



---

[www.GossamerSec.com](http://www.GossamerSec.com)

# ASSURANCE ACTIVITY REPORT FOR IBM QRADAR SECURITY INTELLIGENCE PLATFORM VERSION 7.5

---

Version 0.4  
June 26, 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



## REVISION HISTORY

Revision	Date	Authors	Summary
Version 0.1	May 10, 2023	Gossamer Security Solutions	Initial draft
0.2	June 8, 2023	Gossamer	Resolve validation comments
0.3	June 20, 2023	Gossamer	Resolve validation comments
0.4	June 26, 2023	Gos	Resolve validation comments

### The TOE Evaluation was Sponsored by:

IBM, Corporation  
1 New Orchard Road  
Armonk, New York 10504

### Evaluation Personnel:

- John Messiha
- Cornelius Haley

### 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 .....6
  - 1.1 REFERENCES** .....6
  - 1.2 CAVP Certificate Justification .....6
- 2. Protection Profile SFR Assurance Activities .....9
  - 2.1 Security audit (FAU) .....9
    - 2.1.1 Audit Data Generation (NDcPP22e:FAU\_GEN.1) .....9
    - 2.1.2 User identity association (NDcPP22e:FAU\_GEN.2) .....10
    - 2.1.3 Protected Audit Event Storage (NDcPP22e:FAU\_STG\_EXT.1) .....11
  - 2.2 Cryptographic support (FCS) .....15
    - 2.2.1 Cryptographic Key Generation (NDcPP22e:FCS\_CKM.1) .....15
    - 2.2.2 Cryptographic Key Establishment (NDcPP22e:FCS\_CKM.2) .....18
    - 2.2.3 Cryptographic Key Destruction (NDcPP22e:FCS\_CKM.4) .....21
    - 2.2.4 Cryptographic Operation (AES Data Encryption/Decryption) (NDcPP22e:FCS\_COP.1/DataEncryption) 23
    - 2.2.5 Cryptographic Operation (Hash Algorithm) (NDcPP22e:FCS\_COP.1/Hash) .....27
    - 2.2.6 Cryptographic Operation (Keyed Hash Algorithm) (NDcPP22e:FCS\_COP.1/KeyedHash) .....29
    - 2.2.7 Cryptographic Operation (Signature Generation and Verification) (NDcPP22e:FCS\_COP.1/SigGen) ...30
    - 2.2.8 HTTPS Protocol (NDcPP22e:FCS\_HTTPS\_EXT.1) .....32
    - 2.2.9 Random Bit Generation (NDcPP22e:FCS\_RBG\_EXT.1) .....33
    - 2.2.10 SSH Server Protocol - per TD0631 (NDcPP22e:FCS\_SSHS\_EXT.1) .....34
    - 2.2.11 TLS Client Protocol Without Mutual Authentication - per TD0634 & TD0670 (NDcPP22e:FCS\_TLSC\_EXT.1) .....42
    - 2.2.12 TLS Client Support for Mutual Authentication - per TD0670 (NDcPP22e:FCS\_TLSC\_EXT.2) .....51
    - 2.2.13 TLS Server Protocol Without Mutual Authentication - per TD0635 (NDcPP22e:FCS\_TLSS\_EXT.1) ..52
  - 2.3 Identification and authentication (FIA) .....58
    - 2.3.1 Authentication Failure Management (NDcPP22e:FIA\_AFL.1) .....58
    - 2.3.2 Password Management (NDcPP22e:FIA\_PMG\_EXT.1) .....60
    - 2.3.3 Protected Authentication Feedback (NDcPP22e:FIA\_UAU.7) .....62
    - 2.3.4 Password-based Authentication Mechanism (NDcPP22e:FIA\_UAU\_EXT.2) .....62
    - 2.3.5 User Identification and Authentication (NDcPP22e:FIA\_UIA\_EXT.1) .....63



- 2.3.6 X.509 Certificate Validation (NDcPP22e:FIA\_X509\_EXT.1/Rev).....65
- 2.3.7 X.509 Certificate Authentication (NDcPP22e:FIA\_X509\_EXT.2) .....70
- 2.3.8 X.509 Certificate Requests (NDcPP22e:FIA\_X509\_EXT.3).....72
- 2.4 Security management (FMT).....73
  - 2.4.1 Management of security functions behaviour (NDcPP22e:FMT\_MOF.1/ManualUpdate).....73
  - 2.4.2 Management of TSF Data (NDcPP22e:FMT\_MTD.1/CoreData).....74
  - 2.4.3 Management of TSF Data (NDcPP22e:FMT\_MTD.1/CryptoKeys).....75
  - 2.4.4 Specification of Management Functions - per TD0631 (NDcPP22e:FMT\_SMF.1) .....76
  - 2.4.5 Restrictions on Security Roles (NDcPP22e:FMT\_SMR.2) .....78
- 2.5 Protection of the TSF (FPT) .....79
  - 2.5.1 Protection of Administrator Passwords (NDcPP22e:FPT\_APW\_EXT.1) .....79
  - 2.5.2 Protection of TSF Data (for reading of all pre-shared, symmetric and private keys) (NDcPP22e:FPT\_SKP\_EXT.1) .....80
  - 2.5.3 Reliable Time Stamps - per TD0632 (NDcPP22e:FPT\_STM\_EXT.1) .....80
  - 2.5.4 TSF testing (NDcPP22e:FPT\_TST\_EXT.1) .....82
  - 2.5.5 Trusted update (NDcPP22e:FPT\_TUD\_EXT.1).....84
- 2.6 TOE access (FTA) .....88
  - 2.6.1 TSF-initiated Termination (NDcPP22e:FTA\_SSL.3).....88
  - 2.6.2 User-initiated Termination (NDcPP22e:FTA\_SSL.4).....89
  - 2.6.3 TSF-initiated Session Locking (NDcPP22e:FTA\_SSL\_EXT.1).....90
  - 2.6.4 Default TOE Access Banners (NDcPP22e:FTA\_TAB.1).....91
- 2.7 Trusted path/channels (FTP).....92
  - 2.7.1 Inter-TSF trusted channel - per TD0639 (NDcPP22e:FTP\_ITC.1).....92
  - 2.7.2 Trusted Path - per TD0639 (NDcPP22e:FTP\_TRP.1/Admin).....94
- 3. Protection Profile SAR Assurance Activities .....97
  - 3.1 Development (ADV) .....97
    - 3.1.1 Basic Functional Specification (ADV\_FSP.1).....97
  - 3.2 Guidance documents (AGD).....98
    - 3.2.1 Operational User Guidance (AGD\_OPE.1) .....98
    - 3.2.2 Preparative Procedures (AGD\_PRE.1).....99
  - 3.3 Life-cycle support (ALC).....100



- 3.3.1 Labelling of the TOE (ALC\_CMC.1) .....100
- 3.3.2 TOE CM Coverage (ALC\_CMS.1).....101
- 3.4 Tests (ATE).....101
  - 3.4.1 Independent Testing - Conformance (ATE\_IND.1).....101
- 3.5 Vulnerability assessment (AVA) .....103
  - 3.5.1 Vulnerability Survey (AVA\_VAN.1).....103



## 1. INTRODUCTION

This document presents evaluations results of the IBM QRadar Security Intelligence Platform Version 7.5 NDcPP22e evaluation. This document contains a description of the assurance activities and associated results as performed by the evaluators.

