being sent directly from NGIPS to a remote audit server via syslog over TLS.

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 4.2 “Auditable Events” in the Admin Guide describes the format for the audit records including a description of each field. The appliances that are part of the Cisco Firepower System generate an audit event for each user interaction with the web interface and CLI 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. Section 4.2 describes the common fields and messages found in the audit events for the Web UI, the CLI and Syslog format. Samples of audit messages viewable via the FMC WebUI are shown in screenshots throughout the Admin Guide in sections describing configuration actions relevant to the auditable actions. Section 4.2.1 “Audit Messages Generated by Firepower Services” and section 4.2.2 “Audit Messages Generated by Firepower Management” Center in the Admin Guide also provides tables mapping sample syslog audit records to the SFRs for the NGIPSv and FMC/FMCv respectively. This includes administrative actions found under FMT_MTD.1/CoreData. The evaluator compared the audit table and the SFR claims in the ST and determined all necessary audits were described in guidance including the necessary administrative actions.

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

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

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

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

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

2.1.2.1 IPS10:FAU_GEN.1.1/IPS

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.2.2 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 6.1, IPS_SBD_EXT.1 of ST explains that an administrator configures an IPS sensor in either inline or passive mode, creates intrusion policies composed of intrusion rules, and assigns intrusion policies to access policies which are applied to TOE interfaces. A network analysis policy governs packet processing in phases. First the system decodes packets through the first three TCP/IP layers, then continues with normalizing, preprocessing, and detecting protocol anomalies:

- The packet decoder converts packet headers and payloads into a format that can be easily used by the preprocessors and later, intrusion rules. Each layer of the TCP/IP stack is decoded in turn, beginning with the data link layer and continuing through the network and transport layers. The packet decoder also detects various anomalous behaviors in packet headers.

- The inline normalization preprocessor reformats (i.e., normalizes) traffic to minimize the chances of attackers evading detection. It prepares packets for examination by other preprocessors and intrusion rules, and helps ensure that the packets the system processes are the same as the packets received by the hosts on your network.

- Various network and transport layers preprocessors detect attacks that exploit IP fragmentation, perform checksum validation, and perform TCP and UDP session preprocessing.

- Various application-layer protocol decoders normalize specific types of packet data into formats that the intrusion rules engine can analyze. Normalizing application-layer protocol encodings allows the system to effectively apply the same content-related intrusion rules to packets whose data is represented differently, and to obtain meaningful results.

- Several preprocessors allow administrators to detect specific threats, such as IP/TCP/UDP/ICMP portscans, ICMP/TCP flooding, DoS attacks and other rate-based attacks (“frequency”). The administrator can configure threshold that mimics normal expected frequency and configure the TOE to detect and drop events exceeding the configured thresholds.

Section 6.1, FAU_GEN.1/IPS states the TOE will generate an event log for each intrusion event that occurs (also referred to as an intrusion event). Each event log will include a record of the date, time, type of exploit, and contextual information about the source of the attack and its target. For packet-based events, a copy of the packet or packets that triggered the event is also recorded. Managed Sensors will transmit their events to the FMC where the administrators can view the aggregated data and gain a greater understanding of the attacks against the entire network. The administrators can also deploy the managed Sensors in inline allowing them to configure the Sensors to drop or modify packets that are harmful.

The section states that there is a feature called Threshold where the administrators can control the number of events that are generated per rule over time. They can limit notification to the specified number of event instances per time period or provide notification once per time period after a specified number of event instances. The administrator must specify if the event instances will be tracked by source or destination IP address, the count or the number of event instances, and the number of seconds for the time period for which event instances are tracked.

There are certain header fields that should not be used to trigger intrusion events (in Inline mode or Passive mode). Logging events related to these fields would generate a deluge of intrusion audit records that would prevent IPS analysts from figuring out what security incidents occur in their monitored network. In addition, logging these fields will provide no benefits. Per version 2.11 of IPS EP, the following fields can be inspected and if in inline mode, dropped or modified (i.e., normalized):

- All checksum fields
- TCP Reserved field
- TCP Urgent Pointer field

In inline mode, the TOE can count invalid checksum packets that are dropped. The TOE can also count the packets that gets normalized or dropped because of failed normalization.

The web-based UI is the only way to view the intrusion events (Analysis > Intrusions > Events). Section 6.1, FAU_GEN.1/IPS of ST contains a list of the intrusion event information that can be viewed, searched, filtered, and sorted by the system. Examples of values in this list are Access Control Policy, Access ControlRule, Application protocol, Intrusion Policy, Source User and many other values. In addition, basic contents such as date, time, and type can also be used to filter and sort. The section continues to describe the different fields that are included in logging an intrusion event.

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 4.7 of the Admin Guide describes how to configure IPS policies and rules which cause IPS data logging.

Section 4.7.4 of the Admin Guide describes how to configure audit thresholds for different IPS rules to limit the number of similar audit events that are generated of the same IPS rule in a certain time period.

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.

2.1.3 User identity association (NDcPP22e:FAU_GEN.2)

2.1.3.1 NDcPP22e:FAU_GEN.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

The TSS and Guidance Documentation requirements for NDcPP22e:FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for NDcPP22e: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.

The TSS and Guidance Documentation requirements for NDcPP22e:FAU_GEN.2 are already covered by the TSS and Guidance Documentation requirements for NDcPP22e: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 NDcPP2e: FAU_GEN.1.1

2.1.4 Audit Review (IPS10:FAU_SAR.1)

2.1.4.1 IPS10:FAU_SAR.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.4.2 IPS10:FAU_SAR.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 verify that it describes the ability of administrators to view IPS data from the IPS events, the format in which this IPS data is displayed, and how an administrator is authorized to view this data.

Section 6.1 (FAU_SAR.1 [IPS]) in the ST states that the TOE will generate an event log for each intrusion event that occurs. Each event log will include a record of the date, time, type of exploit, and contextual information about the source of the attack and its target. For packet-based events, a copy of the packet or packets that triggered the event is also recorded. Managed Sensors will transmit their events to the FMC where the administrators can view the aggregated data and gain a greater understanding of the attacks against the entire network. Only Administrators and Intrusion Admins have access to the intrusion events. The administrators can also deploy the managed Sensors inline allowing them to configure the Sensors to drop or modify packets that are harmful. The web-based UI is the only way to view the intrusion events (Analysis > Intrusions > Events). This section further provides a list and description of the intrusion event information that can be viewed, searched, filtered, and sorted by the system. In addition, basic contents such as date, time, and type can also be used to filter and sort.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to verify that it provides instructions on how to access and interpret IPS events using the TOE's management interface.

Section “Management of Intrusion Events” in the Admin Guide states that when the system identifies a possible intrusion, it generates an intrusion event, which is a record of the date, time, the type of exploit, and contextual information about the source of the attack and its target. For packet-based events, a copy of the packet or packets that triggered the event is also recorded. Managed Devices transmit their events to the Firepower Management Center where the aggregated data can be viewed in order to gain a greater understanding of the attacks against network assets. A managed device can also be deployed as an inline, switched, or routed intrusion system, which allows the device to be configured to drop or replace packets that are known to be harmful.

Sections “Viewing Intrusion Events”, “Searching Intrusion Events” and “Sorting and filtering Intrusion Events” in the Admin Guide provide instructions for viewing, searching and filtering intrusion event information. This section also provides a sample view of some intrusion events and a description of each field.

Section “Managing Intrusion Policies” in the Supplement contains multiple subsections, each outlining instructions on performing or understanding the different tasks related to administering intrusion policies. These subsections include instructions for how to access and interpret IPS events and include the following topics: Create Intrusion Policy, View Intrusion Rules in an Intrusion Policy, Intrusion Rule States, Adding and Modifying Intrusion Event Thresholds, Intrusion Rules Editor, Intrusion Rules Import, Configure Dynamic Rule State, Global Rule Threshold, Stateful Session Behaviors, Configure Anomaly Detection, Portscan Detection, Rate-Based Attack Prevention, Specific Attacks, Checksum Verification and Portscan Event Packet View.

Component Testing Assurance Activities: The evaluator shall devise tests that demonstrate that IPS data (generated as defined in FAU_GEN.1/IPS) can be interpreted by authorized administrators from the TOE's management interface.

Throughout the course of all IPS testing the events were logged and stored on the FMC devices. All IPS event data is viewed from an FMC device and is stored in the same manner regardless of the managed device (NGIPSv). The evaluator logged onto the FMC device as an authorized administrator and was able to view details of the IPS events.

2.1.5 Restricted Audit Review (IPS10:FAU_SAR.2)

2.1.5.1 IPS10:FAU_SAR.2.1

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 identifies what group of users is considered to be 'administrators' for the purpose of being granted access to view IPS data.

Section 6.1 (FMT_SMF.1/IPS) in the ST states that the Administrators can deploy intrusion policy with intrusion rules to any interface. An interface, however, can only have one policy applied to that interface. The Administrators can also import vendor-defined signatures from Cisco, create their own intrusion rules, create rules to define which traffic is inspected and analyzed, enable anomaly rules/detections, modify thresholds and threshold duration, and configure known-good/known-bad. The IPS Analysts (Intrusion Admins) can create, modify, or delete intrusion policies but only the IPS Administrators can deploy the policies. Here are the security roles in addition to the all-powerful "Administrator" role.

- "IPS Administrator" (or Administrator): Have all privileges and access
- "IPS Analyst" (or Intrusion Admin): Have all access to intrusion policies, IPS policies and network analysis privileges but cannot deploy policies
- Access Admin: Have all access to access control policies but cannot deploy policies
- Discovery Admin: Have all access to network discovery, application detection, and correlation features but cannot deploy policies
- Security Analyst: Have all access to security event analysis feature

The above defined user account types provide the user with a subset of permissions regarding IPS rules, with only the IPS administrator having access to everything.

Component Guidance Assurance Activities: The evaluator shall examine the operational guidance to determine what actions are needed to grant or revoke a user's ability to view IPS data.

Section "User and Role Management" in the Admin Guide states that the Administrator Role can use the web interface to view and manage user accounts on an FMC or a managed Device, including adding, modifying, and deleting accounts. User accounts without Administrator Role are restricted from accessing user management functions.

Section "Adding New User Accounts" in the Admin Guide provides instructions for configuring user accounts and identifies the access roles that can be granted to the user including the "IPS Administrator" (or Administrator) role which has all privileges and access. The roles that can be selected are consistent with those identified in the ST: IPS Administrator/Administrator, IPS Analyst, Access Admin, Discovery Admin and Security Analyst.

Section "Modifying and Deleting User Accounts" provides instructions to modify any user settings (including access role) or delete an existing user. This can be used to revoke a user's ability to view IPS data.

Component Testing Assurance Activities: The evaluator shall log on to the TOE as various users to confirm that only those users that are intended to be granted read access to IPS data are able to view it.

This test was performed as part of IPS10:FAU_STG.1 where the evaluator logged on as an administrator who had permission to view the IPS data, and a discovery admin who did not have permissions to view the IPS data. The evaluator confirmed that accounts without permissions to view the IPS data could not view it.

2.1.6 Selectable Audit Review (IPS10:FAU_SAR.3)

2.1.6.1 IPS10:FAU_SAR.3.1

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 includes a description of how the TOE has the ability to apply filtering and sorting of IPS data using the parameters listed in the requirement.

Section 6.1 (FAU_SAR.3 [IPS]) in the ST states that the TOE will generate an event log for each intrusion event that occurs. Each event log will include a record of the date, time, type of exploit, and contextual information about the source of the attack and its target. For packet-based events, a copy of the packet or packets that triggered the event is also recorded. Managed Sensors will transmit their events to the FMC where the administrators can view the aggregated data and gain a greater understanding of the attacks against the entire network. Only Administrators and Intrusion Admins have access to the intrusion events. Administrators view the intrusion events via the web-based UI (Analysis > Intrusions > Events). This section further provides a list and description of the intrusion event information that can be viewed, searched, filtered, and sorted by the system. This includes the parameters listed in the requirement such as Application Risk, Destination IP, Source IP, Inline Result (Actions) and Signature ID. In addition, basic contents such as date, time, and event type can also be used to filter and sort.

Component Guidance Assurance Activities: The evaluator shall review the operational guidance to ensure that the guidance itemizes all event types, as well as describes all attributes that are to be selectable in accordance with the requirement, to include those attributes listed in the assignment. The administrative guidance shall also contain instructions on how to set the pre- selection, as well as explain the syntax (if present) for multi-value pre-selection. The administrative guidance shall also identify those audit records that are always recorded, regardless of the selection criteria currently being enforced.

Section 4.9.2 of the Admin Guide addresses the issue of viewing, searching and sorting of Intrusion events. These operations include support for filters available for searching Intrusion events. The set of operations supported by the TOE match those in the Guidance and are the same as that described by the SFR and TSS. Section 4.9.2 also indicates that a priority can be assigned to each field displayed, as a way of defining the sorting of displayed events.

The TOE does not support pre-selection criteria.

No IPS events are always recorded. All IPS events are only recorded if their corresponding Intrusion rule is configured to generate events as described in Section 4.7.3 of the Admin Guide.

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

Test 1: For each attribute listed in the requirement, the evaluator shall devise a test to show that selecting the attribute causes only audit events with that attribute (or those that are always recorded, as identified in the administrative guidance) to be recorded.

Test 2 [conditional]: If the TSF supports specification of more complex audit pre-selection criteria (e.g., multiple attributes, logical expressions using attributes) then the evaluator shall devise tests showing that this capability is correctly implemented. The evaluator shall also, in the test plan, provide a short narrative justifying the set of tests as representative and sufficient to exercise the capability.

Test 1: The evaluator utilized the TOE's IPS event filtering abilities and was able to successfully filter and sort through logged IPS events.

Test 2: Not applicable. The TOE does not support preselection criteria.

2.1.7 Protected Audit Trail Storage (IPS Data) (IPS10:FAU_STG.1/IPS)

2.1.7.1 IPS10:FAU_STG.1.1/IPS

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.1.7.2 IPS10:FAU_STG.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 ensure that the TSS identifies how IPS data is protected from unauthorized modification and deletion.

Section 6.1 (FAU_STG.1 [IPS]) in the ST states that only Administrators and Intrusion Admins have access to intrusion events. The intrusion events cannot be modified but they can be deleted by the Administrators or Intrusion Admins who have restricted access. When the intrusion events storage is full, the newest data will overwrite the oldest data.

Component Guidance Assurance Activities: The evaluator shall confirm the guidance documentation describes how to protect IPS data from unauthorized modification and deletion.

The TOE protects IPS data based upon a user's role. Section 4.9.5 of the Admin Guide identifies the available roles, and permissions for each role.

- "IPS Administrator" (or Administrator): Have all privileges and access
- "IPS Analyst" (or Intrusion Admin): Have all access to intrusion policies and network analysis privileges but cannot deploy policies
- Access Admin: Have all access to control policies but cannot deploy policies
- Discovery Admin: Have all access to network discovery, application detection, and correlation features but cannot deploy policies
- Security Analyst: Have all access to security event analysis feature
- Section 4.9.2 of the Admin Guide states the only accounts able to view intrusion events are accounts that have been assigned the "Administrator" or "Intrusion Admin" roles, and intrusion events can only be viewed via the FMC GUI, they cannot be viewed via CLI on either NGIPS or FMC.

Section 4.9.1 of the Admin Guide states that admin users can delete the IPS audit events but the events cannot be modified.

Component Testing Assurance Activities: The evaluator shall devise tests that demonstrate that IPS data can be protected from unauthorized modification and deletion.

The evaluator logged into the TOE devices using different permission-based accounts and tested the ability to delete IPS data. Only administrators specified to have deletion privileges were able to delete IPS 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 6.1 (FAU_GEN.1) in the ST states the TOE can record activity on the system in two ways. The system can generate an audit record for each user interaction with the web interface and each command in the CLI interface in the audit log and can also record system status messages in the system log (i.e., syslog). For example, when an administrator imports or deletes a certificate (with associated keys), an audit message is generated that indicates which administrator performed the action, and includes a unique identifier for the certificate (keys cannot be imported or deleted independently of their associated certificates).

In addition to auditing administrative activity, the TOE can generate traffic events as part of the intrusion and access control policies and these event records are stored in logs separate from the audit logs for performance and security reasons. More information about the traffic events is presented in IPS sections. Section 6.1 (FAU_GEN.1/IPS) states the TOE will generate an event log for each intrusion event that occurs. Each event log will include a record of the date, time, type of exploit, and contextual information about the source of the attack and its target. For packet-based events, a copy of the packet or packets that triggered the event is also recorded. Managed Sensors will transmit their events to the FMC where the administrators can view the aggregated data and gain a greater understanding of the attacks against the entire network. The administrators can also deploy the managed Sensors in inline allowing them to configure the Sensors to drop or modify packets that are harmful.