### 1.1 REFERENCES

The following evidence was used to complete the Assurance Activities:

- [ST] IBM QRadar Security Intelligence Platform Version 7.5 Security Target
- [CC-Guide] IBM QRadar Version 7.5 Common Criteria for NIAP Revision 1.0

### 1.2 CAVP CERTIFICATE JUSTIFICATION

The Target of Evaluation (TOE) is QRadar Security Intelligence Platform version 7.5. IBM Security QRadar Security Intelligence Platform is also known as the IBM QRadar Security Information and Event Management (SIEM). The evaluated product is a single All-in-one device running QRadar SIEM device. The single model of the TOE that was evaluated is the QRadar All-In-One: Dell 3128C, model that utilizes an Intel® Xeon® Gold 5118 CPU without PAA and operating system support from Red Hat Enterprise Linux server release 7.9.

The TOE utilizes cryptographic support from RedHat’s OpenSSL (1.0.2k-FIPS) and IBM FIPS JCE (version 0.8.4) library. Both are multi-algorithm libraries providing general-purpose cryptographic services. The purpose of the OpenSSL library is to provide an API for cryptographic functionality for system services. The purpose of the IBM FIPS JCE library is to provide only CAVP verified algorithms for use in Java based processes within the TOE. The TOE utilizes these libraries for cryptographic operations it performs (including those associated with TLS, certificate operations, system integrity, and update verification). The libraries used by the TOE have obtained CAVP certificates as shown in the following tables.

Functions	Standards	SFR	CAVP Certificate
<b>Encryption/Decryption</b>			
AES CBC (128 and 256 bits) AES GCM (128 bits)	ISO 18033-3 ISO 10116 ISO 19772 FIPS Pub 197 NIST SP 800-38A	FCS_COP.1/DataEncryption	A2425
<b>Cryptographic hashing</b>			
SHA-256/512	FIPS Pub 180-4 ISO/IEC 10118-3:2004	FCS_COP.1/Hash	A2425
<b>Keyed-hash message authentication</b>			
HMAC_SHA-256, HMAC-SHA-512 (digest sizes and block sizes of 256, 384 and 512 bits)	FIPS Pub 198-1 FIPS Pub 180-4 ISO/IEC 9797-2:2011	FCS_COP.1/KeyedHash	A2425
<b>Cryptographic Signature Services</b>			
RSA 2048-bit	FIPS Pub 186-4	FCS_COP.1/SigGen	A2425



Signature Gen & Verify w/ SHA-256 and SHA-512	ISO/IEC 9796-2		
<b>Random bit generation</b>			
HASH_DRBG	FIPS SP 800-90B ISO/IEC 18031:2011	FCS_RBG_EXT.1	A2425
<b>Cryptographic Key Generation</b>			
RSA 2048-bit Key Gen	FIPS Pub 186-4 ISO/IEC 9796-2	FCS_CKM.1	A2425
ECDSA P-256, P-384, P-521 Key Gen	FIPS Pub 186-4	FCS_CKM.1	A2425
<b>Key Establishment</b>			
RSA	RSAES-PKCS1-v1_5	FCS_CKM.2	Vendor Affirmed
KAS ECC	FIPS Pub 800-56A Rev 3	FCS_CKM.2	A2425
FFC Schemes using safe-prime groups Diffie-Hellman Group 14	NIST SP 800-56A Rev 3	FCS_CKM.2	Vendor Affirmed.

**Table 1 CAVP Certificates for IBMFIPSJCE library**

The selections made in each SFR identified by the “SFR” column in **Table 1** and **Table 2**, correspond to the Functions and Standards shown by these same tables. For further analysis of the CAVP certificate refer to the ETR.

Functions	Standards	SFR	CAVP Certificate
<b>Encryption/Decryption</b>			
AES CBC (128 and 256 bits) AES GCM (128 bits)	ISO 18033-3 ISO 10116 ISO 19772 FIPS Pub 197 NIST SP 800-38A	FCS_COP.1/DataEncryption	A2426
<b>Cryptographic hashing</b>			
SHA-256/512	FIPS Pub 180-4 ISO/IEC 10118-3:2004	FCS_COP.1/Hash	A2426
<b>Keyed-hash message authentication</b>			
HMAC_SHA-256, HMAC-SHA-512 (digest sizes and block sizes of 256, 384 and 512 bits)	FIPS Pub 198-1 FIPS Pub 180-4 ISO/IEC 9797-2:2011	FCS_COP.1/KeyedHash	A2426
<b>Cryptographic Signature Services</b>			
RSA 2048-bit Signature Gen & Verify w/ SHA-256 and SHA-512	FIPS Pub 186-4 ISO/IEC 9796-2	FCS_COP.1/SigGen	A2426
<b>Random bit generation</b>			
CTR_DRBG (AES)	FIPS SP 800-90B ISO/IEC 18031:2011	FCS_RBG_EXT.1	A2426
<b>Cryptographic Key Generation</b>			
RSA 2048-bit Key Gen	FIPS Pub 186-4 ISO/IEC 9796-2	FCS_CKM.1	A2426
ECDSA P-256, P-384, P-521 Key Gen	FIPS PUB 186-4	FCS_CKM.1	A2426
<b>Key Establishment</b>			
RSA	RSAES-PKCS1-v1_5	FCS_CKM.2	Vendor Affirmed
KAS ECC	FIPS Pub 800-56A Rev 3	FCS_CKM.2	A2426
FFC Schemes using safe-prime groups Diffie-Hellman Group 14	NIST SP 800-56A Rev 3	FCS_CKM.2	Vendor Affirmed

**Table 2 CAVP Certificates for OpenSSL library**







## 2. PROTECTION PROFILE SFR ASSURANCE ACTIVITIES

### 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 of the [ST] states the audit records generated by the TOE include a date and time, event type, success or failure indication. Audit records contain a certificate's entire domain name to identify the certificate.

**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.

The section entitled "Sample Audits" in the [CC-Guide] contains samples of audit records associated with each auditable event identified by the Security Target.

This information includes details about the audit records which the TOE generates including details encompassing the required content. During testing, the evaluator mapped the entries in the tables in this section to the TOE generated events, showing that the section provides include examples/descriptions of all required audit events.

The evaluator verified the administrative commands when performing all other guidance AA. Specific references to commands can be found throughout this AAR.

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

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

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

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

### **2.1.2 USER IDENTITY ASSOCIATION (NDCPP22E:FAU\_GEN.2)**



### 2.1.2.1 NDCPP22E:FAU\_GEN.2.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

**Component TSS Assurance Activities:** The TSS and Guidance Documentation requirements for FAU\_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU\_GEN.1.

The TSS and Guidance Documentation requirements for FAU\_GEN.2 are already covered by the TSS requirements for FAU\_GEN.1.

**Component Guidance Assurance Activities:** The TSS and Guidance Documentation requirements for FAU\_GEN.2 are already covered by the TSS and Guidance Documentation requirements for FAU\_GEN.1.