Section 6.1 (FAU_STG_EXT.1) in the ST indicates that administrators can configure the system to transmit all the audit events (i.e., audit log and syslog) in real-time over a secure TLS connection to an external audit server in the operational environment. When an audit event is generated, it is sent to the local database and external audit server simultaneously. This ensures that current audit events can be viewed locally while all events, new or old, are stored off-line.

Section 6.1 (FAU_GEN.1) in the ST also indicates the TOE includes an internal log database implementation on FMC that can be used to store and review audit records locally on FMC. The internal audit database on FMC store a maximum of 100,000 audit records (to configure the size, go to System > Configuration > Database, and click on "Audit Event Database"). When the audit log is full, the oldest audit records are overwritten by the newest audit records. In addition, the TOE also includes a local syslog storage in /var/log/messages. Similar to the audit log, when the syslog is full, the oldest syslogs messages are overwritten by the newest one. The NGIPSv, like the FMC, stores all its audit messages locally and when the audit log is full, the oldest audit records are overwritten by the newest audit records. The NGIPSv also sends the IPS events to FMC/FMCv.

For audit log, the events are stored in partitioned event tables. The TOE will prune (i.e., delete) the oldest partition whenever the oldest partition can be pruned without dropping the number of events count below the configured event limit. Note this limit defaults to 10,000 if you set it any lower. For example, if you set the limit to 10,000 events, the events count may need to exceed 15,000 events before the oldest partition can be deleted. For syslog, the logs are stored in /var/log/messages and are rotated daily or when the log file size exceeds 25 MB. After the maximum number of backlog files is reached, the oldest is deleted and the numbers on the other backlogs file are incremented.

Section 6.1, FAU_GEN.1 of the ST includes a table that shows which auditable events are generated, stored, and transmitted by each TOE component. (Refer to the TSS Assurance Activity for NDcPP22e:FAU_GEN.1 above where the table from the ST is provided within this document.) Each component that generates a message also transmits that message to a remote audit server via syslog over TLS; IPS events are generated by NGIPS and are sent to FMC for storage in addition to being sent directly from NGIPS to a remote audit server via syslog over TLS.

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 4.4 of the Admin Guide describes how to configure a secure connection with an audit server using the syslog protocol for both FMC and the NGIPS managed devices. It explains the process for configuring the TOE components with an audit client certificate (i.e., a certificate used to identify the TOE to the audit server). It also explains how to specify the external audit server using the Web UI.

Section 4.4 of the Admin Guide describes that an administrator can configure the system such that it can transmit audit and syslog records securely to an external audit server while also storing the audit and syslog records locally. Section 4.9.1 of the Admin Guide identifies two types of audit data (“Audit” and “syslog”) that are viewed through different Web UI screens. It explains that the “Audit” log stores 100,000 entries, it also indicates that this does not include audit data identified as “syslog”.

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 TLS 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 TLS 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 option “overwrite previous audit records” is selected in FAU_STG_EXT.1.3. The logs for the NGIPSv are stored locally in the filesystem. The evaluator copied random data to the log file until it exceeded limit specified in the administrative guidance. The evaluator observed that the audit data rolled over into a maximum of 4 messages.gz files. The FMC similarly includes a local syslog storage on the filesystem and after 3 to 4 months of testing, the evaluators also found that a listing of the filesystem contents confirmed that log records from the start of testing had been overwritten. Additionally, the FMC includes an internal log database implementation that can be used to store audit records locally up to a configured limit before purging the oldest logs periodically. For testing purposes, the evaluator configured a limit of 5 records for local audit data and performed repeated administrative actions to generate audit data. The evaluator monitored the total number of audit records stored by the TOE before and after this limit was configured. After the new audit storage limit was configured and about 10 minutes had passed, the evaluator confirmed that the number of audits was purged down.

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 6.1 (FAU_STG_EXT.4) in the ST indicates that the TOE includes an internal log database implementation on FMC that can be used to store and review audit records locally on FMC. The internal audit database on FMC store a maximum of 100,000 audit records (to configure the size, go to System > Configuration > Database, and click on “Audit Event Database”). When the audit log is full, the oldest audit records are overwritten by the newest audit records. In addition, the TOE also includes a local syslog storage in /var/log/messages. Similar to the audit log, when the syslog is full, the oldest syslogs messages are overwritten by the newest one. The NGIPSv, like the FMC, stores all its audit messages locally and when the audit log is full, the oldest audit records are overwritten by the newest audit records. The NGIPSv also sends the IPS events to FMC/FMCv.

Section 6.1 (FPT_ITT.) states the communication between the TOE components is protected by TLSv1.2 security protocol.

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 4.9.3 of the Admin Guide describes the process for registering and un-registering the sensors on the FMC device and that this step can only be performed by the administrator role. The section states that before you manage a device with a Firepower Management Center, you must make sure that the network settings are configured correctly on the device. This is usually completed as part of the installation process. In addition, the management network should be an internal, trusted network separated physically or logically from the monitored network.

Section 4.2 of Admin Guide indicates that NGIPS includes an internal log database implementation that can be used to store and review audit records locally. The appliance includes an internal log database implementation that can be used to store and review audit records locally. However, the internal log only stores a default of 100,000 entries in the local database (to configure the size, go to System > Configuration > Database, and click on “Audit Event Database”). When the audit log is full, the oldest audit records are overwritten by the newest audit records. In addition, the appliance also includes a local syslog storage in /var/log/messages. Similar to the audit log, when the syslog is full, the oldest syslogs messages are overwritten by the newest one.

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 TLS channel transporting those audits was demonstrated in NDcPP22E:FTP_ITC.1-t4.

Test 2: This test was performed in conjunction with NDcPP22e:FAU_STG_EXT.1 where results of encrypted audit log transmission between TOE components is captured.

Test 3: This test was performed in conjunction with NDcPP22e:FPT_ITT.1 where encrypted IPS and syslog records are transmitted between TOE components.

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.

See FAU_STG_EXT.4 where this activity has already been performed.

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 4.9.3 of the Admin Guide describes the process for registering and un-registering the sensors on the FMC device and that this step can only be performed by the administrator role. The section states that before you manage a device with a Firepower Management Center, you must make sure that the network settings are configured correctly on the device. This is usually completed as part of the installation process. In addition, the management network should be an internal, trusted network separated physically or logically from the monitored network.

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_ITT.1_t3, demonstrate that the TLS handshake in the ITT channel completed successfully, and application data is seen between the FMC1600/FMCv and the managed device.

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 6.1 (FCO_CPC_EXT.1) in the ST states that in order for TOE components to communicate as part of a distributed TOE System, they must successfully complete a registration process. Each TOE component comes with a manufacture’s TLS certificate. To start the registration process, the administrator must enable or register the TOE components. On the FMC, the administrator must go to Device Management UI and click on “Add Device”. At the same time, the administrator must go to the FTD CLI or GUI, and click or enter “Configure Manager Add”. The administrator must specify the peer hostname or IP address and the registration key used for the initial authentication. During the registration process, the manufacture’s TLS certificates are used to setup the initial TLS channel on the internal trusted management network. If the authentication succeeded, the resident CA on the FMC will sign and issue a TLS certificate along with the private key to the FTD which will be used for subsequent TLS channel. To disable or de-register the FTD, the administrator must initiate a “Delete Device” on the FMC Device Management UI and then perform a “Configure Manager Delete” action on the CLI of the FTD. This will destroy (i.e., zeroize) the TLS certificate and private key. Once this has occurred, no farther communication can happen without another registration process.

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 4.9.3 of the Admin Guide describes the process for registering and un-registering the sensors on the FMC device and that this step can only be performed by the administrator role. The section states that before you manage a device with a Firepower Management Center, you must make sure that the network settings are configured correctly on the device. This is usually completed as part of the installation process. In addition, the management network should be an internal, trusted network separated physically or logically from the monitored network.

The registration process requires: a) manually setting a registration key on the device to be registered, and setting the hostname or IP address of the FMC that will be managing the device; and b) manually setting the same registration key on the FMC, as well as the hostname or IP address of the device being registered. When the key and FMC IP address are set on the device, the device will periodically attempt to establish a TLS connection with the FMC, and will listen for TLS connections from the FMC. Likewise, when the registration key and device hostname or IP address are set on the FMC, the FMC will attempt to initiate a TLS connection to the device and will listen for a TLS connection from that device. When the initial TLS connection is established the device and FMC will authenticate each other using the registration key, and will each generate and exchange new X.509v3 certificates. Those certificates will be used for authentication of all subsequent TLS connections between the device and the FMC.

During device registration if there’s an interruption to the TLS connection between the FMC and the device being registered the registration will fail, and will be automatically reattempted when connectivity is resumed.

To de-register a Device, just click on the trash can icon next to the Device you want to remove.

Section 4.3 of the Admin Guide describes the steps to configure the TOE in “CC” mode. Enabling CC mode will restrict the SSH algorithms, SSH rekey, TLS versions and TLS cipher suites (including elliptical curves) to the Approved ones claimed in the ST. There are additional features such as enabling the power-up integrity HMAC-SHA-512 self-test, enabling FIPS mode, and other TLS required checks such as the ones specified in section 6 of RFC 6125.

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: The evaluator verified the Distributed TOE Communication is not occurring, registered the TOE NGIPS component and verified distributed communication does occur.

Test 2: The evaluator verified that the TOE NGIPS component was registered then unregistered the component. The evaluator then verified that TOE communication between distributed components was disabled.

Test 3 - This test is met by NDcPP22e:FPT_ITT.1 where the ITT channel was tested and shown to encrypt all ITT communication.

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 6.1 (FCS_CKM.1) in the ST provides a table which identifies the usage for each scheme. The TOE (FMC and the NGIPSv) uses a cryptographic module (i.e., Cisco FIPS Object Module) to provide supporting cryptographic functions. The TOE supports RSA, FFC, and ECDSA in the evaluated configuration. The RSA modulus key size is 2048 bit, which according to NIST PUB 800-57, is equivalent to a symmetric key strength of 112 bits. The FMC component of the TOE generates ECC curve P-256 remote administration via TLSv1.2. the TOE also supports FFC Schemes using “safe-prime” groups (i.e., DH group 14) key generation/establishment scheme that meets standard RFC 3526, section 3 for interoperability.

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 4.3 of Admin Guide describes the steps to configure the TOE in “CC” mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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 section entitled, "CAVP Equivalence" earlier in this document.

2.3.2 Cryptographic Key Establishment (NDcPP22e:FCS_CKM.2)

2.3.2.1 NDcPP22e:FCS_CKM.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

```
Scheme          |          SFR          |        Service

-------------------------------------------------------------------------------------------

RSA             |  FCS_TLSS_EXT.1  | Administration

-------------------------------------------------------------------------------------------

ECDH            |  FCS_SSHC_EXT.1  | Audit Server

-------------------------------------------------------------------------------------------

ECDH            |  FCS_IPSEC_EXT.1 | Authentication Server

-------------------------------------------------------------------------------------------
```

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

(TD0580 applied)

Section 6.1 (FCS_CKM.2) in ST is consistent with the operations in the SFR and states the TOE supports RSA-based and Elliptic Curve-based schemes as part of the TLS session establishments. In addition, the TOE also supports FFC Schemes using “safe prime” groups (i.e., DH group 14) key establishment scheme that meets standard RFC 3526, section 3 for interoperability. The section also provides a table mapping key establishment methods to security protocols and services.

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

Section 4.3 of the Admin Guide describes the steps to configure the TOE in “CC” mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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 section entitled, "CAVP Equivalence" earlier in this document. FFC safe primes has been tested using a known good implementation

2.3.3 Cryptographic Key Destruction (NDcPP22e:FCS_CKM.4)

2.3.3.1 NDcPP22e:FCS_CKM.4.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

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

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

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

Section 6.1 (FCS_CKM.4) in the ST indicates that the TOE is designed to zeroize secret and private keys when they are no longer required by the TOE. The secret keys used for symmetric encryption, private keys, and CSPs used to generate keys, are zeroized immediately on system shutdown. For plaintext keys, the TOE destroys the reference to the keys stored in volatile memory directly followed by a request for garbage collection; the TOE destroys the abstraction that represents the key for keys stored in non-volatile storage the TSF.

Section 7.2 in the ST provides a table which identifies the applicable secret and private keys, their storage location and how and when they are zeroized. The ST does not identify any circumstances or configurations where the key destruction does not conform to the requirement.

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 ST states that the TOE meets all requirements specified in FIPS 140-2 for destruction of keys and Critical Security Parameters (CSPs). The ST does not identify any circumstances or configurations where the key destruction does not conform to the requirement.

Section 2.2. of the Admin Guide describes a situation that could prevent or delay plaintext key destruction. The TOE contains SSD storage media in all hardware appliances, and could also contain SSD storage on an NGIPSv and FMCv (the underlying Cisco UCS server hardware supports SSD storage options). SSD storage devices use wear-leveling that could result in blocks of residual data remaining when the SSD marks worn blocks as inactive. When these TOE components are being decommissioned, TOE administrators should follow their own organizational security policies and guidelines for destruction of sensitive data on wear-leveling SSD storage media.

Component Testing Assurance Activities: None Defined

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

2.3.4.1 NDcPP22e:FCS_COP.1.1/DataEncryption

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 7.3, Table 24 in the ST identifies the AES algorithm used in CBC and GCM mode with key sizes 128 and 256 bits.

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

Section 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. This includes the encryption algorithms and key sizes used in support of these ciphersuites, protocols and versions. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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 section entitled, "CAVP Equivalence" earlier in this document.

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

2.3.5.1 NDcPP22e:FCS_COP.1.1/Hash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FCS_COP.1/Hash) in ST indicates that the TOE provides cryptographic hashing services using SHA-1, SHA-256, SHA-384, and SHA-512. This indicates that the hash algorithms also support keyed-hashing. This section states that the TOE supports SHA hashing algorithms for hashing of communication data, in support other security protocols such as SSHv2 and TLSv1.2.

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 4.3 of the Admin Guide describes the steps to configure the TOE in “UCAPL/CC” mode which defines the set of ciphersuites, versions and protocols that the TOE supports. This includes the hash sizes used in support of these ciphersuites, protocols and versions. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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 section entitled, "CAVP Equivalence" earlier in this document.

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

2.3.6.1 NDcPP22e:FCS_COP.1.1/KeyedHash

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FCS_COP.1/KeyedHash) in ST indicates the TOE provides cryptographic hashing services using SHA-1 (, SHA-256, SHA-384, and SHA-512, and keyed-hash message authentication using HMAC-SHA-1 (key length and output length of 160-bits), HMAC-SHA-256 (key length and output length of 256-bit), HMAC-SHA-384 (key length and output length of 384-bit), and HMAC-SHA-512 (key length and output length of 512-bit) with block size of 64 bytes (HMAC-SHA-1 and HMAC-SHA-256) and 128 bytes (HMAC-SHA-384 and HMAC-SHA-512).

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 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. This includes the values used by the HMAC function: key length, hash function used, block size, and output MAC length used in support of other TOE security features. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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 section entitled, "CAVP Equivalence" earlier in this document.

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

2.3.7.1 NDcPP22e:FCS_COP.1.1/SigGen

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

Section 6.1 (FCS_COP.1/SigGen) in ST indicates the TOE supports Approved RSA and ECDSA digital signature algorithm for signature generation and verification, in support of digital certificates. The TOE provides signature services for 2048-bit RSA keys, as well as ECDSA signatures using curves P-256, P-384 and P-521.

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

Section 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. This includes the cryptographic algorithms and key sizes for signature services used in support of these ciphersuites, protocols and versions. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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 section entitled, "CAVP Equivalence" earlier in this document.

2.3.8 HTTPS Protocol (NDcPP22e:FCS_HTTPS_EXT.1)

2.3.8.1 NDcPP22e:FCS_HTTPS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.2 NDcPP22e:FCS_HTTPS_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.8.3 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 6.1 (FCS_HTTPS_EXT.1) in the ST indicates that the TOE implements HTTP over TLS (or HTTPS) to support remote administration on FMC. A remote administrator can connect over HTTPS to the TOE with their web browser. The TOE implements HTTPS according to RFC 2818 by using a TLSv1.2 session to secure the HTTP connection. If the TLS client does not support TLSv1.2, the TLS connection will fail and the administrators will not establish a HTTPS web-based session with the TOE.

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.

The ST indicates TOE supports HTTPS web-based secure administrator sessions. The only configuration necessary in order to cause the TOE to provide its HTTPS web-based administrative interface is to specify a certificate to be used by the TOE. Section 4.9.4 of the Admin Guide describes how to configure a custom web server certificate in the FMC.

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 eval. The testing of the HTTPS channel was demonstrated in FCS_TLSS_EXT.1 and FTP_TRP.1.

2.3.9 Random Bit Generation (NDcPP22e:FCS_RBG_EXT.1)

2.3.9.1 NDcPP22e:FCS_RBG_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.9.2 NDcPP22e:FCS_RBG_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

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

Section 6.1 (FCS_RBG_EXT.1) in the ST states that the TOE uses Approved NIST 800-90 DRBG implementation to generate random numbers for generating cryptographic keys, and seeds the DRBG with a hardware-based entropy source.