The TSS and Guidance Documentation requirements for FAU\_GEN.2 are already covered by the TSS requirements for 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 FAU\_GEN.1.1.

### 2.1.3 PROTECTED AUDIT EVENT STORAGE (NDCPP22E:FAU\_STG\_EXT.1)

#### 2.1.3.1 NDCPP22E:FAU\_STG\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined



### 2.1.3.2 NDcPP22E:FAU\_STG\_EXT.1.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.1.3.3 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 of the [ST] explains that the TOE can store audit data as well as transmit audit data to an external audit server through a trusted channel protected by TLS in real time. The external audit server can be any server supporting the transmission of audit data tunneled within TLS.

All audit logs when stored internally are stored in plain text. The authorized administrator can only view audit records stored locally. The TOE provides the administrator this ability to view audit records only through the WebUI. These logs are archived and compressed when the audit log file size reaches 200 MB. The current log file is named audit.log. When the file reaches 200 MB, the file is compressed and renamed to audit.log.1.gz. The file number increments each time that a log file is archived. QRadar stores up to 50 archived log files. The log files rotation settings are controlled by internal settings and are not available for modification through the administrative interface.

Section 6.1 of the [ST] explains that the TOE acts as a standalone network device generating audit data. Once 50 archived log files exist, the TOE overwrites previous audit records, by deleting the oldest archived log file and renumbering existing files when a new file is to be archived.

**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.

Chapter 1, the section entitled "Supported cipher suites" in the [CC-Guide] states that the TOE is configured to use the TLS protocol version 1.2 with the supported cipher suites. Chapter 1 also details limiting the cipher suites allowed in the evaluated configuration.

Chapter 6, the sections entitled "Adding forwarding destinations to QRadar" and "Configuring routing rules for event forwarding" describe how to define what level of audit log detail is forwarded and creation of filter based routing rules to forward the event data. These sections also detail the configuration options for how data is stored in the database and sent to forwarding destinations. In Online mode, the data remains current because forwarding is performed in real time. If the forwarding destination becomes unreachable, data can potentially be lost. In



Offline mode, all data is stored in the database and then sent to the forwarding destination. This assures that no data is lost, however, there might be delays in data forwarding.

Chapter 7, the section entitled "Viewing the audit logs" in the [CC-Guide] describes the behavior for audit storage and backup. Audit logs are stored in plain text and are archived and compressed when the audit log file size reaches 200 MB. The current log file is named audit.log. When the file reaches 200 MB, the file is compressed and renamed to audit.log.1.gz. The file number increments each time that a log file is archived. QRadar stores up to 50 archived log files.

Chapter 5 in the [CC-Guide] also describes configuring the TOE for exporting syslog data. It details configuring and importing root CA's and client certificates through the Certificate Management Application.

**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 evaluator configured the system (per guidance) to securely transfer audit data. The evaluator then captured network traffic between the TOE and the external audit server. The evaluator verified that the packet capture showed the audit data was not cleartext on the network.

Test 2: The evaluator also continued to generate audit data until the local storage space was exceeded. The evaluator verified that when the local audit storage was filled to the maximum, the existing audit data was overwritten based on the following rule: overwrite oldest records first.

Test 3 and 4 are not applicable.

## 2.2 CRYPTOGRAPHIC SUPPORT (FCS)

### 2.2.1 CRYPTOGRAPHIC KEY GENERATION (NDCPP22E:FCS\_CKM.1)

#### 2.2.1.1 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 6.2 of the [ST] explains the TOE generates RSA keys of size 2048-bits. The TOE also supports FFC w/ safe-primes using Diffie-Hellman key group 14 (2048 bit) key generation for SSH key establishment. The TOE supports ECC key generation as part of TLS using NIST curves P-256, P-384, P-521 which are not configurable by administrators. The TOE is a TLS client for syslog connections and a TLS and SSH server for management.

**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.

Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] describes how to limit the DH groups in the TOE to "diffie-hellman-group14-sha1" and "diffie hellman-group14-sha256".

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] describe how to limit the TLS ciphers to only those that are allowed in the evaluated configuration.





Chapter 4, the section entitled "Creating a CSR request by using a 2048-bit RSA key" in the [CC-Guide] describes generating a 2048 RSA private key file with openssl.

These sections mentioned from the [CC-Guide] list the SSH and TLS cryptographic parameters supported by the TOE. They contain steps to configure the cryptographic compliance to limit the cryptographic algorithms used by the TOE for TLS and SSH sessions to only those allowed by the Security Target. There is also a step in chapter 2, "QRadar installations for highly secure environments" that requires QRadar to be installed with FIPS mode enabled which further ensures that only the evaluated cryptographic functions are used. These steps also impact the encryption algorithms and key sizes supported by the TOE.

**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  $p_1, p_2, q_1, q_2, p$  and  $q$  shall all be provable primes
- Primes  $p_1, p_2, q_1$ , and  $q_2$  shall be provable primes and  $p$  and  $q$  shall be probable primes
- Primes  $p_1, p_2, q_1, q_2, p$  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  $x$ :

- $\text{len}(q)$  bit output of RBG where  $1 \leq x \leq q-1$
- $\text{len}(q) + 64$  bit output of RBG, followed by a mod  $q-1$  operation and a  $+1$  operation, where  $1 \leq x \leq q-1$ .

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

- $g \neq 0, 1$
- $q$  divides  $p-1$
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

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 Certificate Justification" earlier in this document. Testing for FFC Schemes using safe-prime groups is done as part of testing in NDcPP22e:FCS\_CKM.2.1.

## 2.2.2 CRYPTOGRAPHIC KEY ESTABLISHMENT (NDcPP22e:FCS\_CKM.2)

### 2.2.2.1 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)

Table 6-4 "Key Establishment Methods" in section 6.2 of the [ST] maps the key establishment schemes used by the TOE to the requirements and services offered by the TOE. This table indicates that the key establishment schemes; FFC Schemes using 'safe-prime' and RSA are used by SSH for remote administration. The ST also explains that the TOE supports ECC key establishment for TLS, which is used for remote administration (i.e., the WebUI) and for audit server communication.

**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).

Chapter 3, the sections entitled "Configuring the SSHD\_config file", "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] list the TLS ciphersuites and SSH key exchange methods that are allowed in the evaluated configuration.

**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 for ECC schemes only. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

NIST does not offer CAVP certification for RSA key establishment or FFC Safe-primers schemes. These schemes were tested against known-good 3rd party implementation to ensure the TOE is complaint with the RFC for Diffie-Hellman group 14 and RSA Keys establishment within SSH. RSA schemes are tested under FCS\_SSHS\_EXT.1 and FTP\_TRP.1/Admin. The validation of a DH Group 14 is performed under FCS\_SSHS\_EXT.1.7 Test 2. During the use of these tests, OpenSSH was leveraged to validate the TOE's implementation.