In addition, the DRBG is seeded by an entropy source that is at least 256-bit value derived from various highly sensitive and proprietary noise sources described in the proprietary Entropy Design document.

Section 7.3, Table 24 in the ST identifies the DRBG type as CTR_DRBG (AES) which is consistent with the requirement.

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) Cisco proprietary document that has been delivered to NIAP for approval. Note that the entropy analysis has been accepted by NIAP/NSA.

Section 4.3 of the Admin Guide contains subsection labelled “Enable CC Compliance (also known as CC Mode)” which outlines what must be done in order to enable CC mode, thus also enabling 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 section entitled, "CAVP Equivalence" earlier in this document.

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

2.3.10.1 NDcPP22e:FCS_SSHS_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.3.10.2 NDcPP22e:FCS_SSHS_EXT.1.2

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

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

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

Section 6.1 (FCS_SSHS_EXT.1) in the ST indicates that the TOE supports SSH public key authentication using ssh-rsa (NGIPSv) or ecdsa-sha2-nistp256 (FMC/FMCv).. This is consistent with the algorithms selected in FCS_COP.1/SigGen. The TOE ensures and verifies that the SSH client's presented public key matches one that is stored within the TOE's SSH server's authorized keys file. The TOE also supports password-based user authentication for SSH.

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)

The following tests were performed on the FMC/FMCv and the NGIPSv.

Test 1: This test has been performed as part of FIA_UIA_EXT.1-t1 where the evaluator demonstrated a successful login using an RSA public key. The TOE claims support for RSA public keys only.

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. The TOE claims support for RSA public keys only.

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.

2.3.10.3 NDcPP22e:FCS_SSHS_EXT.1.3

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

Section 6.1 (FCS_SSHS_EXT.1) in the ST states that SSH connections will be dropped if the TOE receives a packet larger than 262149 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 following test was performed on the FMC/FMCv and the NGIPSv.

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.

2.3.10.4 NDcPP22e:FCS_SSHS_EXT.1.4

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

Section 6.1 (FCS_SSHS_EXT.1) of ST explains that the TOE supports SSHv2 with AES (in CBC or GCM mode) 128 or 256 bits cipher for encryption. The encryption algorithms match the SFR claims for FCS_SSHS_EXT.1.4.

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 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of SSH protocol parameters that the TOE supports. Once in CC mode, the TOE supports only the encryption algorithms, MAC algorithms, and exchange methods defined by the ST.

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 following test was performed on the FMC/FMCv and the NGIPSv.

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.

2.3.10.5 NDcPP22e:FCS_SSHS_EXT.1.5

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

Section 6.1 (FCS_SSHS_EXT.1) in the ST indicates that the TOE supports SSH host key implementation using rsa-sha2-256, or rsa-sha2-512 as its public key algorithm. These algorithms match the ST claims identified in the selection of FCS_SSHS_EXT.1.5. The TOE saves public keys in an "authorized_keys" file stored in that user's home directory on the TOE component.

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 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of SSH protocol parameters that the TOE supports. Once in CC mode, the TOE supports only the encryption algorithms, MAC algorithms, and exchange methods defined by the ST. The section also describes how to configure SSH public key authentication.

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)

The following tests were performed on the FMC/FMCv and the NGIPSv.

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.

2.3.10.6 NDcPP22e:FCS_SSHS_EXT.1.6

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

Section 6.1 (FCS_SSHS_EXT.1) of ST explains that the TOE supports SSHv2 with HMAC-SHA1, HMAC-SHA-256 or HMAC-SHA-512 for integrity and authenticity. This section also indicates that the TOE supports AES-GCM, which also provides integrity and authenticity for SSHv2. Support for AES GCM encryption algorithms indicates the implicit support for AES GCM MAC algorithms.

This claim in the TSS is the same as the data integrity MAC algorithm identified by the selections in the requirement.

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 4.3 of the Admin Guide contains subsection labelled "Enable CC Compliance (also known as CC Mode)" which outlines what must be done in order to enable CC mode, thus also enabling FIPS mode. This ensures only the allowed data integrity algorithms are used by SSH.

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.

The following tests were performed on the FMC/FMCv and the NGIPSv.

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.

2.3.10.7 NDcPP22e:FCS_SSHS_EXT.1.7

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

Section 6.1 (FCS_SSHS_EXT.1) in the ST indicates that the TOE supports the diffie-hellman-group14-sha1 key exchange method. This is consistent with the requirement.

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 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of SSH protocol parameters that the TOE supports. Once in CC mode, the TOE supports only the encryption algorithms, MAC algorithms, and key exchange methods defined by the ST.

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.

The following tests were performed on the FMC/FMCv and the NGIPSv.

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: diffiehellman-group14-sha1. The connection succeeded.

2.3.10.8 NDcPP22e:FCS_SSHS_EXT.1.8

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

a) Both thresholds are checked by the TOE.

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

Section 6.1 (FCS_SSHS_EXT.1) in the ST indicates that the TOE manages a tracking mechanism for each SSH session so that it can initiate a new key exchange when either approximately 1 hour of time or 1GB of data is reached. An audit event is generated when a successful SSH rekey occurs when either of the thresholds occur.

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 4.3 of the Admin Guide describes that when CC mode is enabled, SSH re-keying will occur approximately at 1 hour of time or after 1 GB of data has been transmitted, whichever occurs first. To change these values to be smaller, the administrator can configure these during the pre-operational state ONLY using the local management connection. This section goes on to provide step-by-step instructions to change these limits.

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 following tests were performed on the FMC/FMCv and the NGIPSv.

The SSH Data and time rekey thresholds are only configurable in the pre-operational state, therefore the default of 1GB and 1 hour was tested for all devices. The evaluator attempted to connect to the TOE using a SSH client generating 1GB 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

2.3.11 TLS Client Protocol Without Mutual Authentication - per TD0634 & TD0670 (NDcPP22e:FCS_TLSC_EXT.1)

2.3.11.1 NDcPP22e:FCS_TLSC_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 include those listed for this component.

Section 6.1 (FCS_TLSC_EXT.1) in the ST states that the TOE implements TLS clients to support secure syslog connections, and TLS server and clients to support FPT_ITT.1. This section specifies the ciphersuites for each TOE component and TLS communication channel by reference to the requirements in Section 5.3.3.11.

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.

Section 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

Section 4.9.3 of the Admin Guide Describes the process of registering a sensor to a Firepower Management device. This registration is necessary to enable distributed TOE communication over TLS.

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

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

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

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

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

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

Test 5: The evaluator shall perform the following modifications to the traffic:

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

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

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

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

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

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

These tests were iterated on all TOE components for 2 variations: Distributed TOE communication and Remote Syslog. For all TOE components the evaluator performed the following tests:

Test 1: The evaluator established a TLS session from the TOE to a test server with the test server configured to accept connections with only one of the claimed cipher suites. The evaluator used a network sniffer to capture the TLS session negotiation. The evaluator examined each traffic capture and observed that the expected TLS cipher was negotiated. The evaluator repeated this test for each claimed ciphersuite.

Test 2: As part of FIA_X509_EXT.1, test 1, the evaluator configured the test server to send a certificate with the Server Authentication purpose in the extendedKeyUsage field and observed that the TLS session was accepted by the TOE. Next the TLS session was attempted again with the test server using a cert that is missing the Server Authentication purpose in the extendedKeyUsage field. The evaluator captured the TLS session negotiation and observed that the TLS session was rejected by the TOE.

Test 3: The evaluator established a TLS session from the TOE. A modified test server negotiates an RSA ciphersuite, but returns an ECDSA Certificate. Using a network sniffer to capture the TLS session negotiation, the evaluator observed that the TLS session was rejected.

Test 4: The evaluator configured a test server to accept only the TLS_NULL_WITH_NUL_NULL ciphersuite. The evaluator then attempted to establish a TLS session from the TOE to that test server. Using a network sniffer, the evaluator captured the TLS session negotiation and observed that the TLS session is rejected by the TOE. The evaluator then configured the test server to send a client hello with a ciphersuite that is not supported by the TOE The TLS session is rejected by the TOE. As the TOE supports ciphersuites with ECDH key exchange, the evaluator set up the test server to attempt a TLS session negotiation using an unsupported curve size (P-192). The TOE successfully rejected the connection attempt.

Test 5: The evaluator configured the test server to present a server hello with an un-supported TLS version (version 1.4). The TOE rejected the connection. The evaluator then configured the test server to modify the signature block in the server’s key exchange message. The TOE rejects the connection attempt and no application data flows.

Test 6: The evaluator configured the test server to modify a byte in the server’s Finished handshake message. The TOE rejects the connection attempt and there is no application data flow. The evaluator then modified the test server to send a garbled message after the server issued the ChangeCipherSpec message. The TOE rejects the connection attempt and there is no application data flow. The evaluator then modified the test server to modify a byte in the server hello handshake message nonce. The TOE rejects the connection attempt as appropriate. In the case of the FMC/FMCv, the server denies the client's Finished handshake message due to the use of an RSA ciphersuite and in the case of the NGIPSv, the client rejects the Server Key Exchange handshake message due to the use of an ECDHE ciphersuite.

2.3.11.2 NDcPP22e:FCS_TLSC_EXT.1.2

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

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

If IP addresses are supported in the CN as reference identifiers, the evaluator shall ensure that the TSS describes the TOE's conversion of the text representation of the IP address in the CN to a binary representation of the IP address in network byte order. The evaluator shall also ensure that the TSS describes whether canonical format (RFC5952 for IPv6, RFC 3986 for IPv4) is enforced.

Section 6.1 (FCS_TLSC_EXT.1) of the ST explains that when the TOE acts as a TLS client (e.g., connection to the syslog server), the TOE will verify the server Common Name (CN) and/or Subject Alternative Name (SAN) against the reference identity (wild-card is supported as required in section 6 of RFC 6125). When TOE components are communicating with each other (as part of this distributed TOE), they verify each other using unique reference identifiers (UID) by verifying the UID in the title field of the certificates subject (wild-cards are not supported for this communication, and are an optional part of the requirement). The title field (id-at-title) is used per RFC 5280 Appendix A. In either case (connection to a syslog server, or connections among TOE components), if verification fails, the TLS connection will not be established.

The TOE does not perform a discovery of additional TOE components, instead Section 6.1 (FCO_CPC_EXT.1) of the ST describes the registration process used by the TOE to associate TOE components with one another. This section also indicates that because the administrator can choose the registration key and must provide a specific hostname/IP address, while initiating the registration on both TOE components, the registration action can be assured of the unique identification of the TOE components.

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 includes support for IP addresses, the evaluator shall ensure that the operational guidance provides a set of warnings and/or CA policy recommendations that would result in secure TOE use.

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

Section 4.4 of the Admin Guide explains If the SAN and DNS hostname do not match, or if the SAN is not present and the CN and the DNS hostname do not match, the secure audit log connection will fail. The section entitled "Specify the external audit server" states that for NGIPS and FMC the reference identifier used for the syslog-over-TLS connection to the remote syslog server is the syslog server’s hostname or IP address as entered by the Firepower administrator when adding the syslog host to the configuration.

Section 4.9.3 of the Admin Guide describes how the UUID is used as an identifier for components certificates for distributed TOE communication.

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

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

or

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

or

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

Note that for some tests additional conditions apply.

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

- IPv4: The CN contains a single address that is represented a 32-bit numeric address (IPv4) is written in decimal as four numbers that range from 0-255 separated by periods as specified in RFC 3986.

- IPv6: The CN contains a single IPv6 address that is represented as eight colon separated groups of four lowercase hexadecimal digits, each group representing 16 bits as specified in RFC 4291. Note: Shortened addresses, suppressed zeros, and embedded IPv4 addresses are not tested.

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

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

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

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

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

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

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

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

2) [conditional]: The evaluator shall present a server certificate containing a wildcard in the left-most label (e.g. *.example.com). The evaluator shall configure the reference identifier with a single left-most label (e.g. foo.example.com) and verify that the connection succeeds if wildcards are supported or fails if wildcards are not

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

f) Objective: The objective of this test is to ensure the TOE is able to differentiate between IP address identifiers that are not allowed to contain wildcards and other types of identifiers that may contain wildcards.

Test 6 [conditional]: If IP address identifiers supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g.CN=*.168.0.1 when connecting to 192.168.0.1.20, CN=2001:0DB8:0000:0000:0008:0800:200C:*when connecting to 2001:0DB8:0000:0000:0008:0800:200C:417A). The certificate shall not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each supported IP address version (e.g. IPv4, IPv6). (TD0634 applied)

This negative test corresponds to the following section of the Application Note 64: 'The exception being, the use of wildcards is not supported when using IP address as the reference identifier.'

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

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

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

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

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

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

For all TOE components the evaluator performed the following tests:

Test 1: The evaluator attempted a TLS session from the TOE targeting a server using a valid certificate with a CN matching the domain name used by the client and observed that the connection was successful. The evaluator then attempted a TLS session from the TOE targeting a server using a server certificate that contains an identifier that does not match the Common Name (CN) and omits the Subject Alternative Name (SAN) and observed that connection was rejected.

Test 2: The evaluator attempted a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier, contains the SAN extension, but does not contain an identifier in the SAN that matches the reference identifier. The evaluator observed that the connection was rejected.

Test 3: The evaluator attempted a TLS session from the TOE targeting a server using a server certificate that contains a CN that matches the reference identifier and does not contain the SAN extension. The evaluator observed that the connection was successful.

Test 4: The evaluator attempted a TLS session from the TOE targeting a server using a server certificate that contains a CN that does not match the reference identifier but does contain an identifier in the SAN that matches. The evaluator observed that the connection was successful.

Test 5: The evaluator tested hostname wildcards by configuring an expected DNS on the TOE with the test server's certificates configured with wildcard DNS names. The TOE successfully checked the hostname wildcards and behaved as expected.

Test 6: The TOE does not support IP addresses as the reference identifier.

Test 7: For ITT (Distributed TOE) communications, the ST claims to support matching the id-at-title according to RFC 5280 Appendix A to the presented reference identifier. As such, the evaluator configured the TOE to connect with the distributed TOE peer using TLS with the peer alternately configured with a certificate identifier as listed below. The evaluator verified that the TOE only connected when the identifier fulfilled the required rules:

- Incorrect UUID - the TOE rejected the certificate and the connection failed as expected.

- Correct UUID, wrong attribute - the TOE rejected the certificate and the connection failed as expected.

- Correct UUID, no SAN - the TOE accepts the certificate and proceeds with successful session establishment.

- Wildcard in left side of UUID - the TOE detects a mismatch and rejects the connection.

- Wildcard in right side of UUID - the TOE detects a mismatch and rejects the connection.

2.3.11.3 NDcPP22e:FCS_TLSC_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

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

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

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

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

Test 1: This test was performed as part of NDcPP22e:FCS_TLSC_EXT.1.1-t1 where the various cipher suites were tested. Each successful TLS connection was using a valid cert chain.

Test 2: This test has been performed in several other test activities. Specifically, this test repeats the assurance activities as described here.

- match the reference identifier -- Corresponds to FCS_TLSC_EXT.1.2 Tests 1 through 7.
- validate certificate path -- Corresponds to FIA_X509_EXT.1/REV.1 Test 1
- validate expiration date -- Corresponds to FIA_X509_EXT.1/REV.1 Test 2
- determine the revocation status -- Corresponds to FIA_X509_EXT.2 Test 1.

Test 3: Not applicable. The TOE does not support any override mechanisms.

2.3.11.4 NDcPP22e:FCS_TLSC_EXT.1.4

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

Section 6.1 (FCS_TLSC_EXT.1) in the ST states that the following NIST curves are presented in the Client Hello by default - secp384 r1, secp521r1 and secp256r1.

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

The Admin Guide does not specifically address support for supported elliptic curve extensions. Rather, it indicates that TLS version and TLS ciphersuites are forced to those claimed by the Security Target, when the device is configured into "CC Mode".

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

These tests were iterated on all TOE components for 2 variations: Distributed TOE communication and Remote Syslog. For all TOE components the evaluator performed the following tests:

The evaluator configured the test server to support only one ECDHE group at a time. The evaluator then attempted a connection from the TOE to the test server. The TOE successfully allowed the connection attempts for all of the supported curve sizes.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.3.12 TLS Client Support for Mutual Authentication - per TD0670 (NDcPP22e:FCS_TLSC_EXT.2)

2.3.12.1 NDcPP22e:FCS_TLSC_EXT.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 TSS description required per FIA_X509_EXT.2.1 includes the use of client-side certificates for TLS mutual authentication.

Section 6.1 (FCS_TLSC_EXT.2) of ST explains mutual authentication must be configured with the client-side X.509v3 certificate.

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

Section 4.4 of the Admin Guide explains the process for generating a CSR and importing the X509 certificate to be used by the TOE as a client-side certificate during TLS mutual authentication to a remote audit server. The section also describes the configuration necessary to enable mutual authentication for syslog over TLS.

Section 4.9.3 the Admin Guide states that for distributed TOE communication the FMC and NGIPS generate unique certificates with distinct UUIDs at install and as part of the registration process. Those certificates are only used for communications between Firepower devices, are not configurable, and do not provide interoperability with non-Firepower devices.

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

Test 1: The evaluator shall establish a connection to a peer server that is configured for mutual authentication (i.e. sends a server Certificate Request (type 13) message). The evaluator observes that the TOE TLS client sends both client Certificate (type 11) and client Certificate Verify (type 15) messages during its negotiation of a TLS channel and that Application Data is sent.

In addition, all other testing in FCS_TLSC_EXT.1 and FIA_X509_EXT.* must be performed as per the requirements.

The following TLS channels in the TOE support mutual authentication:

FMC1600 Syslog

FMC Virtual Syslog

NGIPSv Syslog

Test 1: The evaluator configured the test server for mutual authentication and attempted a connection from the TOE acting as the TLS Client for the above channels. The evaluator verified that the TOE sent a client “Certificate” message as well as a “Certificate Verify” message. Additionally, evaluator verified that the negotiation was successful and that application data was sent.

In addition, testing of FCS_TLSC_EXT.1.1 Test 1 and FIA_X509_EXT for these channels were performed as per the requirements.

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

2.3.13.1 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 6.1 (FCS_TLSS_EXT.1) in the ST states that the TOE implements HTTP over TLS (or HTTPS) to support remote administration on FMC and TLS server and clients to support FPT_ITT.1. This section specifies the ciphersuites for each TOE component and TLS communication channel by reference to the requirements in Section 5.3.3.13.

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 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

Section 4.9.3 of the Admin Guide describes the process of registering a sensor to a Firepower Management device. This registration is necessary to enable distributed TOE communication over TLS.

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'.

These results were iterated for 2 variations: HTTPS and ITT (Distributed TOE communications). HTTPS tests were performed on the FMC components. ITT tests were performed on the FMC components and NGIPSv.

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 cipher suite 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. In Scenario 3 a), the client implementation of the TLS protocol was modified as stated in part a 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.

2.3.13.2 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 6.1 (FCS_TLSS_EXT.1) of ST states the TOE is restricted to only support TLSv1.2 for HTTPS sessions and client/server communications between TOE components and both TLSv1.1 and TLSv1.2 for syslog communications. Any TLS or SSL versions not supported is rejected by the TOE.

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

Section 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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.

These results were iterated for 2 variations: HTTPS and ITT (Distributed TOE communications). HTTPS tests were performed on the FMC components. ITT tests were performed on the FMC components and NGIPSv.

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.

2.3.13.3 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 6.1 (FCS_TLSS_EXT.1) in the ST states that the key establishment parameters for each of the TLS connections in the TOE are as follows:

1. FMC/FMCv (HTTPS/TLS)- 2048-bit RSA and ECDHE secp256r1, secp384r1 and secp521r1
2. FMC/FMCv and NGIPSv (FPT_ITT.1) - 2048-bit RSA and ECDHE secp256r1, secp384r1 and secp521r1

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

Section 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

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.

These results were iterated for 2 variations: HTTPS and ITT (Distributed TOE communications). HTTPS tests were performed on the FMC components. ITT tests were performed on the FMC components and NGIPSv.

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: Not applicable. The TOE does not support DHE ciphersuites.

Test 3: The evaluator attempted to establish a TLS session with the TOE and configure the remote peer (i.e., the client) to offer each RSA 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 RSA key sizes.

2.3.13.4 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 6.1 (FCS_TLSS_EXT.1) in the ST indicates that the TOE provides session resumption only for remote administration sessions. Session resumption is not supported by the NGIPS or for Internal TOE communication (i.e., FPT_ITT.1). Only the FMC can resume a remote administration HTTPS session based upon session tickets.

Session tickets are encrypted using the symmetric algorithms and key lengths associated with the negotiated ciphersuite and which are consistent with the selections from FCS_COP.1/DataEncryption. Session tickets adhere to the structural format provided in section 4 of RFC 5077.

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: For the NGIPSv/FMC ITT channel, 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, and the TOE also sends a zero-length SessionID to the remote peer.

Test 2: Not applicable. The TOE does not support session resumption based on session ID.

Test 3: For the FMC web UI, the evaluator first attempted to resume a session using a valid session ticket. The TOE correctly reuses the client's proposed session ticket and the session is successfully resumed. In this case, the sessionID in the second ServerHello packet matches the client hello's proposed sessionID. The evaluator then attempted a second connection in which the session ticket is modified to be different from the server provided session ticket. The TOE correctly rejects the proposed session ticket as an invalid ticket and sends a new NewSessionTicket packet and performs a full handshake.

Component TSS Assurance Activities: None Defined

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.4 Identification and Authentication (FIA)

2.4.1 Authentication Failure Management (NDcPP22e:FIA_AFL.1)

2.4.1.1 NDcPP22e:FIA_AFL.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.1.2 NDcPP22e:FIA_AFL.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

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

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

Section 6.1 (FIA_AFL.1) of the ST states that the administrator can configure the maximum number of times each user can try to login after a failed login attempt before the account is locked. The default setting is five tries. Once the number of failed attempts has exceeded the configured limit, the user will be locked out. Only administrator or user with admin privileges can unlock the user.

By default, the predefined 'admin' account is exempt from becoming locked, but that default is overridden when CC mode is enabled. If all admin accounts become locked for any reason, FMC can be accessed locally using password recovery procedures.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to ensure that instructions for configuring the number of successive unsuccessful authentication attempts and time period (if implemented) are provided, and that the process of allowing the remote administrator to once again successfully log on is described for each 'action' specified (if that option is chosen). If different actions or mechanisms are implemented depending on the secure protocol employed (e.g., TLS vs. SSH), all must be described.

The evaluator shall examine the guidance documentation to confirm that it describes, and identifies the importance of, any actions that are required in order to ensure that administrator access will always be maintained, even if remote administration is made permanently or temporarily unavailable due to blocking of accounts as a result of FIA_AFL.1.

Section 4.9.5 of the Admin Guide describes the method to configure a maximum number of failed login attempts and that the default number is 5 failed login attempts. The maximum number of failed login attemptsis configured as part of adding/editing a user. The section further describes how the accounts can be unlocked by another account with the "Administrator" role by resetting the account activation switch.

Time period is not selected as a method of unlocking for any of the TOE components.

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 5 for the FMC/FMCv and 3 for the NGIPSv. The evaluator then logged in to the TOE using incorrect credentials five times (FMC) and three times (NGIPSv). The evaluator then attempted to use correct credentials and found that the account had been locked out in each case. This test was repeated on FMC and the NGIPSv devices.

Test 2: Continuing Test 1, the evaluator then successfully unlocked the account in each case using the procedures found in the guidance. This test was repeated on both FMC and NGIPSv devices.

2.4.2 Password Management (NDcPP22e:FIA_PMG_EXT.1)

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

Section 6.1 (FIA_PMG_EXT.1) of ST lists supported special characters and provides information about minimum and maximum number of characters supported for administrator passwords. This section states when creating or changing passwords, the passwords must be composed of upper and lower case letters, numbers and special characters including blank space and !@#$%^&*() '(double or single quote/apostrophe), + (plus), - (minus), = (equal), , (comma), . (period), / (forward-slash), \ (back-slash), | (vertical-bar or pipe), : (colon), ; (semi-colon), < > (less-than, greater-than inequality signs), [ ] (square-brackets), { } (braces or curly-brackets),? (question-mark), (underscore), and ~ (tilde). The password must have at least one upper case, one lower case, one number, and one special character. This is configured by checking on "Check Password Strength" option per each user (See CC Supplement User Guide for details). Also, the passwords have to satisfy configured minimum password length which is set in the System Policy for all users. The minimum password length can range from 8 (default) to 127 characters (maximum) long, which includes 15 characters required by the NDcPP. Note: The user password is limited to 127 characters maximum.

Component Guidance Assurance Activities: The evaluator shall examine the guidance documentation to determine that it:
a) identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and
b) provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

Section 4.9.6 of the Admin Guide indicates that all accounts are protected by a password. It indicates that users can change their own password. Section 4.9.5 describes that the TOE can be configured to expire a password or to enforce password strength checking which enforces a minimum length of 8 characters (among other constraints). No other minimum length configuration is possible. It also describes that if password strength checking is enabled, passwords must be at least eight alphanumeric characters of mixed case and must include at least one number and one special character. Passwords 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 each TOE component 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 each TOE component with compositions including the following. All invalid password attempts were rejected.
-short password
-long password
-no numbers
-no special character
-no uppercase
-no lowercase

2.4.3 Protected Authentication Feedback (NDcPP22e:FIA_UAU.7)

2.4.3.1 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.

The ST indicates that the TOE does not echo passwords as they are entered. There are no instructions necessary in the Admin Guide to cause this TOE behavior.

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.

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

2.4.4.1 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.

Section 6.1 (FIA_UAU_EXT.2) in ST indicates that the TOE can be configured to utilize local password authentication or SSH public key authentication..

See also FIA_UIA_EXT.1

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

See FIA_UIA_EXT.1

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

See FIA_UIA_EXT.1

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

2.4.5.1 NDcPP22e:FIA_UIA_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.5.2 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 6.1 (FIA_UIA_EXT.1) of ST explains that users can connect to the TOE via a local console or remotely using SSHv2 (both FMC and NGIPS), or HTTPS (FMC only). In each case, the user is required to log in prior to successfully establishing a session through which TOE security functions can be performed. By default, the Cisco NGIPS System uses internal authentication to check user credentials when a user logs in.

Section 6.1 (FIA_UAU_EXT.2) in ST indicates that the TOE can be configured to utilize local password authentication or SSH public key authentication..

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.

The TOE does not require preparatory configuration to limit services available prior to login by administrators. Login is available only to administrators with existing accounts. Section 4.3 of the Admin Guide describes the steps necessary to allow a specific user account to login using the SSH public key authentication mechanism. All other login is described in section 4.1.1, "Login to Web Interface", section 4.1.2, "Login to CLI Remotely" and section 4.1.3, "Login to CLI Locally."

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 TOE offers the following user interfaces where authentication is provided.

NGIPSv:
• Logon locally to the TOE with passwords.
• Logon to the TOE via SSH with passwords.
• Logon using SSH with public/private key pairs.

FMC:
• Logon locally to the TOE with passwords.
• Logon to the TOE via SSH with passwords.
• Logon using SSH with public/private key pairs.

• Logon to the TOE via the Web UI with passwords

For each of these interfaces, the evaluator conducted testing which showed successful and failed login attempts, and protected authentication feedback.

Test 2: After TOE configuration, the evaluator performed an nmap scan of the TOE looking for additional network services offered by the TOE that were not identified in the Security Target. The scan results did not show any ports open that offered services that were not identified in the ST.

Test 3: The evaluator successfully determined that the TOE's only service prior to login at the local console is that the TOE displays the TOE login banner.

Test 4: The evaluator determined that each of the TOE's components (FTD and FMC) require authentication before TSF actions can be performed. Each component of the TOE is managed by an FMC device and requires the administrator to successfully log into the FMC using valid credentials to gain access to any TOE services. The FMC 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.

2.4.6 X.509 Certificate Validation (NDcPP22e:FIA_X509_EXT.1/ITT)

2.4.6.1 NDcPP22e:FIA_X509_EXT.1.1/ITT

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. It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the device. The evaluator shall perform the following tests for FIA_X509_EXT.1.1/ITT:

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 TOE certificate and revocation of the TOE 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. No testing is required if no revocation method is selected. 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)

These tests were performed on all TOE components.

Test 1a/b: For this test, the evaluator configured the TOE with an invalid CA and verified that the cert chain could not be validated. The successful validation of the cert chain with trusted CA certificates was demonstrated in FCS_TLSC_EXT.1 test 1.

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 TLSC TOE client to the test server and verified that the connection succeeded only if there are no expired certificates.

Test 3: Not applicable. The TOE does not support revocation checking for distributed TOE communication.

Test 4: Not applicable. The TOE does not support revocation checking for distributed TOE communication.

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 then attempted to connect the TLSC TOE client to the test server and verified that the connection succeeded only if the certificate is not modified/corrupted.

Test 6: This test has been performed as part of Test 5.

Test 7: This test has been performed as part of Test 5.

Test 8: Not applicable. The TOE does not support EC certificates in the ITT channel.

2.4.6.2 NDcPP22e:FIA_X509_EXT.1.2/ITT

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/ITT. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/ITT. 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 two certificates: a self-signed root 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 one CA 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).

Test 1: For this test, the evaluator alternately configured a test server to send an authentication certificate issued by a Sub CA with no BasicConstraints and with BasicConstraints but the CA Flag set to false. In each case, the evaluator then attempted to connect the TLSC TOE client to the test server and verified that the connection was rejected in each case.

Test 2: The test was performed as part of Test 1.

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure it 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). If selected, the TSS shall describe how certificate revocation checking is performed. It is not sufficient to verify the status of a X.509 certificate only when it's loaded onto the device.

Section 6.1 (FIA_X509_EXT.1/ITT) in the ST states that the validity check for the certificates takes place at session establishment and/or at time of import depending on the certificate type. For example, server certificate is checked at session establishment while CA certificate is checked at both. The TOE conforms to standard RFC 5280 for certificate and path validation (i.e., peer certificate checked for expiration, peer certificate checked if signed by a trusted CA in the trust chain, peer certificate checked for unauthorized modification, peer certificate checked for revocation). The TOE does not use certificate revocation checking for communication between TOE components.

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 describe how certificate revocation checking is performed.

Section 4.9.3 in the Admin Guide states that when the initial TLS connection is established the device and FMC will authenticate each other using the registration key, and will each generate and exchange new X.509v3 certificates. Those certificates will be used for authentication of all subsequent TLS connections between the device and the FMC. The FMC and NGIPS generate unique certificates with distinct UUIDs at install and as part of the registration process. Those certificates are only used for communications between NGIPS and FMC, are not configurable, and do not provide interoperability with non-Firepower devices.

The ST indicates that no revocation checking is performed for Inter-TOE Transfers.

Component Testing Assurance Activities: None Defined

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

2.4.7.1 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. This test was executed using CRL for all components.

Test 4: For this test, the evaluator alternately configured a test server to send an authentication certificate 1) that is valid, 2) issued by an intermediate CA referring to a CRL revocation server where the signer lacks cRLSign, and 3) issued by an intermediate CA whose issuer CA refers to a CRL revocation server where the signer lacks cRLSign. In each case, the evaluator then attempted to connect the TOE to the test server and confirmed that the connection only succeeded if all retrieved CRLs were signed using certificates with cRLSign. This test was executed using CRL for all components.

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 and c: The FMC and NGIPSv do not claim support for ECDSA ciphersuites in the syslog channel, so these tests are not applicable.

2.4.7.2 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 the TOE with a certificate that did not contain the basicConstraints extension. The evaluator then attempted to make a connection between the peer devices. The connection attempt failed.

Test 2: The evaluator configured the TOE with a certificate that had the cA flag in the basicConstraints extension set to FALSE. The evaluator then attempted to make a connection between the peer devices. The connection attempt failed.

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

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

Section 6.1 (FIA_X509_EXT.1/REV) in the ST states that the validity check for the certificates takes place at session establishment and/or at time of import depending on the certificate type. For example, the server certificate is checked at session establishment while CA certificate is checked at both. The TOE conforms to standard RFC 5280 for certificate and path validation (i.e., peer certificate checked for expiration, peer certificate checked if signed by a trusted CA in the trust chain, peer certificate checked for unauthorized modification, peer certificate checked for revocation). Checking is also done for the 'basicConstraints' extension and the 'cA' flag to determine whether they are present and set to TRUE. If they are not, the CA certificate is not accepted as a trust anchor.

Section 6.1 also states that the TOE conforms to standard RFC 5280 for certificate and path validation. The TOE ensures that the peer certificate is checked for expiration, the peer certificate is checked if it is signed by a trusted CA in the trust chain, the peer certificate is checked for unauthorized modification, and the peer certificate is checked for revocation. NGIPS and FMC each maintains a local cache of CRL files using an administratively configured list of CRL distribution points. NGIPS automatically downloads new CRL files daily at a pre-set time, and FMC automatically downloads new CRL files according to an administratively configured schedule (e.g. hourly, daily, or weekly at an administrator-specified time).

Revocation checking is only applicable to FTP_ITC.1 and FIA_X509_EXT.1/Rev. The TOE does not use certificate revocation checking for communication between TOE components.

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.