## 2.2.3 CRYPTOGRAPHIC KEY DESTRUCTION (NDcPP22E:FCS\_CKM.4)

### 2.2.3.1 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 6.2 of the [ST] explains the TOE stores all persistent secret and private keys on disk and stores all ephemeral keys in RAM. The section provides a list of keys, where they are stored, and how they are cleared. The TOE clears these keys (i.e., plaintext keys in volatile storage), by destroying the reference to the key and requesting a garbage collection action. Additionally, the TOE is designed to zeroize secret and private keys when they are no longer required by the TOE. Note that zeroization occurs as follows: 1) when deleted from disk, the previous value is overwritten once with zeroes; 2) when added or changed on disk, any old value is overwritten completely with the new value; and, 3) the zeroization of values in RAM is achieved by overwriting once with zeroes.

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

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



The TOE does not have conditions that involve delayed key destruction.

**Component Testing Assurance Activities:** None Defined

## 2.2.4 CRYPTOGRAPHIC OPERATION (AES DATA ENCRYPTION/DECRYPTION) (NDcPP22E:FCS\_COP.1/DATAENCRYPTION)

### 2.2.4.1 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 6.2 of [ST] indicates that the TOE provides symmetric encryption and decryption capabilities using AES in CBC mode (128 and 256 bit key sizes) as well as using AES in GCM mode (128 bit key sizes). AES is implemented in support of TLS and SSH protocols.

**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.

The TOE supports different AES encryption algorithms with SSH and TLS.

Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] in step 2 shows that ciphers for SSH are set to AES128-CBC and AES256-CBC.

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] describe how to limit the TLS ciphers to only those that are allowed in the evaluated configuration. See step 2 where the instructions explain how to set the ciphersuites to choose AES128-GCM and AES128-CBC.

These sections mentioned from the [CC-Guide] list the SSH and TLS cryptographic parameters supported by the TOE. They contain steps to configure the cryptographic compliance to limit the cryptographic algorithms used by the TOE for TLS and SSH sessions to only those allowed by the Security Target. There is also a step in chapter 2, "QRadar installations for highly secure environments" that requires QRadar to be installed with FIPS mode enabled which further ensures that only the evaluated cryptographic functions are used. These steps also impact the encryption algorithms and key sizes supported by the TOE.

**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  $1 < i \leq 10$ . 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  $1 < i \leq 10$ . 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<sub>i</sub> 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 key sizes 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 key size.

#### 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 key size.

There is no need to test the decryption engine.

The TOE has been CAVP tested. Refer to the section entitled, "CAVP Certificate Justification" earlier in this document.

## 2.2.5 CRYPTOGRAPHIC OPERATION (HASH ALGORITHM) (NDCPP22E:FCS\_COP.1/HASH)

### 2.2.5.1 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 6.2 of [ST] explains that the TOE supports hashing using SHA-256 and SHA-512 validated conforming to FIPS 180-4, Secure Hash Standard (SHS). SHS hashing is used within several services including, TLS, and SSH. SHA-256 is used in conjunction with RSA signatures for verification of software image integrity.

**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.

Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] describes how to specify the MAC algorithms (see step 2) where Macs can be set to `hmac-sha2-512` and `hmac-sha2-256`

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] describe how to limit the TLS ciphers to only those that are allowed in the evaluated configuration.

These sections mentioned from the [CC-Guide] list the SSH and TLS cryptographic parameters supported by the TOE. They contain steps to configure the cryptographic compliance to limit the cryptographic algorithms used by the TOE for TLS and SSH sessions to only those allowed by the Security Target. There is also a step in chapter 2, "QRadar installations for highly secure environments" that requires QRadar to be installed with FIPS mode enabled which further ensures that only the evaluated cryptographic functions are used. These steps also impact the hash sizes supported by the TOE.

**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  $i$ th message is  $m + 99*i$ , where  $1 \leq i \leq m$ . 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  $i$ th message is  $m + 8*99*i$ , where  $1 \leq i \leq m/8$ . 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 Certificate Justification" earlier in this document.

## 2.2.6 CRYPTOGRAPHIC OPERATION (KEYED HASH ALGORITHM) (NDCPP22E:FCS\_COP.1/KEYEDHASH)

### 2.2.6.1 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 6.2 of [ST] explains that the TOE supports keyed hash HMAC-SHA256, and HMAC-SHA512 conforming to ISO/IEC 9797-2:2011. Supported cryptographic key sizes: 256, 512 bits and message digest sizes: 256, 512 bits. The Keyed hash use matches validated hash algorithms implemented by the module.



**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.

Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] describes how to limit the MAC algorithms used by the TOE for SSH communication. This section indicates that the TOE supports `MacS_hmac-sha2-512`, `hmac-sha2-256`.

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] describe how to limit the TLS ciphers to only those that are allowed in the evaluated configuration. The selection of cipher suites defines the keyed-hash algorithms used by TLS.

**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 Certificate Justification" earlier in this document.

## 2.2.7 CRYPTOGRAPHIC OPERATION (SIGNATURE GENERATION AND VERIFICATION) (NDcPP22E:FCS\_COP.1/SIGGEN)

### 2.2.7.1 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.

Table 6-1 in Section 6.2 of [ST] indicates that RSA Digital Signature Algorithm with modulus of 2048 is used for cryptographic signatures.

**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.

Chapter 4, the section entitled "Creating a CSR request by using a 2048-bit RSA key" in the [CC-Guide] describes generating a 2048 RSA private key file with `openssl`.



These sections mentioned from the [CC-Guide] list the SSH and TLS cryptographic parameters supported by the TOE. They contain steps to configure the cryptographic compliance to limit the cryptographic algorithms used by the TOE for TLS and SSH sessions to only those allowed by the Security Target. There is also a step in chapter 2, "QRadar installations for highly secure environments" that requires QRadar to be installed with FIPS mode enabled which further ensures that only the evaluated cryptographic functions are used. These steps also impact the cryptographic algorithms and key sizes supported by the TOE for signature services.

**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 Certificate Justification" earlier in this document.

## 2.2.8 HTTPS PROTOCOL (NDcPP22E:FCS\_HTTPS\_EXT.1)

### 2.2.8.1 NDcPP22E:FCS\_HTTPS\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.2.8.2 NDcPP22E:FCS\_HTTPS\_EXT.1.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.2.8.3 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 6.2 of [ST] indicates that an HTTPS/TLS connection is available which presents Web GUI and Rest API administrative interfaces. The TOE implements HTTPS per RFC 2818 with TLSv1.2. A connection can be established only if the peer initiates the connection.

**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.

Chapter 2, the section entitled "Logging in to QRadar" in the [CC-Guide] states that the TOE offers a Web GUI that is protected with HTTPS. This section also explains the procedure to access the TOE's Web GUI using a web browser.

**Component Testing Assurance Activities:** This test is now performed as part of FIA\_X509\_EXT.1/Rev testing.

Tests are performed in conjunction with the TLS evaluation activities.





If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA\_X509\_EXT.1.

This test was performed as part of NDcPP22e:FTP\_TRP.1/Admin where the packet capture was inspected and showed that the TOE uses HTTP over TLS to protect administrative communications.

## 2.2.9 RANDOM BIT GENERATION (NDcPP22E:FCS\_RBG\_EXT.1)

### 2.2.9.1 NDcPP22E:FCS\_RBG\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.2.9.2 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.