The TOE supports the extendedKeyUsage fields defined for NDcPP22e:FIA_X509_EXT.1/REV, therefore, there are no rules for exceptions. Section 4.4 explains that to verify the certificate status, configure the system to load one of more certificate revocation lists (CRLs). The system compares the server certificate against those listed in the CRLs. If a server offers a certificate that is listed in a CRL as a revoked certificate, the connection fails.

This section also indicates that an audit log connection fails if the audit server certificate that does not meet either one of the following criteria:

- The certificate is not signed by the CA with cA flag set to TRUE.
- The certificate is not signed by a trusted CA in the certificate chain.
- The certificate Common Name (CN) or Subject Alternative Name (SAN) does not match the expected hostname (i.e., reference identifier).
- The certificate has been revoked or modified.

Component Testing Assurance Activities: None Defined

2.4.8 X.509 Certificate Authentication (NDcPP22e:FIA_X509_EXT.2)

2.4.8.1 NDcPP22e:FIA_X509_EXT.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.8.2 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 6.1, Table 19 (FIA_X509_EXT.2) states that the administrators can configure a trust chain by importing the CA certificate(s) that signed and issued the server (syslog) certificate. This will tell the TOE which CA certificate(s) to use during the validation process. When the TOE cannot establish a connection for the validity check (e.g., CRL checking), the trusted channel is established. Revocation checking (e.g. CRL checking) is not performed for the TLS connection between these TOE components.

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

Section 4.4 "Configure Secure Connection with Audit Server" in the Admin Guide provides instructions for the administrator that outline the steps to configure a syslog server and to import certificates to identify the TOE and the syslog server. This includes instructions for the administrator to generate a Certificate Signing Request (CSR) and import the client certificate to the TOE. It also instructs the administrator to configure the system to load one or more certificate revocation lists (CRLs) in order to verify certificate status. The connection will fail if the server certificate does not meet either one of the following criteria:

- The certificate is not signed by the CA with cA flag set to TRUE.
- The certificate is not signed by a trusted CA in the certificate chain.
- The certificate Subject Alternative Name (SAN) does not match the expected hostname (i.e., reference identifier).
- The certificate has been revoked or modified.

The TOE audit server certificate must be signed by a CA certificate that is in the TOE's audit client certificate chain of trust.

Section 4.9.4, "Custom Web Server Certificate" in the Admin Guide provides instructions for the administrator to generate a certificate request through the local configuration HTTPS Certificate page. The subsection "Importing HTTPS Server Certificate" provides instructions for importing the HTTPS certificate.

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.

Test: 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.

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

2.4.9.1 NDcPP22e:FIA_X509_EXT.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.4.9.2 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.

The ST does not select ‘device-specific information’.

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 4.4 of the Admin Guide explains how to generate a Certificate Signing Request (CSR) on the TOE. The section describes how to add the Country, Organization, Organization Unit, and Common name to the CSR as claimed in the ST.

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.

2.5 Security management (FMT)

2.5.1 Management of security functions behaviour (NDcPP22e:FMT_MOF.1/ManualUpdate)

2.5.1.1 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 6.1 (FMT_MOF.1/ManualUpdate) of the ST states that the TOE provides a web-based GUI (using HTTPS) management interface and CLI or shell (using SSH or serial connection) for all TOE administration, including the policy rule sets, user accounts and roles, and audit functions. The administrator can use the CLI to view, configure, and troubleshoot the NGIPS systems. Note that the CLI contains only a subset of all available functions and is only available on the Sensor. The web-based GUI is available on both the FMC and Sensor with the exception of the virtual Sensor. The web-based GUI on the FMC is highly recommended for daily management of the FMC and its managed Sensors. Only Authorized Administrators can perform updates to all devices through the FMC.

The TOE restricts the access to manage TSF data that can affect the security functions of the TOE to Security Administrators (i.e., administrator roles). The TSF data here includes user accounts and roles, login banner, inactivity timeout values, password complexity setting, TOE updates, audit records, and audit server information.

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 4.9.10 in the Admin Guide provides instructions for manually updating the TOE components (FMC, NGIPSv). It states that the FMC must be updated before the managed devices (ie. NGIPSv). It also provides warnings regarding capabilities that might be affected when an administrator installs an update from a managed device. The following capabilities may be affected: Traffic inspection and connection logging, Traffic flow including switching, routing, and related functionality and Link state.

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 - There is no interface to perform an update prior to authentication. The interfaces to perform an update are only available to administrators that have successfully authenticated to the TOE. The evaluator logged in as a user without administrator privileges and confirmed that there are no operations available to perform an update.

Test 2 - A successful update was performed as part of FPT_TUD_EXT.1

2.5.2 Management of TSF Data (NDcPP22e:FMT_MTD.1/CoreData)

2.5.2.1 NDcPP22e:FMT_MTD.1.1/CoreData

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for each administrative function identified in the guidance documentation; those that are accessible through an interface prior to administrator log-in are identified. For each of these functions, the evaluator shall also confirm that the TSS details how the ability to manipulate the TSF data through these interfaces is disallowed for non-administrative users.

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

Section 6.1 (FMT_MTD.1/CoreData) states The TOE provides a web-based GUI (using HTTPS) management interface and CLI or shell (using SSH or serial connection) for all TOE administration, including the policy rule sets, user accounts and roles, and audit functions. The ability to manage various security attributes, system parameters and all TSF data is controlled and limited to those users who have been assigned the appropriate administrative role and privileges associated with those roles. Note that all users created are TOE administrators.

Section 6.1 (FIA_UIA_EXT.1) of the ST states the TOE requires all users to be identified and authenticated successfully before allowing access to the TOE security function (this includes administering X509v3 certificates). The only action allowed before is viewing the login banner.

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.

See TSS Assurance Activities.

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

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

2.5.3.1 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 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.

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.

All security management functions for each TOE component are identified and described in the TSS Assurance Activities throughout this AAR with the requirement to which they apply. The evaluator confirmed that all relevant aspects of each TOE component are covered by the FMT SFRs.

Section 6.1 (FMT_MTD.1/CryptoKeys) in the ST states that the TOE only provides the ability for authorized administrators to access TOE data, such as audit data, configuration data, trust store, routing tables, and session thresholds.

Section 6.1 (FCS_CKM.4) in the ST states that the TOE is designed to zeroize secret and private keys when they are no longer required by the TOE. The table in section 7.2 identifies the applicable secret and private keys and summarizes how they are deleted.

Section 9 “Annex B: SFR TOE Components Mapping” in the ST provides a mapping of the distributed TOE components to the SFRs in this ST.

Component Guidance Assurance Activities: For distributed TOEs see chapter 2.4.1.2.

For distributed TOEs it is required to verify the Guidance Documentation to describe management of each TOE component. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

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.

The TOE is distributed. All available interfaces to manage the TOE are described in the AGD section 4.1.1, "Login to Web Interface", section 4.1.2, "Login to CLI Remotely" and section 4.1.3, "Login to CLI Locally."

All security management functions and the corresponding configuration information for each TOE component are identified in the Guidance Assurance Activities throughout this AAR with the requirement to which they apply. The evaluator confirmed that all relevant aspects of each TOE component are covered by the FMT SFRs.

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 attempted to modify the certificate store while logged in as a user with no administrative rights and verified that the attempts failed. The evaluator successfully modified the certificate store while logged in as an administrator in FIA_X509_EXT.3

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

2.5.4.1 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.

See the other requirements in this AAR as referenced. All security management functions and the corresponding configuration information are identified or referenced throughout this AAR with the requirement to which they apply.

Section 6.1 of the ST provides the TSS which details all TOE security management functions available through the TOE interfaces. The TOE is compliant with all requirements in the ST as identified in this report.

All TOE security functions as implemented in response to the requirements of the cPP are identified in the Guide and have been tested as documented throughout this AAR.

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.

2.5.5 Specification of Management Functions (IPS) (IPS10:FMT_SMF.1/IPS)

2.5.5.1 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.

Section 6.1 (FMT_SMF.1/IPS) in the ST states that the Administrators can deploy intrusion policy with intrusion rules to any interface. An interface, however, can only have one policy applied to that interface. The Administrators can also import vendor-defined signatures from Cisco, create their own intrusion rules, create rules to define which traffic is inspected and analyzed, enable anomaly rules/detections, modify thresholds and threshold duration, and configure known-good/known-bad. The IPS Analysts (Intrusion Admins) can create, modify, or delete intrusion policies but only the IPS Administrators can deploy the policies.

For additional details, see IPS_ABD_EXT.1, IPS_IPB_EXT.1 and IPS_ABD_EXT.1.

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 Admin Guide contains information for configuring the functions in the ST defined for FMT_SMT.1/IPS which are as follows.

- Section 4.7.3 "Intrusion Rule States" describes how to enable, disable signatures applied to sensor interfaces, and determine the behavior of IPS functionality
- Section 4.5, "Configure Access Control Policy" describes how to modify these parameters that define the network traffic to be collected and analysed:
  - 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
- Section 4.7.5, "Intrusion Rules Editor" describes operations upon signatures including:
  - Update (import) signatures
  - Create custom signatures
- Section 4.8.2, "Configure Anomaly Detection" describes how to configure anomaly detection
- Section 4.7.3 "Intrusion Rule States" describes how to enable and disable actions to be taken when signature or anomaly matches are detected
- Section 4.7.4, "Adding and Modifying Intrusion Event Thresholds" describes how to modify thresholds that trigger IPS reactions
- Section 4.7.7, "Configure Dynamic Rule State" describes how to modify the duration of traffic blocking actions
- Section 4.6, "Configure Security Intelligence ", describes how to:
  - Modify the known-good and known-bad lists (of IP addresses or address ranges)
  - Configure the known-good and known-bad lists to override signature-based IPS policies

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 IPS10: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 3 (ICMPv4 string). Test results show IPS_SBD_EXT.1.2-t1 Part 3 failing because the blocking of traffic did not occur.

Test 3: This test was performed as part of Test 1.

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

2.5.6.1 NDcPP22e:FMT_SMR.2.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.6.2 NDcPP22e:FMT_SMR.2.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.5.6.3 NDcPP22e:FMT_SMR.2.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the TOE supported roles and any restrictions of the roles involving administration of the TOE.

Section 6.1 (FMT_SMR.2) in the ST states that the TOE includes one evaluated role which corresponds to the required 'Security Administrator' role described in the requirement.

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 the Guidance Assurance Activities for NDcPP22E: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 FMC and FMCv, and SSH for the NGIPSv. The evaluator also tested these communication channels by applying the FCS_SSHS and FCS_TLSS tests to the relevant channels on the FMC/FMCv and NGIPSv.

2.6 Protection of the TSF (FPT)

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

2.6.1.1 NDcPP22e:FPT_APW_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.1.2 NDcPP22e:FPT_APW_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details all authentication data that are subject to this requirement, and the method used to obscure the plaintext password data when stored. The TSS shall also detail passwords are stored in such a way that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note.

Section 6.1 (FPT_APW_EXT.1) in the ST states none of the administrative interfaces on these TOE components allow administrators to view administrative passwords in plaintext form. When viewing account configuration details the password field obscures with dots any existing password or any new password being entered into the field.

Section 7.2 of the ST further describes that passwords are stored hashed using SHA-512 with a 32-bit salt value. Only 'root' user account with access to the shell can view the hashed passwords and this is prohibited in the evaluated configuration.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

2.6.2 Basic internal TSF data transfer protection - per TD0639 (NDcPP22e:FPT_ITT.1)

2.6.2.1 NDcPP22e:FPT_ITT.1.1

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.

The evaluator shall examine the TSS to determine that, for all communications between components of a distributed TOE, each communications mechanism is identified in terms of the allowed protocols for that IT entity.

The evaluator shall also confirm that all protocols listed in the TSS for these inter-component communications are specified and included in the requirements in the ST.

Section 6.1 (FPT_ITT.1) in the ST states that the communication between the TOE components is protected by TLSv1.2. TLS provides authentication, key exchange, encryption and integrity protection of all data transmitted between the TOE components. As noted throughout this AAR, the evaluator has confirmed that all protocols listed in the TSS are specified and included in the requirements in the ST.

Component Guidance 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.

The evaluator shall confirm that the guidance documentation contains instructions for establishing the relevant allowed communication channels and protocols between each pair of authorized TOE components, and that it contains recovery instructions should a connection be unintentionally broken.

Section 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST. These same ciphersuites are utilized for communication between TOE components, no further configuration of protocol details is available.

Section 4.9.3 of the Admin Guide describes the process for registering an FMC device to manage a NGIPSv Device. The material covers the steps an administrator must perform on both the sensor and on the FMC device. This section also indicates that if during device registration there’s an interruption to the TLS connection between the FMC and the device being registered the registration will fail, and will be automatically reattempted when connectivity is resumed.

Component Testing 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.

The evaluator shall perform the following tests:

a) Test 1: The evaluator shall ensure that communications using each protocol between each pair of authorized TOE components 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 with an authorized IT entity, the channel data is not sent in plaintext.

c) Test3: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route between distributed components.

The evaluator shall ensure that, for each different pair of non-equivalent component types, the connection is physically interrupted for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration that is shorter than the application layer timeout but is of sufficient length to interrupt the network link layer.

The evaluator shall ensure that when physical connectivity is restored, either communications are appropriately protected, or the secure channel is terminated and the registration process (as described in the FTP_TRP.1/Join) re-initiated, with the TOE generating adequate warnings to alert the Security Administrator.

In the case that 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 components. 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.

Test 1: This test was performed as part of Test 3 below where testing demonstrated that the TLS handshake completed successfully and application data is seen between the FMC and the managed device.

Test 2: This test was performed as part of Test 3 below where testing demonstrated that the channel data is TLS encrypted and not sent in plaintext.

Test 3: The evaluator first noted that the session establishment between the FMC and FTD/NGIPSv was successful because the TLS handshake completed and application data was passed between the two devices. The application data is encrypted, so no channel data was sent in plaintext.

In the MAC layer timeout test, the evaluator disconnected the connection. The managed device still attempts to pass data to the FMC, but the packets are lost. After approximately 10-20 seconds, the evaluator reconnected the cable between the two devices and observed the continuation of application data between the devices. The evaluator successfully verified that the communications were adequately protected after coming back online. In the App layer timeout test, the evaluator disconnected the connection. About 15 minutes later, the evaluator reconnected the cable and observed that a new TLS handshake was initiated. The channel data was protected after the disconnect.

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

2.6.3.1 NDcPP22e:FPT_SKP_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details how any pre-shared keys, symmetric keys, and private keys are stored and that they are unable to be viewed through an interface designed specifically for that purpose, as outlined in the application note. If these values are not stored in plaintext, the TSS shall describe how they are protected/obscured.

Section 6.1 (FPT_SKP_EXT.1) in the ST states the TOE components (NGIPS and FMC) are designed to not to disclose or store plaintext keys (e.g., pre-shared keys are never recorded in the audit records or displayed during any authentication process). Pre-shared keys are stored in plaintext and not visible via the FMC GUI or in any configuration file even by accounts that have full administrative access such as the default 'admin' account. Only the 'root' account would be able to view pre-shared keys, and use of the root account to access the Linux shell is prohibited in the evaluated configuration. The same is true for cryptographic keys such as encryption symmetric keys and private keys. The public keys can be viewed but cannot be modified without detection. Note that access to public keys is restricted to administrators.

Component Guidance Assurance Activities: None Defined

Component Testing Assurance Activities: None Defined

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

2.6.4.1 NDcPP22e:FPT_STM_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.4.2 NDcPP22e:FPT_STM_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it lists each security function that makes use of time, and that it provides a description of how the time is maintained and considered reliable in the context of each of the time related functions.

If 'obtain time from the underlying virtualization system' is selected, the evaluator shall examine the TSS to ensure that it identifies the VS interface the TOE uses to obtain time. If there is a delay between updates to the time on the VS and updating the time on the TOE, the TSS shall identify the maximum possible delay.

Section 6.1 (FPT_STM_EXT.1) in the ST states that The TOE is a hardware appliance that includes a hardware-based real-time clock or uses a virtual real-time clock for the virtual appliances. The TOE's embedded OS manages the clock and the GUI exposes the clock management function to the administrators. The time source is updated frequently from the time server to ensure accuracy. The time is used for the timestamp in the audit records and events. The FMC/FMCv clock can only be managed manually as it does not connect to NTP in the evaluated configuration. The NGIPS/NGIPSv can sync with its FMC/FMCv using NTP via a trusted channel using TLS with the FMC/FMCv (in the same TLS connection used for all other connections between TOE components relevant to FPT_ITT).

Component Guidance Assurance Activities: The evaluator examines the guidance documentation to ensure it instructs the administrator how to set the time. If the TOE supports the use of an NTP server, the guidance documentation instructs how a communication path is established between the TOE and the NTP server, and any configuration of the NTP client on the TOE to support this communication.

If the TOE supports obtaining time from the underlying VS, the evaluator shall verify the Guidance Documentation specifies any configuration steps necessary. If no configuration is necessary, no statement is necessary in the Guidance Documentation. If there is a delay between updates to the time on the VS and updating the time on the TOE, the evaluator shall ensure the Guidance Documentation informs the administrator of the maximum possible delay.