Section 6.2 of [ST] identifies the TOE's DRBG as a NIST-approved AES-CTR Deterministic Random Bit Generator (DRBG), as specified in SP 800-90B. AES-256 is used in conjunction with a minimum of 256 bits of entropy accumulated from CPU jitter. The TOE also includes the IBMFIPSJCE HASH\_DRBG which is seeded by drawing random data from the JitterEntropy software noise source with at least 256-bits of entropy. 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.

**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 TOE does not offer any configuration for the RNG functionality.



**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  $\leq$  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 Certificate Justification" earlier in this document.

## **2.2.10 SSH SERVER PROTOCOL - PER TD063 1 (NDcPP22E:FCS\_SSHS\_EXT.1)**

### **2.2.10.1 NDcPP22E:FCS\_SSHS\_EXT.1.1**

**TSS Assurance Activities:** None Defined



**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.2.10.2 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 6.2 of [ST] states that the supports RSA public-key authentication for users connecting to the TOE with a public-key. The evaluator confirmed this is included in the FCS\_COP.1/SigGen selections in [ST]. Administrators must associate a public key with each user account that is authenticated with public-keys.

This section also states that the TOE SSHv2 supports both public-key and password based authentication.

**Guidance Assurance Activities:** None Defined

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

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

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

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



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

(TD0631 applied)

Test 1: The evaluator configured a user to be able to login using the SSH interface with public-key based authentication, and observed the user login was successful.

Test 2: The evaluator attempted to login using the SSH interface with public-key authentication without configuring a public key for that user and observed that the login attempt was not successful.

Test 3: The evaluator configured the TOE for password authentication on the SSH interface. The evaluator logged in using an SSH client and the correct password. The login was successful.

Test 4: The evaluator attempted an SSH connection using an invalid password. The evaluator was not able to log in.

### **2.2.10.3 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 6.2 states that packets are limited to 262127 bytes and are dropped if they exceed the limit.

**Guidance Assurance Activities:** None Defined

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

The evaluator created and sent a packet to the TOE that was larger than the maximum packet size of 262127 bytes. The TOE rejected the packet and the connection was closed.

### **2.2.10.4 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 6.2 of the [ST] states that the TOE uses the AES encryption algorithm in CBC mode with key sizes of either 128 or 256. This is consistent with the selection in the requirement. This section in [ST] also does not indicate any optional aspects of the protocol are supported.



**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).

Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] lists the host key algorithms, encryption ciphers/algorithms, key exchange methods, and MAC ciphers, supported by the TOE in the evaluated configuration. These instructions allow the administrator to configure the TOE to conform to the SSH Server requirements of [ST].

The section entitled "Configuring Java TLS ciphers" contains steps to configure the TOE to limit the cryptographic algorithms used by the TOE for TLS. Additionally, there is a step in chapter 2, "QRadar installations for highly secure environments" that requires QRadar to be installed with FIPS mode enabled which further ensures that only the evaluated cryptographic functions are used.

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

The evaluator attempted to establish an SSH connection with each of the following SSH algorithms in the SFR to encrypt the session. The evaluator captured packets associated with each of the connection attempts and observed through testing that the TOE supports the following:

- aes128-ctr
- aes256-ctr

### **2.2.10.5 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 6.2 of [ST] states that TOE allow use of ssh-rsa, rsa-sha2-256, and rsa-sha2-512 for authentication of the TOE to the client. The evaluator confirmed that FCS\_SSHS\_EXT.1.5 included the following selection: ssh-rsa, rsa-sha2-256, and rsa-sha2-512.



**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).

The section entitled "Configuring SSH to use public key authentication" in [CC-Guide] contains instructions to allow an administrator to configure the TOE SSH server to accept authentication using either public-key or passwords. This section also explains how to load a public-key for each user account that wishes to use 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)

Test 1: The evaluator attempted to establish an SSH connection with each of the following SSH public key algorithms. The evaluator captured packets associated with each of the connection attempts. The evaluator observed through testing that the TOE supports the following:

- ssh-rsa,
- rsa-sha2-256,
- rsa-sha2-512

Test 2: The evaluator attempted to establish an SSH connection using ssh-dsa. The evaluator captured packets and was able to determine the connection attempt failed as expected.

### **2.2.10.6 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 6.2 of [ST] states that the TOE supports HMAC-SHA -256, and HMAC-SHA -512 for message integrity. This list is consistent with the selection 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).

Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] describes how to limit the MAC algorithms used by the TOE for SSH communication. This section indicates that the TOE supports `macs hmac-sha2-512, hmac-sha2-256`.

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

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

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

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

Test 1: The evaluator attempted to establish an SSH connection with each of the following SSH transport MAC algorithms. The evaluator captured packets associated with each of these connection attempts. The evaluator observed through testing that the TOE supports the following:

- hmac-sha2-256,
- hmac-sha2-512.

Test 2: The evaluator attempted to connect to the TOE using HMAC-MD5. The TOE rejects the attempt as expected.

### **2.2.10.7 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 6.2 of the [ST] states that the TOE supports the following key exchange methods: diffie-hellman-group14-sha1, and diffie-hellman-group14-sha256. This list is consistent with the selection 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 key exchange algorithms are used in SSH connections with the TOE.



Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] lists the key exchange methods that are supported by the TOE SSH Server in the evaluated configuration. This section includes the following key exchanges: diffie-hellman-group14-sha1, and diffie-hellman-group14-sha256.

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

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

Test 1 - The evaluator attempted to connect to the TOE using Diffie-Hellman-Group1. The TOE rejects the attempt as expected.

Test 2 - The evaluator attempted to establish an SSH connection with each of the following key exchange methods: diffie-hellman-group14-sha1 and diffie-hellman-group14-sha256. The evaluator captured packets associated with each of these connection attempts.

### 2.2.10.8 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 6.2 states that an SSH session is rekeyed by the TOE after being established one hour or after transmitting 1Gb of traffic whichever comes first. A client initiated rekey, resets both the time and traffic counters.

**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.

Chapter 3, the section entitled "Configuring the SSHD\_config file" in the [CC-Guide] states that the SSH session rekey is hardcoded to rekey after one hour or following the exchange of 1 GB of data, whichever comes first.

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

For testing of the time-based threshold the evaluator shall use an SSH client to connect to the TOE and keep the session open until the threshold is reached. The evaluator shall verify that the SSH session has been active longer





than the threshold value and shall verify that the TOE initiated a rekey (the method of verification shall be reported by the evaluator).

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

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

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

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

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

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

- a) An argument is present in the TSS section describing this hardware-based limitation and
- b) All hardware components that are the basis of such argument are definitively identified in the ST. For example, if specific Ethernet Controller or WiFi radio chip is the root cause of such limitation, these chips must be identified.

The evaluator attempted to connect to the TOE using an SSH client. The rekey time limit on the TOE is 1 hour. The evaluator performed the rekey Time Limit test and found that the TOE initiated a rekey event before 1 hour had elapsed. The rekey data limit on the TOE is 1GB. The evaluator performed the Data Limit rekey test and found that the TOE rekeyed before 1GB of traffic had passed through the connection.

**Component TSS Assurance Activities:** None Defined

**Component Guidance Assurance Activities:** None Defined

**Component Testing Assurance Activities:** None Defined



## 2.2.11 TLS CLIENT PROTOCOL WITHOUT MUTUAL AUTHENTICATION - PER TD0634 & TD0670 (NDCPP22E:FCS\_TLSC\_EXT.1)

### 2.2.11.1 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 6.2 of the [ST] states the TOE supports the following TLS ciphersuites to connect with external syslog audit servers (provided by the IBMFIPSJCE library).

- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256

This list is consistent with the requirement.

**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.

Chapter 1, the section entitled "Supported cipher suites" in the [CC-Guide] states that during the STIG hardening, QRadar is configured to use TLS protocol version 1.2 and to use the supported cipher suites only. This section also lists the supported cipher suites.

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] list the TLS Cryptographic Configuration supported by the TOE and describe how to limit the cipher suites in use to match only those allowed by the Security Target.

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

Test 2: The evaluator shall 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.

For the following tests the evaluator configured the test server to require mutual authentication and configured the TOE to establish a TLS session with this test server.

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.



Test 2: The evaluator configured the test server to send a certificate with the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is accepted by the TOE. The evaluator reconfigured the test server to retry the TLS session using a cert that is missing the Server Authentication purpose in the extendedKeyUsage field. Using a network sniffer the evaluator captured the TLS session negotiation and observed that the TLS session is rejected by the TOE.

Test 3: The evaluator established a TLS session from the TOE. A modified test server negotiates TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256, but returns an ECDSA Certificate. Using a network sniffer to capture the TLS session negotiation and observed that the TLS session is not negotiated successfully.

Test 4a: The evaluator configured a test server to offer only the TLS\_NULL\_WITH\_NULL\_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.

Test 4b: The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to choose a ciphersuite that the TOE did not offer in its Client Hello handshake message. The evaluator observed that the TOE rejected the connection.

Test 4c: The evaluator configured the TOE to connect to a test server using TLS with a TOE supported ECDHE key exchange method. The evaluator also configured the test server to accept that same ECHDE key exchange method, but to require a curve that was not supported by the TOE (i.e., P-192). The evaluator then observed the TOE reject the connection attempt.

Test 5a: The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to use a TLS version in the Server Hello that is a non-supported TLS version (version 1.4 represented by two bytes 0x0305). Please note the distinction between the TLS record layer version (which remained unchanged as version 1.2) and the TLS version within the Server Hello message (which indicates the Server's selected TLS version to govern the remaining handshake messages). The test requires alternation of the TLS version in the Server Hello message, not the TLS version in the TLS record layer. The evaluator verified that the TOE client rejected the negotiation.

Test 5b: The evaluator configured the TOE to connect to a test server using TLS. During the connection the evaluator caused the server to modify the signature block in the Server's Key Exchange handshake message. The evaluator verified that the TOE client rejected the negotiation.

Test 6a & 6b: The evaluator obtained a packet capture of the TLS session negotiation between the TOE (client) and a test server. The server implementation of the TLS protocol was modified as stated in the assurance activity 'a' and 'b'. The evaluator verified that the client did not finish the negotiation and no application data was transferred.

Test 6c: The evaluator obtained a packet capture of the TLS session negotiation between the TOE (client) and a test server. The server implementation of the TLS protocol was modified as stated in the assurance activity. The evaluator verified that the client rejected the Server Key Exchange handshake message.



### 2.2.11.2 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 6.2 of [ST] states that when validating a certificate, the TOE ensures the distinguished name (DN) or Subject Alternate Name (SAN) fields in the certificate match the peer identifier DNS name. The TOE supports the use of wildcard values within a certificate's DN or SAN field. The TOE does not support certificate pinning nor does it support use of an IP address as a reference identifiers.

The TOE is not distributed.

**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.

The [CC-Guide] describes certificate validation as two parts. The first is validation of a certificate for import. This type of validation focuses upon the validity of a certificate with respect to validity dates, KeyUsages, ExtendedKeyUsages, Basic Constraints flags, CRL distribution points, certificate chain validation and other



certificate characteristics beyond the Subject DN and/or Subject Alternate Name extensions. These checks are described in the section entitled "Importing an endpoint certificate".

The second form of validation involves checks to ensure the certificate being validated appropriately identifies the expected peer. These checks vary depending upon whether the TOE is a TLS client or a TLS server. The section entitled "Certificate validation during establishment of a trusted TLS channel" in chapter 7 of [CC-Guide] describes how the TOE validates certificates. It states that the TOE is first validated according to the "first validation" described above with the addition of the following verification tests:

If the TOE is the TLS client (as in Chapter 6), the peer (server) certificate will be authenticated based upon the hostname configured in the TLS client. The certificate will be accepted if:

- o The Subject Alternative Name (SAN) is present, then the requested hostname must be present in the list of alternative names. The Common Name (CN) field of the Subject will not be considered.
- o If the SAN is not present, then the CN must match the requested hostname.

If the TOE is the TLS server and it requires the client peer to provide a certificate for authentication, the certificate will optionally (see configuration in Chapter 5) examine the CN and/or the Issuer.

- o If the CN is in the CN Allow List configured in the server, the certificate will be accepted. This is a string comparison.
- o If the Issuer of the peer certificate matches the Issuer PEM configured in the server, the certificate will be accepted. This is an addition signature verification.

The section entitled "Adding forwarding destinations to QRadar" describes how an administrator can specify the hostname for the syslog server. This section states that the destination must match the hostname provided by the certificate presented by the TLS peer. This section also states that the TOE does not allow IP addresses to be configured as the identifiers associated with a TLS connection.)>

The TOE is not distributed.

**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.)

The TOE utilizes TLS for FTP\_ITC.1 and FTP\_TRP communications, and the TOE supports only DNS names as a reference identifier. Therefore only tests 1 through 5 are applicable.

The evaluator performed these tests using only a DNS name.

Test 1: The TOE was configured to expect these reference identifiers in either the CN or SAN. The evaluator then established a TLS session from the TOE targeting a server using a valid certificate with a CN matching the domain name used by the client. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed a successful connection. The evaluator then established a TLS session from the TOE targeting a server using a server certificate that does not contain an identifier in either the Subject Alternative Name (SAN) or Common Name (CN) that matches the reference identifier. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 2: The evaluator established 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. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was not negotiated successfully.

Test 3: The evaluator established 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. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 4: The evaluator established 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. Using a network sniffer to capture the TLS session negotiation the evaluator examined the traffic capture and observed that the TLS session was negotiated successfully.

Test 5: This test was performed using DNS reference identifiers only. 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. The evaluator used a network sniffer to capture the TLS session negotiation and observed that the TLS session was negotiated as shown in column 3 of the following table.

<u>Certificate Contents</u>	<u>Host ID</u>	<u>Expected Result</u>
CN=bar.*.example.com	bar.foo.example.com	No Connection



SAN=bar.*.example.com	bar.foo.example.com	No Connection
CN=*.example.com	foo.example.com	Successful Connection
SAN=*.example.com	foo.example.com	Successful Connection
CN=*.example.com	example.com	No Connection
SAN=*.example.com	example.com	No Connection
CN=*.example.com	bar.foo.example.com	No Connection
SAN=*.example.com	bar.foo.example.com	No Connection

Test 6: This test does not apply because the TOE does not support IP addresses as reference identifiers.

Test 7: The TOE does not utilize TLS for FTP\_ITT communication and therefore this test is not applicable.