Section 4.9.7 of the Admin Guide describes that the time can be set manually on the FMC and NTP via the FMC on the sensors through the inter-TOE TLS communication. The steps needed to set time manually from the FMC are provided.

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 followed the guidance instructions to manually configure the time on the FMC devices. The evaluator read the time from the TOE using a date command and also found audit records confirming that the time was successfully changed. The evaluator also viewed that this manual time change was passed down to the managed NGIPSv. The syncing of the Firepower Device to the FMC ensures that all Distributed TOE components maintain the same time for audit purposes.

Test 2: The TOE does not support the use of an external NTP server and therefore this test is not applicable.

Test 3: Not applicable. The TOE does not obtain time from an underlying VS.

2.6.5 TSF testing (NDcPP22e:FPT_TST_EXT.1)

2.6.5.1 NDcPP22e:FPT_TST_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to ensure that it details the self-tests that are run by the TSF; this description should include an outline of what the tests are actually doing (e.g., rather than saying 'memory is tested', a description similar to 'memory is tested by writing a value to each memory location and reading it back to ensure it is identical to what was written' shall be used). The evaluator shall ensure that the TSS makes an argument that the tests are sufficient to demonstrate that the TSF is operating correctly.

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

Section 6.1 (FPT_TST_EXT.1) in ST describes that the NGIPSv and FMC/FMCv includes a number of built in diagnostic tests that are run during start-up to determine whether the TOE is operating properly. When CC mode is enabled, the TOE will run a HMAC-SHA512 integrity tests at power-up covering the whole kernel, all binaries and libraries, modules and boot loader of the system. If the hash verification fails, the Process Manager (PM) will not start and the system will not enter operational state. In addition, the TOE is designed to run the power-on self-tests that comply with the FIPS 140-2 requirements for self-test (e.g., known answer tests (KATs) and zeroization tests). If the TOE fails any of the FIPS power-on self-tests, the TOE will enter an error state and will not be operational. The following self-tests are executed: AES encryption/decryption KAT, RSA key generation and encryption/decryption KAT, SHA hash KATs, HMAC-SHA hash KATs, PRNG KATs, and key overwriting tests. All parts of the distributed TOE perform the same start-up tests. The NGIPSv and FMCv also performs the same self-tests as the physical FMC. Thus, all components of the TOE run tests (verifying the integrity of the software image prior to loading it, and verifying the correct operation of cryptographic operations prior to the TOE becoming operational) are sufficient to demonstrate that the TSF is operating correctly. Each cryptographic module includes self tests demonstrating the correct operation of the cryptographic operations it can perform. When crypto modules are the same on different components, the cryptographic tests are the same. The approach to integrity testing is the same on the different components of the TOE, however the integrity values may differ.

Component Guidance Assurance Activities: The evaluator shall also ensure that the guidance documentation describes the possible errors that may result from such tests, and actions the administrator should take in response; these possible errors shall correspond to those described in the TSS.

For distributed TOEs the evaluator shall ensure that the guidance documentation describes how to determine from an error message returned which TOE component has failed the self-test.

Section 4.3 of the Admin Guide describes the steps to configure the TOE in “CC” mode. Enabling CC mode will restrict the SSH algorithms, SSH rekey, TLS versions and TLS cipher suites (including elliptical curves) to the Approved ones claimed in the ST. There are additional features such as enabling the power-up integrity HMAC-SHA-512 self-test, enabling FIPS mode, and other TLS required checks such as the ones specified in section 6 of RFC 6125.

Section 4.9.11 of the Admin Guide indicates that Cisco 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 will not enter operational state. If this occurs, please re-boot the appliance. If the product still does not enter operational state, please contact Cisco Support (e-mail support@Cisco.com or call us at 1-800-917-4134 or 1-410-423-1901).

In the CC-certified configuration, the Firepower Management Center (FMC) and its managed Firepower devices are considered 'distributed' components. If any self-test fails on the FMC the failure error will be displayed at the FMC console and the FMC will automatically reboot. If any self-test fails for a managed Firepower device that device will lose connectivity to FMC and thus will appear in FMC (under Devices > Device Management) to be in an error state. To view the details of device error states, view the health monitor (System > Health > Monitor) for a summary of errors for each device, and optionally click on the icon for any device to see more details, which should include “Process Status: All processes are running correctly.” If the error state continues for longer than required for the device to reboot, view the output at the device’s console port for any errors indicating whether any self-tests have failed.

The following possible errors that can occur during this self-test are:

Known Answer Test (KAT) failures, including AES encryption/decryption KAT, RSA key generation and encryption/decryption KAT, SHA hash KATs, HMAC-SHA hash KATs, PRNG KATs. Zeroization Test failure (key overwriting tests) Software integrity failure (HMAC-SHA512 integrity tests)

The actual output of FIPS 140-2 self-tests can only be accessed using the shell access with root permission. The status output is located in /var/log/openssl-selftest.log

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.

2.6.6 Trusted update (NDcPP22e:FPT_TUD_EXT.1)

2.6.6.1 NDcPP22e:FPT_TUD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.6.2 NDcPP22e:FPT_TUD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.6.6.3 NDcPP22e:FPT_TUD_EXT.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall verify that the TSS describe how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the TSS needs to describe how and when the inactive version becomes active. The evaluator shall verify this description.

The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system firmware and software (for simplicity the term 'software' will be used in the following although the requirements apply to firmware and software). The evaluator shall verify that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. Alternatively an approach using a published hash can be used. In this case the TSS shall detail this mechanism instead of the digital signature verification mechanism. The evaluator shall verify that the TSS describes the method by which the digital signature or published hash is verified to include how the candidate updates are obtained, the processing associated with verifying the digital signature or published hash of the update, and the actions that take place for both successful and unsuccessful signature verification or published hash verification.

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

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

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

Section 6.1 (FPT_TUD_EXT.1) in the ST explains that the current version of NGIPSv and the FMC can be queried through the FMC WebUI. For manual update, the user will download the TOE upgrade file to FMC/FMCv and the digital signature of the downloaded file will be automatically verified as soon as the download is complete. If the digital signature verification fails the file will be automatically deleted and will not be available to be installed. This ensures TOE updates are always validated prior to installation. When an administrator attempts to initiate installation of any validated update file, the system will only allow selection of TOE components (FMC/FMCv or NGIPSv) for which the upgrade version would be appropriate (e.g., the system will not allow attempting to upgrade to an older version).

Component Guidance Assurance Activities: The evaluator shall verify that the guidance documentation describes how to query the currently active version. If a trusted update can be installed on the TOE with a delayed activation, the guidance documentation needs to describe how to query the loaded but inactive version.

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

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

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

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

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

Section 4.9.10 of the Admin Guide describes the process for updating the FMC and Sensor devices. It specifies how the Product Updates page shows the version of each update, as well as the date and time it was generated, including the current version as shown in the provided screenshot. It indicates that the FMC must be updated prior to update of any sensor it manages. As upgrade files are uploaded to FMC, FMC will automatically verify the integrity of the files using digital signature to ensure they have not been modified since they were created by Cisco, and that they were properly signed by Cisco. If any upgrade file fails the automatic integrity verification the file will be automatically deleted and will not be available to install to the FMC or any managed device. Any upgrade file listed on FMC's Product Updates page has been verified and can be installed by an authorized administrator.

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

a) Test 1: The evaluator performs the version verification activity to determine the current version of the product. If a trusted update can be installed on the TOE with a delayed activation, the evaluator shall also query the most recently installed version (for this test the TOE shall be in a state where these two versions match). The evaluator obtains a legitimate update using procedures described in the guidance documentation and verifies that it is successfully installed on the TOE. For some TOEs loading the update onto the TOE and activation of the update are separate steps ('activation' could be performed e.g. by a distinct activation step or by rebooting the device). In that case the evaluator verifies after loading the update onto the TOE but before activation of the update that the current version of the product did not change but the most recently installed version has changed to the new product version. After the update, the evaluator performs the version verification activity again to verify the version correctly corresponds to that of the update and that current version of the product and most recently installed version match again.

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

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

2) An image that has not been signed

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

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

described in the guidance documentation. After the TOE has rejected the update the evaluator shall verify, that both, current version and most recently installed version, reflect the same version information as prior to the update attempt.

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

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

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

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

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

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

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

Test 1: The evaluator began by performing a valid update test on all components of the TOE. The evaluator displayed the version of the product and then attempted to install a valid update. All Firepower devices are updated through the FMC/FMCv. The signatures verified and the updates on all devices were successfully installed. The evaluator verified the version number had been updated.

Test 2: The evaluator then attempted several invalid updates on all components of the TOE. All Firepower Devices are updated through the FMC/FMCv, so the FMC/FMCv rejects the invalid updates before they can be installed on the Firepower Devices. The following invalid updates were loaded and rejected:

i. an image that has been modified with a hex editor.
ii. an image without a signature
iii. an image with a bad signature.

After the TOE rejected all the updates the evaluator verified that the version number had not changed.

Test 3: Not applicable. Published hash is not used to verify the integrity of the TOE updates.

2.7 TOE ACCESS (FTA)

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

2.7.1.1 NDCPP22E:FTA_SSL.3.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that it details the administrative remote session termination and the related inactivity time period.

Section 6.1 (FTA_SSL.3) in the ST explains that the TOE terminates remote sessions that have been inactive for an administrator-configured period of time. The TOE can be configured by an administrator to set an interactive session timeout value in the system policy or platform settings, as with the login banner. The setting applies to all users and for both local and remote interactive sessions. The timeout value can be any positive integer value from 1 minute to 1,440 minutes (24 hours), with 0 disabling the timeout (the default timeout value is 60 minutes for web UI and disabled by default for CLI).

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 4.9.9 of the Admin Guide states that by default, all user sessions (web-based and CLI) automatically log out after 60 minutes (1 hour) of inactivity, unless you are otherwise configured to be exempt from session timeout. Users with Administrator Role can change the inactivity timeout value in the system policy to meet their security needs. The section continues to describe the configuration for setting the session inactivity timeout for both web based and CLI based administration.

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 NGIPSv and FMC, 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.

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

2.7.2.1 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 6.1 (FTA_SSL.3) in ST explains that the TOE provides a logout option for users to terminate their own sessions when they choose. The ability to logout is available on the FMC Web session and FMC/NGIPSv CLI.

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

Section 4.1.4 of the Admin Guide described the method to terminate a local or remote session.

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 by issuing the "exit" command.

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 by issuing the "exit" command.

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

2.7.3.1 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 6.1 (FTA_SSL_EXT.1) in the ST states that an administrator can configure maximum inactivity times for both local and remote administrative sessions. When a session is inactive (i.e., no session input) for the 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 timeout value can be any positive integer value from 1 minute to 1,440 minutes (24 hours), with 0 disabling the timeout – the default timeout value is 60 minutes for web UI and disabled by default for CLI.

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 4.9.9 of the Admin Guide states that by default, all user sessions (web-based and CLI) automatically log out after 60 minutes (1 hour) of inactivity, unless you are otherwise configured to be exempt from session timeout. Users with Administrator Role can change the inactivity timeout value in the system policy to meet their security needs. The section continues to describe the configuration for setting the session inactivity timeout for both web based and CLI based administration.

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 NGIPSv to both 5 and 7 minutes and observed that the user was logged out automatically in 5 and 7 minutes, respectively. For the FMC, the console timeout value is actually 1 minute plus the configured value. The evaluator set the timeout to both 2 minutes and 5 minutes and observed that the user was logged out automatically in 3 minutes and 6 minutes, respectively.

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

2.7.4.1 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 6.1 (FTA_TAB.1) in the ST states that the TOE provides administrators with the capability to configure advisory banner or warning message(s) that will be displayed prior to completion of the logon process at the local console or via any remote connection (e.g., SSH or HTTPS). The TOE displays an advisory notice and a consent warning message for each administrative method of access:

• FMC/FMCv: Console, SSH, and WebUI

• NGIPS: The NGIPS CLI (SSH) and console

Component Guidance Assurance Activities: The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

Section 4.9.8 of the Admin Guide states that the Administrator can create a custom login banner that appears when users log into the appliance using SSH and on the login page of the web interface. Banners can contain any printable characters except the less-than symbol (<) and the greater-than symbol (>). The section describes how to configure the banner message.

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

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

Test 1: The evaluator followed the steps outlined in the guidance to configure a login banner, nd then as part of testing NDcPP22E:FIA_UIA_EXT.1 test 1, logged into the TOE via each possible login method to see the login banner updated and displayed properly.

2.8 Trusted path/channels (FTP)

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

2.8.1.1 NDcPP22e:FTP_ITC.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.2 NDcPP22e:FTP_ITC.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.1.3 NDcPP22e:FTP_ITC.1.3

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each secure communication mechanism is identified in terms of the allowed protocols for that IT entity, whether the TOE acts as a server or a client, and the method of assured identification of the non-TSF endpoint. The evaluator shall also confirm that all secure communication mechanisms are described in sufficient detail to allow the evaluator to match them to the cryptographic protocol Security Functional Requirements listed in the ST.

Section 6.1 (FTP_ITC.1) in ST indicates that the TOE can be configured to transmit audit records to an external audit server. In order to protect exported audit records from disclosure or modification, the TOE utilizes syslog over TLS connections. The TLS provides authentication, key exchange, encryption and integrity protection of the data. For every audit event generated, the TOE stores it locally and sends it to the audit server.

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.

The only trusted channel included in the evaluation is the TLS connection between the TOE and an external audit server (syslog server). Properly configuring this connection involves ensuring that the proper TLS ciphersuites and protocol versions are used, by enabling "UCAPL/CC" Mode and configuring the details of the connection to the syslog server.

Section 4.3 of the Admin Guide describes the steps to configure the TOE in "UCAPL/CC" Mode which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

Section 4.4 of the Admin Guide describes how to configure a secure connection with an audit server using the syslog protocol. It explains the process for configuring the TOE components with an audit client certificate (i.e., a certificate used to identify the TOE to the audit server). It also explains how to specify the external audit server using the Web UI.

Section 4.9.3 of the Admin Guide indicates that if during device registration there's an interruption to the TLS connection between the FMC and the device being registered the registration will fail, and will be automatically reattempted when connectivity is resumed.

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_TLSC_EXT.1 where the FMC and NGIPSv TLS channel were 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: These tests were repeated for the FMC TLS and NGIPSv TLS and connections with the test peer:

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 (FMC) and 10 seconds (NGIPSv), the connection was restored. After the restoration, the evaluator observed the TLS 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 15 minutes (FMC) and 30 minutes (NGIPSv), the connection was restored. After the restoration, the evaluator observed that the TLS connection had to be reestablished and data remained protected.

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

2.8.2.1 NDcPP22e:FTP_TRP.1.1/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.2 NDcPP22e:FTP_TRP.1.2/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.8.2.3 NDcPP22e:FTP_TRP.1.3/Admin

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

Component TSS Assurance Activities: The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Section 6.1 (FTP_TRP.1) in the ST indicates the TOE includes implementations of SSHv2 (by OpenSSH on NGIPS and FMC) and HTTPS (HTTP over TLS on FMC only).

Component Guidance Assurance Activities: The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Section 4.3 of the Admin Guide describes the steps to configure the TOE in “UCAPL/CC mode” which defines the set of ciphersuites, versions and protocols that the TOE supports. Once in CC mode, the TOE supports only ciphersuites, protocols and versions as defined by the ST.

Section 4.9.4 of the Admin Guide describes the process for obtaining or generating a CSR and importing a certificate to be used by the TOE on its HTTPS Web UI.

Section 4.1 of the Admin Guide describes how to log onto the TOE remotely.

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 on all devices and FCS_TLSS_EXT.1 for HTTPS on the FMC devices.

2.9 Intrusion Prevention (IPS)

2.9.1 Anomaly-Based IPS Functionality (IPS10:IPS_ABD_EXT.1)

2.9.1.1 IPS10:IPS_ABD_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.1.2 IPS10:IPS_ABD_EXT.1.2

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.1.3 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 6.1 (IPS_ABD_EXT.1) in the ST describes how in an intrusion detection/prevention deployment, the TOE (NGIPS) examines packets based on network analysis policies and intrusion policies. The detection of anomalies is performed through rules configured by the administrator. The operations associated with the anomaly-based IPS policies are to allow the traffic flow for any sensor interface in any mode and allow the traffic flow and block/drop the traffic flow in inline mode. Administrators can define strings to match URLs/URIs, and web page content for pattern-matching. Several preprocessors allow administrators to detect specific threats, such as IP/TCP/UDP/ICMP portscans, ICMP/TCP flooding, DoS attacks and other rate-based attacks (“frequency”). The administrator can configure thresholds that mimic normal expected frequency and configure the TOE to detect and drop events exceeding the configured thresholds. (The ST notes that the TOE’s definition of the term “threshold” matches the definition of the term “frequency” in the IPS PP module, thus ‘frequency’ rather than ‘threshold’ is selected in the IPS_ABD_EXT.1.1 requirement).