### 2.2.11.3 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: As part of testing FIA\_X509\_EXT.1.1/Rev Test 1, the evaluator loaded certificates needed to validate the certificate that was presented by an external entity and demonstrated that the function succeeds and a trusted channel was established.

Test 2: This test has been performed as part of several other test activities namely:

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: The TOE does not offer the ability to override certificate validation failures.

#### **2.2.11.4 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 6.2 of [ST] indicates that by default the TOE supports Elliptic Curves with secp256r1, secp384r1, secp521r1 in TLS ECDHE key exchanges.

**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 section entitled "Limiting the cipher suites in use" in [CC-Guide] includes a note stating that the TOE does not allow administrators to configure the Supported Elliptic Curves/Supported Groups Extension sent during TLS negotiations.

**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.

Test 1: The evaluator attempted to establish a TLS session between the TOE and a test server configured to allow only one key exchange method. The evaluator observed that the TOE was able to connect with the test server using the following key exchange methods.

ECDHE w/ P-256 curve

ECDHE w/ P-384 curve

ECDHE w/ P-521 curve

**Component TSS Assurance Activities:** None Defined

**Component Guidance Assurance Activities:** None Defined

**Component Testing Assurance Activities:** None Defined

#### **2.2.12 TLS CLIENT SUPPORT FOR MUTUAL AUTHENTICATION - PER TD0670 (NDcPP22E:FCS\_TLSC\_EXT.2)**



### 2.2.12.1 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 6.2 of [ST] indicates that when a TLS server requests TLS authentication via certificates, the TOE is capable of providing an X509v3 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.

Chapter 5, "Configure TLS syslog for inbound data" in the [CC-Guide] contains steps to install certificates used to authenticate the TOE to the syslog server. Chapter 4, the section entitled "Creating a CSR request by using a 2048-bit RSA key" in the [CC-Guide] also has instructions that describe the process of generating a CSR, exporting the CSR, and importing the signed certificates.

**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.

Using the packet capture from FCS\_TLSC\_EXT.1.1 Test 2, the evaluator confirmed that the TOE does provide a Certificate (type 11) and Certificate Verify (type 15) messages during the negotiation of a TLS channel and that for valid connections, Application Data is transferred. The specific messages can be seen in the mentioned packet capture.

### 2.2.13 TLS SERVER PROTOCOL WITHOUT MUTUAL AUTHENTICATION - PER TD0635 (NDcPP22E:FCS\_TLSS\_EXT.1)

#### 2.2.13.1 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 6.2 of the [ST] states the TOE allows inbound TLS connections using only the following ciphersuites (provided by OpenSSL library).

- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_CBC\_SHA256
- TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256

This list is consistent with the requirement.

**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).

Chapter 1, the section entitled "Supported cipher suites" in the [CC-Guide] states that during the STIG hardening, QRadar is configured to use TLS protocol version 1.2 and to use the supported cipher suites only. This section also lists the supported cipher suites.

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] list the TLS Cryptographic Configuration supported by the TOE and describe how to limit the cipher suites in use to match only those allowed by the Security Target.

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

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

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

- a) Modify a byte in the Client Finished handshake message, and verify that the server rejects the connection and does not send any application data.
- b) (Test Intent: The intent of this test is to ensure that the server's TLS implementation immediately makes use of the key exchange and authentication algorithms to: a) Correctly encrypt (D)TLS Finished message and b) Encrypt every (D)TLS message after session keys are negotiated.)



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

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

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

Test 2: The evaluator sent 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 verified that the server denied all such connection attempts. Additionally, the evaluator sent a Client Hello to the server containing only the TLS\_NULL\_WITH\_NULL\_NULL ciphersuite and verified that the server denied the connection.

Test 3(a): The evaluator modified a byte in the Finished handshake message and observed in packet capture that the connection was rejected.

Test 3(b): The evaluator made connection attempts from a client to the TOE. The evaluator located the ChangeCipherSpec message (Content type hexadecimal 14) in the packet capture and found it to be followed by an EncryptedHandshakeMessage (Content type hexadecimal 16). The evaluator examined the payload of the EncryptedHandshakeMessage and determined it contained an encrypted version of the required Finished message (as defined by the assurance activity).

### **2.2.13.2 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 6.2 of the [ST] explains the TOE acts as a TLS server supporting TLSv1.2 only. No older versions of TLS, and no older version of TLS or of SSL are supported.



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

Chapter 1, the section entitled "Supported cipher suites" in the [CC-Guide] states that during the STIG hardening, QRadar is configured to use TLS protocol version 1.2 and to use the supported cipher suites only. This section also lists the supported cipher suites.

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] list the TLS Cryptographic Configuration supported by the TOE and describe how to limit the cipher suites in use to match only those allowed by the Security Target.

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

The evaluator attempted to establish a TLS session on the TOE with each of the unsupported versions of SSL and TLS. The evaluator used a network sniffer to capture the session negotiation and observed that the expected protocol and version were offered and rejected during negotiation. Only TLSv1.2 is supported by the TOE.

### 2.2.13.3 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 6.2 of the [ST] states the TOE utilizes RSA w/ key sizes of 2048-bits for authentication and can support ECDHE key exchanges. It also indicates that the TOE supports Elliptic Curves with secp256r1, secp384r1, secp521r1 in TLS ECDHE key exchanges.

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

Chapter 1, the section entitled "Supported cipher suites" in the [CC-Guide] states that during the STIG hardening, QRadar is configured to use TLS protocol version 1.2 and to use the supported cipher suites only. This section also lists the supported cipher suites.

Chapter 3, the sections entitled "Configuring Java TLS ciphers" and "Limiting the cipher suites in use" in the [CC-Guide] list the TLS Cryptographic Configuration supported by the TOE and describe how to limit the cipher suites in use to match only those allowed by the Security Target.

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

- a) The evaluator shall repeat this test for each supported elliptic curve. The evaluator shall attempt a connection using a supported ECDHE ciphersuite and a single supported elliptic curve specified in the Elliptic Curves Extension.



The Evaluator shall verify (through a packet capture or instrumented client) that the TOE selects the same curve in the Server Key Exchange message and successfully establishes the connection.

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

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

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

Test 1a: The evaluator attempted to establish a TLS session with the TOE when the evaluator's TLS client specified only one key exchange method in the Client Hello. The evaluator observed that the TOE selected the same curve as offered by the client for the following key exchange methods.

- ECDHE w/ P-256 curve
- ECDHE w/ P-384 curve
- ECDHE w/ P-521 curve

Test 1b: The evaluator also attempted a connection using ECDHE w/ P-192 curve and observed the TOE rejected the connection attempt.

Test 2: Not applicable. The TOE does not support ciphersuites that use DHE key exchange.

Test 3: Not applicable. The TOE does not support ciphersuites that use RSA key exchange.

#### **2.2.13.4 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 6.2 of [ST] states that the TOE support session resumption based on session tickets (adhering to the structural format from section 4 of RFC 5077). The session tickets are encrypted using AES CBC mode with a 128-bit key.

**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: Not applicable. The TOE does claim to support session resumption based on session tickets.

Test 2: Not applicable. The TOE does NOT claim to support session resumption based on session ID values.