When the system identifies a possible intrusion, it generates an intrusion or preprocessor event. Managed Sensors transmit their events to the FMC, where the administrators can view the aggregated data and gain a greater understanding of the attacks against their network assets. In an inline deployment, managed Sensors can also drop or replace packets that are known to be harmful. The packet decoder, the preprocessors, and the intrusion rules engine can all cause the TOE to generate an event. The TOE can also be configured to use intrusion rules to detect various attacks such as Teardrop, Bonk, Ping of Death, etc. The administrators can use pre-defined anomaly-based rules or can create custom rules to detect these and many other attacks.

Section 6.1 (IPS_ABD_EXT.1) in the ST states that the administrator can configure the Sensor in either a passive or inline deployment. In a passive (promiscuous) IPS deployment, the Sensor monitors traffic flowing across a network using a switch SPAN or mirror port. The SPAN or mirror port allows for traffic to be copied from other ports on the switch. This provides the system visibility within the network without being in the flow of network traffic. When configured in a passive deployment, the system cannot take certain actions such as blocking or shaping traffic. The administrator can configure one or more physical ports (Gigabit ethernet interfaces) on a managed Sensor as passive interfaces and deploy the intrusion policy to that interface via security zone (i.e., the interface is added to the zone). In an inline IPS deployment, the administrator configures the Sensor transparently on a network segment by binding two ports together. The administrator can configure one or more physical ports (Gigabit ethernet interfaces) on a managed Sensor as inline interfaces then assign a pair of inline interfaces to an inline set. The intrusion policy is then deployed to that inline set via security zone. The management interface (typically eth0) is separate from the other data monitoring interfaces (used as passive or inline) on the Sensor. It is used to set up and register the Sensor to the FMC.

Section 7.1 in the ST indicates that the TOE supports alert rules, pass rules and drop rules. The drop rule is available when operating in "inline mode".

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 4.8.2, "Configure Anomaly Detection" of the Admin Guide describes how to create and modify policies with anomaly-based rules to detect, drop and log applicable traffic. These policies can be associated with access policies (as described by Section 4.5) which apply to distinct network interfaces.

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.

2.9.2 IP Blocking (IPS10:IPS_IPB_EXT.1)

2.9.2.1 IPS10:IPS_IPB_EXT.1.1

TSS Assurance Activities: None Defined

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.2.2 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 6.1 (IPS_IPB_EXT.1 [IPS]) in the ST states before traffic can be inspected by intrusion policies, the TOE enforces any defined Security Intelligence (SI) policy, which consists of one or more known-good entries in a ”Do-Not-Block List” and/or one or more known-bad entries in a “Block List”. An IP address in a Do-Not-Block List or Block List matches a packet if the IP address in the list matches either the source address or the destination address in the packet. Entries in the Block List take precedence over entries in the Do-Not-Block List, so if the same IP address appears in both lists, the action for the Block List is applied. Matching a Do-Not-Block List entry results in skipping further IPS inspection, and is generally used to avoid false-positive IPS alerts for known-good addresses. Matching a Block List entry results in dropping the packet with no further IPS inspection, and is generally used to minimize impacts (alerts, and use of system resources) related to IPS inspection for traffic to/from known malicious entities. IPS Policy elements like URLs and domain names can also be configured like IP addresses described above.

The IPS_IPB_EXT.1.2 requirement indicates that IPS Administrators, Intrusion Admin and Access Admin can configure the IPS policy elements. Section 6.1 (FMT_SMF.1/IPS) in the ST identifies and describes the roles and level of access for each as follows:

- "IPS Administrator" (or Administrator): Have all privileges and access
- "IPS Analyst" (or Intrusion Admin): Have all access to intrusion policies, IPS policies and network analysis privileges but cannot deploy policies
- Access Admin: Have all access to access control policies but cannot deploy policies
- Discovery Admin: Have all access to network discovery, application detection, and correlation features but cannot deploy policies
- Security Analyst: Have all access to security event analysis feature

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 4.6, "Configure Security Intelligence" of the Admin Guide describes the steps an administrator or Access Admin takes to configure a Do-Not-Block List or Block List. Also, Section 4.5.1 of the Admin Guide describes the steps to create and modify Access Control Policies. Access Control Policies allow Administrators to define rules to map intrusion rules to types of network traffic.

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: The evaluator used the AGD to create an access control list that includes a block list. This consisted of network objects that contained IP addresses and ranges of IP addresses to be blocked. At the same time an intrusion policy was created to allow all traffic. Because block-lists and do-not-block lists are enforced prior to intrusion policies, the packets that matched the configured block-list were blocked when they would normally be allowed by the intrusion policy. That resulted in the following behavior:

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 DST addresses: Pass -- packet not allowed as expected.
Attempt to send traffic using a SRC address blocked by a Known-Bad list rule specifying multiple SRC addresses: Pass -- packet not allowed as expected.

Test 2: For this test, the evaluator created a do-not-block list with address rules of various types, using both single addresses and a range of addresses. At the same time an intrusion policy was created to block all traffic. Because block-lists and do-not-block lists are enforced prior to intrusion policies, the packets that matched the configured do-not-block-list were allowed when they would normally be blocked by the intrusion policy. That resulted in the following behavior:

Variation
Attempt to send traffic using a SRC address allowed by a Known-Good list rule specifying only a single SRC address: Pass -- packet allowed as expected.
Attempt to send traffic using a DST address allowed by a Known-Good list rule specifying only a single DST address: Pass -- packet allowed as expected.
Attempt to send traffic using a DST address allowed by a Known-Good list rule specifying multiple DST addresses:

Pass -- packet allowed as expected.
Attempt to send traffic using a SRC address allowed by a Known-Good list rule specifying multiple SRC addresses: Pass -- packet allowed as expected.

Test 3: For this test, the evaluator configured both the block lists and do-not-block lists from test 1 and test 2 at the same time. As a result all packets that overlap both lists should be blocked as the block list takes priority. This is seen in the follow test parts as in each case the packet sent was blocked as expected.

Variation
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 not allowed as expected.
Send traffic using a DST address that is in both a Known-Good list rule and in a Known-Bad list rule: Pass -- packet not allowed as expected.
Send traffic using a SRC address in a Known-Good list rule and in a multiple address Known-Bad list: Pass -- packet not allowed as expected.
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.
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 not allowed as expected.
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.
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 not allowed as expected.
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 not allowed as expected.
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 not allowed as expected.
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 not allowed as expected.
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.
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 not allowed as expected.
Send traffic using a 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.
Send traffic using a 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 not allowed as expected.

2.9.3 Network Traffic Analysis (IPS10:IPS_NTA_EXT.1)

2.9.3.1 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 FAU_GEN.1/IPS in the ST describes the Access Control Policy which is associated with the intrusion policy where the intrusion, preprocessor, or decoder rule that generates an event is enabled. Access control rules invoke the intrusion policy such that all traffic permitted by the access control policy is then inspected by the designated intrusion policy.

Section 6.1 (IPS_NTA_EXT.1) in the ST indicates that network analysis policies (anomaly-based rules) govern the decoding and preprocessing of traffic after traffic is first filtered by Security Intelligence (the known-bad list takes precedence over the known-good list) and before the traffic is inspected by access control rules and intrusion policies (signature-based rules). A network analysis policy governs packet processing in phases. First the system decodes packets through the first three TCP/IP layers, then continues with normalizing, preprocessing, and detecting protocol anomalies.

- The packet decoder converts packet headers and payloads into a format that can be easily used by the preprocessors and later, intrusion rules. Each layer of the TCP/IP stack is decoded in turn, beginning with the data link layer and continuing through the network and transport layers. The packet decoder also detects various anomalous behaviors in packet headers.
- The inline normalization preprocessor reformats (i.e., normalizes) traffic to minimize the chances of attackers evading detection. It prepares packets for examination by other preprocessors and intrusion rules, and helps ensure that the packets the system processes are the same as the packets received by the hosts on your network.
- Various network and transport layers preprocessors detect attacks that exploit IP fragmentation, perform checksum validation, and perform TCP and UDP session preprocessing.
- Various application-layer protocol decoders normalize specific types of packet data into formats that the intrusion rules engine can analyze. Normalizing application-layer protocol encodings allows the system to effectively apply the same content-related intrusion rules to packets whose data is represented differently, and to obtain meaningful results.
- The Modbus and DNP3 SCADA preprocessors detect traffic anomalies and provide data to intrusion rules. The baselines are provided by the preprocessors and detection of anomalies through rules configured by the administrator.
- Several preprocessors allow administrators to detect specific threats, such as IP/TCP/UDP/ICMP portscans, ICMP/TCP flooding, DoS attacks and other rate-based attacks (“frequency”). The administrator can configure threshold that mimics normal expected frequency and configure the TOE to detect and drop events exceeding the configured thresholds. (The ST notes that the TOE’s definition of the term “threshold” matches the definition of the term “frequency” in the IPS PP module, thus ‘frequency’ rather than ‘threshold’ is selected in the IPS_ABD_EXT.1.1 requirement).
- The TOE can perform network analysis and deploy intrusion policies to any data monitoring interface as described above. The policy hierarchy order is not configurable and follows this order: Security Intelligence (the known-bad list takes precedence over the known-good list), anomaly-based rules, then signature-based rules. Conformance with protocols identified throughout the discussion of IPS is demonstrated by protocol compliance testing by the vendor.

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 4.6 of the Admin Guide indicates that the policy hierarchy order is not configurable and follows this order: Security Intelligence (Block List takes precedence over Do-Not-Block List), anomaly-based rules, and then signature-based rules.

Testing Assurance Activities: None Defined

2.9.3.2 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 6.1 (IPS_NTA_EXT.1) in the ST describes how the network analysis policy governs packet processing. First the system decodes packets through the first three TCP/IP layers, then continues normalizing, preprocessing and detecting protocol anomalies.

- The packet decoder converts packet headers and payloads into a format that can be easily used by the preprocessors and later, intrusion rules. Each layer of the TCP/IP stack is decoded in turn, beginning with the data link layer and continuing through the network and transport layers. The packet decoder also detects various anomalous behaviors in packet headers.
- The inline normalization preprocessor reformats (i.e., normalizes) traffic to minimize the chances of attackers evading detection. It prepares packets for examination by other preprocessors and intrusion rules, and helps ensure that the packets the system processes are the same as the packets received by the hosts on your network.

- Various network and transport layers preprocessors detect attacks that exploit IP fragmentation, perform checksum validation, and perform TCP and UDP session preprocessing.
- Various application-layer protocol decoders normalize specific types of packet data into formats that the intrusion rules engine can analyze. Normalizing application-layer protocol encodings allows the system to effectively apply the same content-related intrusion rules to packets whose data is represented differently, and to obtain meaningful results. Conformance to protocols has been verified via compliance testing.
- The Modbus and DNP3 SCADA preprocessors detect traffic anomalies and provide data to intrusion rules. The baselines are provided by the preprocessors and detection of anomalies through rules configured by the administrator. The operations associated with the anomaly-based IPS policies are allow the traffic flow for any sensor interface in any mode and allow the traffic flow and block/drop the traffic flow in inline mode. Administrators can define strings to match URLs/URIs, and web page content for pattern matching.
- Several preprocessors allow administrators to detect specific threats, such as IP/TCP/UDP/ICMP portscans, ICMP/TCP flooding, DoS attacks and other rate-based attacks (“frequency”). The administrator can configure threshold that mimics normal expected frequency and configure the TOE to detect and drop events exceeding the configured thresholds. (The ST notes that the TOE’s definition of the term “threshold” matches the definition of the term “frequency” in the IPS PP module, thus ‘frequency’ rather than ‘threshold’ is selected in the IPS_ABD_EXT.1.1 requirement).

Section 7.1.1 in the ST indicates that Intrusion rules can specify protocols: ICMPv4, ICMPv6, IPv4, IPv6, TCP and UDP.

Guidance Assurance Activities: None Defined

Testing Assurance Activities: None Defined

2.9.3.3 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 6.1 (IPS_NTA_EXT.1) states that the administrator can configure the Sensor in either a passive or inline deployment. In a passive (promiscuous) IPS deployment, the Sensor monitors traffic flowing across a network using a switch SPAN or mirror port. The SPAN or mirror port allows for traffic to be copied from other ports on the switch. This provides the system visibility within the network without being in the flow of network traffic. When configured in a passive deployment, the system cannot take certain actions such as blocking or shaping traffic. The administrator can configure one or more physical ports (Gigabit ethernet interfaces) on a managed Sensor as passive interfaces and deploy the intrusion policy to that interface via security zone (i.e., the interface is added to the zone). In an inline IPS deployment, the administrator configures the Sensor transparently on a network segment by binding two ports together. The administrator can configure one or more physical ports (Gigabit ethernet interfaces) on a managed Sensor as inline interfaces then assign a pair of inline interfaces to an inline set. The intrusion policy is then deployed to that inline set via security zone. The management interface (typically eth0) is separate from the other data monitoring interfaces (used as passive or inline) on the Sensor. It is used to set up and register the Sensor to the FMC.

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 4.8.6 of the Admin Guide describes the two approaches for deploying a sensor device. In a passive deployment, the system is deployed out of band from the flow of network traffic. In an inline deployment, the system is configured transparently on a network segment by binding two ports together. Section 4.7.3, which describes the state of an Intrusion Rule, addresses the applicability of the state to a particular deployment type. That is, the “Drop and Generate Event” state of a rule will cause the rule to generate an event, but will NOT drop the packet when in a passive deployment.

Section 2.1 of the Admin Guide indicates that the management network should be physically or logically separated (e.g., VLANs) from the monitored network.

Section 4.5.2 of the Admin Guide describes how to activate Intrusion Policies on the Firepower Devices’ interfaces.

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

2.9.4 Signature-Based IPS Functionality - per TD0722 (IPS10:IPS_SBD_EXT.1)

2.9.4.1 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 in the ST states that an intrusion rule is a set of keywords and arguments that the system uses to detect attempts to exploit vulnerabilities on your network. As the system analyzes network traffic, it compares packets against the conditions specified in each rule. If the packet data matches all the conditions specified in a rule, the rule triggers. If a rule is an alert rule, it generates an intrusion event. If it is a pass rule, it ignores the traffic. For a drop rule in an inline deployment, the system drops the packet and generates an event.

All rules contain two logical sections: the rule header and the rule options. The rule header contains:

- the rule's action or type
- the protocol
- the source and destination IP addresses and netmasks
- direction indicators showing the flow of traffic from source to destination
- the source and destination ports

The rule options section contains:

- event messages
- keywords and their parameters and arguments
- patterns that a packet's payload must match to trigger the rule
- specifications of which parts of the packet the rules engine should inspect

Section 6.1 (IPS_SBD_EXT.1) in the ST states that an administrator can configure one or more physical ports (Gigabit ethernet interfaces) on a managed Sensor as passive interfaces and deploy the intrusion policy to that interface via security zone (i.e., the interface is added to the zone). The administrator can configure one or more physical ports (Gigabit ethernet interfaces) on a managed Sensor as inline interfaces then assign a pair of inline interfaces to an inline set. The intrusion policy is then deployed to that inline set via security zone. The management interface (typically eth0) is separate from the other data monitoring interfaces (used as passive or inline) on the Sensor. It is used to set up and register the Sensor to the FMC.

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 4.7.5 in the Admin Guide describes an intrusion rule as a set of keywords and arguments that the system uses to detect attempts to exploit vulnerabilities on your network. It also describes the structure, keywords and arguments for intrusion rules. An intrusion rule can match against any value within the packet. Section 4.7.3, describes the how individual built-in intrusion rules can be enabled or disabled. This section describes that a rule state can be set to one of the following values:

- Generate Events
- Drop and Generate Events
- Disable

The following table, also found in Section 4.7.5 of the Admin Guide, maps the Intrusion Rule keywords to the fields identified by the Assurance Activity. All fields are mapped to some rule keyword, argument, or built-in rule.

1 Required Header Field Inspection	2 Intrusion Rule Keyword or Rule
3 IPv4:	4

5 Version	*6* alert (msg:"DECODE_NOT_IPV4_DGRAM"; sid:1; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
7 Header Length	8 alert (msg:"DECODE_IPV4_INVALID_HEADER_LEN"; sid:2; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
9 Packet Length	10 alert (msg:"DECODE_IPV4_DGRAM_LT_IPHDR"; sid:3; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;) alert (msg:"DECODE_IPV4_DGRAM_GT_CAPLEN"; sid:6; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
11 ID	*12 id*
13 IP Flags	*14 fragbits*
15 Fragment Offset	*16 fragoffset*
17 Time to Live (TTL)	*18 ttl*
19 Protocol	*~~20~~ ip_proto*
21 Header Checksum	*22* Inspected by “Checksum Verification” preprocessor.
23 Source Address	*24 Source IP* 25 OR *26* alert (msg:"DECODE_IP4_SRC_MULTICAST"; sid:410; gid:116; rev:1; metadata:rule-type decode; classtype:misc-activity;)
27 Destination Address	*28 Destination IP* 29 OR 30 alert (msg:"DECODE_IP4_DST_RESERVED"; sid:412; gid:116; rev:1; metadata:rule-type decode; classtype:misc-activity;)
31 IP Options.	*32 ipopts*

33 IPv6:	34
35 Version	~~36~~ alert (msg:"DECODE_IPV6_IS_NOT"; sid:271; gid:116; rev:1; metadata:rule-type decode; classtype: protocol-command-decode;)
37 payload length	*38 dsize* 39 OR *40* alert (msg:"DECODE_IPV6_TRUNCATED_EXT"; sid:272; gid:116; rev:1; metadata:rule-type decode; classtype:bad-unknown;)
41 next header	*42* alert (msg:"DECODE_IPV6_BAD_NEXT_HEADER"; sid:281; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
43 hop limit	*44* alert (msg:"DECODE_IPV6_MIN_TTL"; sid:270; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
45 source address	*46 Source IP* 47 OR *48* alert (msg:"DECODE_IPV6_SRC_MULTICAST"; sid:277; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
49 destination address	*50 Destination IP* 51 OR 52 alert (msg:"DECODE_IPV6_DST_RESERVED_MULTICAST"; sid:278; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;) 53 *54* alert (msg:"DECODE_IPV6_DST_ZERO"; sid:276; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
55 routing header	*56* alert (msg:"DECODE_IPV6_ROUTE_AND_HOPBYHOP"; sid:282; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)

	alert (msg:"DECODE_IPV6_TWO_ROUTE_HEADERS"; sid:283; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
57 ICMP:	*58*
59 Type	*60 itype*
61 Code	*62 icode*
63 Header Checksum	*64* Inspected by “Checksum Verification” preprocessor.
65 Rest of Header(varies based on the ICMP type and code)	*66 icmp_id, icmp_seq*
67 ICMPv6:	*68*
69 Type	*70 itype*
71 Code	*72 icode*
73 Header Checksum	*74* Inspected by “Checksum Verification” preprocessor.
75 TCP:	*76*
77 source port	*78 Source Port*
79 destination port	*80 Destination Port*
81 sequence number	*82 seq*
83 acknowledgement number	*84 ack*
85 offset	86 alert (msg:"DECODE_TCP_INVALID_OFFSET"; sid:46; gid:116; rev:1; metadata:rule-type decode; reference:cve,2004-0816; classtype:bad-unknown;)
87 reserved	*88* Inspected and normalized by preprocessor, if configured.
89 TCP flags	*90 flags*
91 window	*92 window*
93 checksum	*94* Inspected by “Checksum Verification” preprocessor.

95 urgent pointer	96 alert (msg:"DECODE_TCP_BAD_URP"; sid:419; gid:116; rev:1; metadata:rule-type decode; classtype: misc-activity;) 97 OR *98* Inspected and normalized by preprocessor, if configured.
99 TCP options	*100* alert (msg:"DECODE_TCPOPT_TRUNCATED"; sid:55; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
101 UDP:	*102*
103 Source port	*104 Source Port*
105 destination port	*106 Destination Port*
107 length;	*108* alert (msg:"DECODE_UDP_DGRAM_INVALID_LENGTH"; sid:96; gid:116; rev:1; metadata:rule-type decode; classtype:protocol-command-decode;)
109 UDP checksum	*110* Inspected by “Checksum Verification” preprocessor.

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.

2.9.4.2 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 6.1 (IPS_SBD_EXT.1) in the ST states that the operations associated with the anomaly-based IPS policies are allow the traffic flow for any sensor interface in any mode and allow the traffic flow and block/drop the traffic flow in inline mode. Administrators can define strings to match URLs/URIs, and web page content for pattern-matching.

Section 7.1.2 in the ST describes the rule options that can be used to match information in a packet for a given rule (where every rule has an action of pass, drop or alert). One option, the content keyword, specifies a data pattern inside a packet. The pattern may be presented in the form of an ASCII string or as binary data in the form of hexadecimal characters.

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 4.7.1 in the Admin Guide describes how to create an intrusion policy that can be assigned different intrusion rules.

Section 4.7.5 in the Admin Guide describes an intrusion rule as a set of keywords and arguments that the system uses to detect attempts to exploit vulnerabilities on your network. It also describes the structure, keywords and arguments for intrusion rules. An intrusion rule can match against any value within the packet. The content keyword can be used to specify data pattern inside a packet. The pattern may be presented in the form of an ASCII string or as binary data in the form of hexadecimal characters. This approach is used to detect the various strings required to meet the IPS_SBD_EXT.1.2 requirement.

Section 4.5.2 of the Admin Guide describes how to activate Intrusion Policies on the Firepower Devices' interfaces.

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 evaluator repeated the UDP string matching test sending 40 packets instead of just one. The TOE detected and blocked all 40 packets.

2.9.4.3 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 6.1 (IPS_SBD_EXT.1) in the ST explains that the TOE can be configured to use intrusion rules to detect various attacks such as Teardrop, Bonk, Ping of Death, etc. The administrators can use pre-defined rules or create custom rules to detect these attacks and many more. The ‘action’ for a given rule defines the reaction when a rule detecting these attacks is triggered.

Section 7.1.1 in the ST describes the ‘action (alert)’ option which generates an intrusion event when triggered and operations (allow, block/drop) associated with policies.

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 4.8.4 in the Admin Guide includes a description of specific attacks that several pre-defined rules address. The following table, also found in Section 4.8.4, identifies the attacks from IPS_SBD_EXT.1.3 and the Rule that applies to that attack.

• Attack	• Attack Category	• Rule
• Teardrop	• IP Attack	• Rule 123:2 “FRAG2_TEARDROP”
• Bonk	• IP Attack	• Rule 123:4 “FRAG3_ANOMALY_OVERSIZE”
• Boink	• IP Attack	• Rule 123:4 “FRAG3_ANOMALY_OVERSIZE”
• Land	• IP Attack	• Rule 116:151 “DECODE_BAD_TRAFFIC_SAME_SRCDST”
• Nuke	• ICMP Attack	• alert tcp $EXTERNAL_NET any -> $HOME_NET 135:139 (msg:"SERVER-OTHER Winnuke attack";

		flow:stateless; flags:U+; metadata:ruleset community; reference:bugtraq,2010; reference:cve,1999-0153; classtype:attempted-dos; sid:1257000; rev:15; gid:1001;)
• Ping of Death	• ICMP Attack	• Rule 123:7 “FRAG3_ANOMALY_BADSIZE_LG”
• Null flags	• TCP Attack	• alert tcp $EXTERNAL_NET any -> $HOME_NET any (msg:"Null TCP attack"; flags:0; classtype:attempted-dos; sid:269; rev:3;)
• SYN+FIN flags	• TCP Attack	• Rule 116:420 “DECODE_TCP_SYN_FIN”
• FIN only flags	• TCP Attack	• alert tcp any any -> any any (sid:1000003; gid:1; flags:F*; msg:"FIN only"; classtype:attempted-dos; rev:3;)
• SYN+RST flags	• TCP Attack	• Rule 116:421 “DECODE_TCP_SYN_RST”
• Bomb	• UDP Attack	• Rule 116:98 “DECODE_UDP_DGRAM_LONG_PACKET”
• Chargen DoS	• UDP Attack	• Rule 1:271 “SERVER-OTHER echo+chargen bomb”

The default behavior for each rule is determined by which “Base Policy” is used to create a custom-defined Intrusion Policy that’s deployed to a Firepower device. The available pre-configured Base Policies are: No Rules Active; Connectivity Over Security; Balanced Security and Connectivity; Security Over Connectivity; and Maximum Detection. Regardless of which Base Policy is used to create the Intrusion Policy that’s deployed to the Firepower device, each Base Policy contains the same set of default rules and the behavior of each rule can be modified as desired by editing any Intrusion Policy.

Section 4.7.3, “Intrusion Rule States” in the Admin Guide describes the 3 states which a rule can be in, as “Generate Events”, “Drop and Generate Events” or “Disable” which controls the manner in which the rule is applied. These correspond to the reactions from IPS_SBD_EXT.1.5.

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 syslogs that each attack was detected and dropped or logged depending on which mode the TOE was configured in.

2.9.4.4 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 6.1 (IPS_SBD_EXT.1) in the ST indicates that during network analysis, the TOE can detect specific threats, such as IP/TCP/UDP/ICMP portscans, ICMP/TCP flooding, DoS attacks and other rate based attacks ("frequency"). The attacks specified here include all of those attacks identified by IPS_SBD_EXT.1.4.

When the system identifies a possible intrusion, it generates an intrusion or preprocessor event (sometimes collectively called intrusion events). Managed Sensors transmit their events to the Firepower Management Center (FMC), where the administrators can view the aggregated data and gain a greater understanding of the attacks against their network assets. In an inline deployment, managed Sensors can also drop or replace packets that are known to be harmful.

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 4.8.4, "Rate-based Attack Prevention" of the Admin Guide describes how to define Network Analysis policies and to enable "Rate-Based Attack Prevention" and cause the appropriate reaction from those identified by IPS_SBD_EXT.1.5. This addresses the attacks to flood a network or host that are included in IPS_SBD_EXT.1.4 and listed below. Section 4.8.3, "Portscan Detection" of the Admin Guide contains instructions to configure rules to cause the system to react to Protocol and port scanning attacks. The following list enumerates the attacks identified by IPS_SBD_EXT.1.4.

1) Flooding a host (DoS attack)
a) ICMP flooding (Smurf attack, and ping flood)
b) TCP flooding (e.g. SYN flood)
2) Flooding a network (DoS attack)
3) Protocol and port scanning
a) IP protocol scanning
b) TCP port scanning
c) UDP port scanning
d) ICMP scanning

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.

2.9.4.5 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.

2.9.4.6 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 4.7.5 of 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

Section 4.8.7 of the Admin Guide provides instructions for configuring the NGIPSv interfaces for either passive or inline deployment.

Section 4.8 “Stateful Session Behaviors” of the ST describes the “IP Defragmentation Preprocessor” and that it must be enabled to detect payload across multiple packets. The following section 4.8.1 describes the configuration to enable this preprocessor.

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

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 only configuration of cryptographic functions is to place the TOE into “UCAPL/CC mode”, which is described in section 4.3 of the Admin Guide.

Section 4.9.10 of the Admin Guide describes the process for updating the FMC and Sensor devices. This section indicates that the administrator must verify updates using a published hash to determine if the update is valid. This section also indicates that after an update, the “Device Management” screen should show the new version of software installed on the sensor, in order to determine whether the update process was successful.

Section 1.1 of the Admin Guide defines the scope of evaluation, and states that any features not associated with SFRs in the claimed NDcPP and extended package were not evaluated.

3.2.2 Preparative Procedures (AGD_PRE.1)

Assurance Activities: As with the operational guidance, the developer should look to the Evaluation Activities to determine the required content with respect to preparative procedures.

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO_CPC_EXT.1 and FTP_TRP.1(2)/Join.

The evaluator performs the CEM work units associated with the AGD_PRE.1 SAR. Specific requirements and EAs on the preparative documentation are identified (and where relevant are captured in the Guidance Documentation portions of the EAs) in the individual EAs for each SFR.

Preparative procedures are distributed to administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

In addition, the evaluator performs the EAs specified below.

The evaluator shall examine the Preparative procedures to ensure they include a description of how the administrator verifies that the operational environment can fulfil its role to support the security functionality (including the requirements of the Security Objectives for the Operational Environment specified in the Security Target).

The documentation should be in an informal style and should be written with sufficient detail and explanation that they can be understood and used by the target audience (which will typically include IT staff who have general IT experience but not necessarily experience with the TOE product itself).

The evaluator shall examine the Preparative procedures to ensure they are provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

The evaluator shall examine the preparative procedures to ensure they include instructions to successfully install the TSF in each Operational Environment.

The evaluator shall examine the preparative procedures to ensure they include instructions to manage the security of the TSF as a product and as a component of the larger operational environment.

In addition the evaluator shall ensure that the following requirements are also met.

The preparative procedures must

a) include instructions to provide a protected administrative capability; and

b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

The evaluator confirmed that the material in the Admin Guide could be used to configure the TOE. The completeness of the manuals 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.

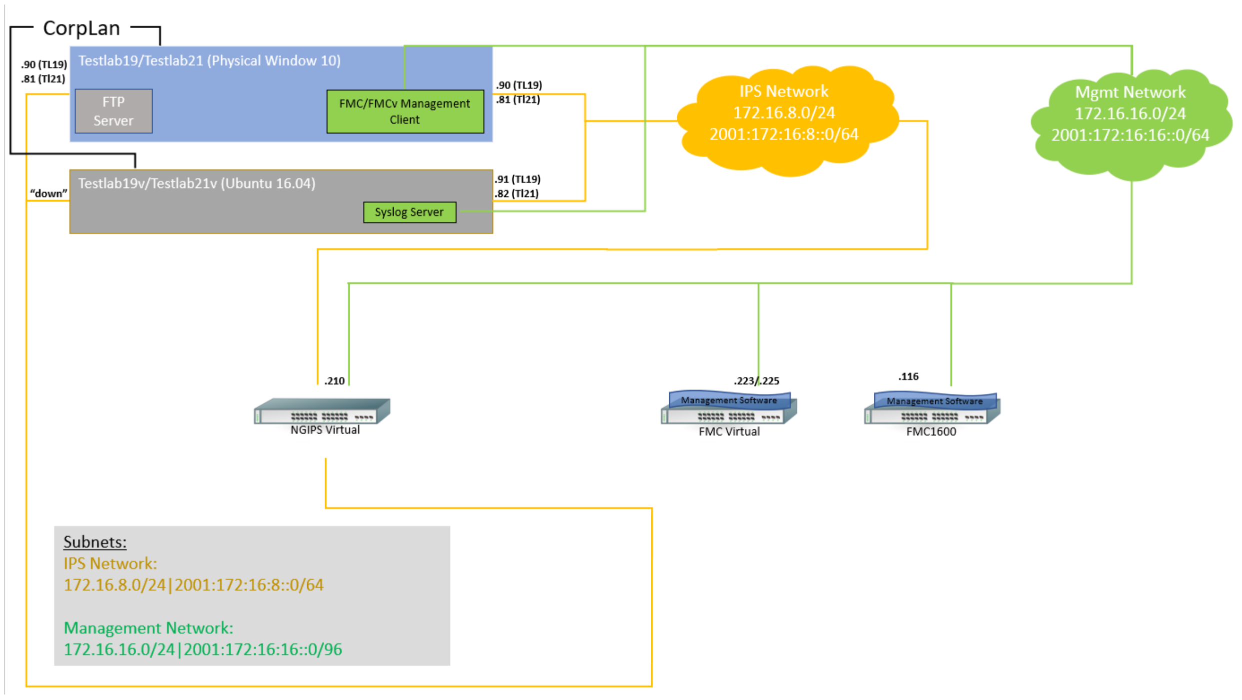
The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

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:

- Cisco FMC1600 running FMC v7.0

- Cisco FMCv running FMC v7.0 in a VM provided by ESXi 7.0 running on a Cisco UCSC-C220-M5
- Cisco NGIPSv running NGIPS software v7.0 in a VM provided by ESXi 7.0 running on a Cisco UCSC-C220-M5

Supporting Software:

- Windows 10.0
- Wireshark version 2.6.6
- Windows SSH Client – Putty version 0.68 (used to connect to device console and SSH)

The Gossamer Test servers with an Ubuntu environment acted as platforms to initiate testing. The test servers 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 5.2.5
- 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 5/17/2023 with the following search terms: "Cisco FirePOWER", "FMC1600-K9", "FMCv 7.0", "NGIPSv", "VMware v7.0", "UCSC-C220-M5", "Intel Xeon Scalable", "Intel Xeon Skylake", "Intel Xeon Silver 4110", "Intel Xeon Silver 4116", "Intel Xeon E5-2600 v3", "Intel Xeon E5-2600 v4", "Intel Xeon D", "linux-yocto-4.18.45", "ESXi-7.0", "OpenSSH 7.6p1", "OpenSSL", "CiscoSSL FOM 7.3sp", "syslog-ng 3.3.2", "auditd", "MySQL 15.1".

3.5.2 Additional Flaw Hypotheses (AVA_VLA.1)

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

The TOE utilizes TLS ciphersuites that contain the RSA algorithm in the key exchange portion of the TLS handshake. The evaluator used testssl.sh (https://testssl.sh/bleichenbacher/), which is a third-party developed script that tests TLS servers for the ROBOT vulnerability. The evaluator concluded that none of the TOE's TLS servers are vulnerable to the ROBOT attack.