Test 3: The TOE does support session resumption via session tickets, therefore the evaluator attempted to resume a session using a valid session ticket. The server correctly reuses the client's proposed session ticket and the session is successfully resumed. 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 does not reuse the existing session, but instead requires full renegotiation of the TLS Session.

**Component TSS Assurance Activities:** None Defined

**Component Guidance Assurance Activities:** None Defined

**Component Testing Assurance Activities:** None Defined

## 2.3 IDENTIFICATION AND AUTHENTICATION (FIA)

### 2.3.1 AUTHENTICATION FAILURE MANAGEMENT (NDcPP22E:FIA\_AFL.1)

#### 2.3.1.1 NDcPP22E:FIA\_AFL.1.1



**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.3.1.2 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 6.3 of the ST states the TOE allows an administrator to configure the number of failed authentication attempts that trigger the locking of a user account. The limit of failed authentication attempts (a.k.a., the Max Login Failure limit) can be set to between 1 and 2,147,483,647. When the Max Login Failure limit is reached, the user account is locked. This prevents successful login of that account using all remote administration methods until the configured time limit has passed from the last failed authentication attempt. The configured lockout time is specified by an administrator and can be set to between 1 minute and 2,147,483,647 minutes. Admin accounts are never locked out from using the local console.

**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.

Chapter 7, the section entitled "Administrative logins" in the [CC-Guide] provides instructions to enable and disable the account lockout policy which sets the number of successive unsuccessful authentication attempts before an



account becomes locked. The instructions state that the administrator can define the policy to keep the account locked until the locked account can be automatically unlocked after a specified period. This section provides instructions to define the unlock period.

This section also states that the TOE can be configured to block remote login requests from a user account for a configurable period of time after a configurable number of failed remote login attempts. Admin accounts are never locked out from using the local console.

**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 & 2: The evaluator configured the number of unsuccessful authentication attempts to two, and then performed two login attempts into the "admin" user account with an incorrect password followed by a login attempt with the correct password, and verified that the login failed. The evaluator then logged into the TOE after the time expired and verified the authentication succeeded.

### 2.3.2 PASSWORD MANAGEMENT (NDCPP22E:FIA\_PMG\_EXT.1)

#### 2.3.2.1 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 6.3 of the [ST] states that the TOE allows the administrator to configure a minimum length password between 15 and 255 characters (including a length of 15 characters). The TOE also supports the use of uppercase letters, lowercase letters, numeric values and special characters for passwords. The special characters permitted to be used in passwords including “!”, “@”, “#”, “\$”, “%”, “^”, “&”, “\*”, “(“ and “)“.

**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.

Chapter 7, the section entitled "Configuring the password policy" in the [CC-Guide] contains steps that identifies the characters that may be used in passwords. This section includes guidance to security administrators on the composition of strong passwords. It also requires security administrators to set the minimum password length to 15 characters.

Also, this section explains that the administrator may configure a minimum password length between 15 and 255 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.

The evaluator first configured a password policy on the TOE as described in the guidance documentation. The evaluator then attempted to create a user with three different password combinations (a 14-character password meeting the complexity requirements, a 15-character password lacking the complexity requirements, and a valid 15-character password meeting the complexity requirements). Valid passwords are accepted by the TOE, and invalid passwords will display an error. In each case, the password policy was enforced.



### 2.3.3 PROTECTED AUTHENTICATION FEEDBACK (NDcPP22E:FIA\_UAU.7)

#### 2.3.3.1 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.

Since section 6.3 of the [ST] indicates that the TOE obscures passwords when entered by administrators by default, the only required configuration is the creation of account (with passwords) for administrators. This is described in Chapter 2, the section entitled "Creating a non-root user in a STIG-compliant environment" in the [CC-Guide] contains a step indicating that each admin user has their own account and password.

**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.

The evaluator observed that passwords are obscured while logging in at the console, via SSH and via the web UI. This was tested as part of FIA\_UIA\_EXT.1-t1.

### 2.3.4 PASSWORD-BASED AUTHENTICATION MECHANISM (NDcPP22E:FIA\_UAU\_EXT.2)

#### 2.3.4.1 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.

See 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.

Evaluation Activities for this requirement are covered under those for NDcPP22e: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

### 2.3.5 USER IDENTIFICATION AND AUTHENTICATION (NDcPP22E:FIA\_UIA\_EXT.1)

#### 2.3.5.1 NDcPP22E:FIA\_UIA\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.3.5.2 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 6.3 of the [ST] states that the TOE authenticates administrators using a local password-based mechanism or using a public-key based authentication mechanism. The password-based mechanism is available for use through the local console CLI, the remote SSH CLI, the TLS-protected WebUI and the TLS-protected RestAPI. The public-key authentication method is supported only for SSH connections.

Section 6.3 also states that the TOE does not offer any services through the administrative interfaces prior to authenticating the connecting user other than the display of a TOE warning banner. In order for an administrative user to access the TOE (i.e., to perform any functions except to see a configured login banner or to access network or SAN services), a user account including a user name and password must be created for the user. A successful logon occurs when the user presents to the TOE a valid administrative user name along with either the correct password or verification of a public-key.

The TOE is not distributed.

**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 provides the ability for security administrators to securely configure and manage the TOE via a locally connected terminal, a remote SSH connection or an HTTPS/TLS Web-based GUI connection. The majority of the administrative functionality is available only through the Web GUI.

Chapter 1, the section entitled "Supported cipher suites" in the [CC-Guide] states that during the STIG hardening, QRadar is configured to use TLS protocol version 1.2 and to use the supported cipher suites only. It also lists all the supported cipher suites configured at the time of initial configuration.

Chapter 2, "QRadar installations for highly secure environments" in the [CC-Guide] describes how to install the TOE into a STIG environment as part of the requirements for configuring the TOE into its evaluated configuration. The section entitled "Running the hardening script on the Console" includes instructions for creating a password for the root account. The section entitled "Creating a non-root user in a STIG-compliant environment" describes how to create a non-root user in order to log into the TOE remotely. The section entitled "Logging in to QRadar" details logging into the device remotely using a web browser.

Chapter 3, the section entitled "Configure SSH to use public key authentication only" in the [CC-Guide] details configuring the TOE for ssh public key and the section entitled "Configuring the SSHD\_config file" describes how to modify the SSH config file for algorithm selection. The section entitled "Limiting the cipher suites in use" describes how to limit the TLS ciphers.





**Component Testing Assurance Activities:** The evaluator shall perform the following tests for each method by which administrators access the TOE (local and remote), as well as for each type of credential supported by the login method:

- a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method. For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.
- b) Test 2: The evaluator shall configure the services allowed (if any) according to the guidance documentation, and then determine the services available to an external remote entity. The evaluator shall determine that the list of services available is limited to those specified in the requirement.
- c) Test 3: For local access, the evaluator shall determine what services are available to a local administrator prior to logging in, and make sure this list is consistent with the requirement.
- d) Test 4: For distributed TOEs where not all TOE components support the authentication of Security Administrators according to FIA\_UIA\_EXT.1 and FIA\_UAU\_EXT.2, the evaluator shall test that the components authenticate Security Administrators as described in the TSS.

Test 1 - The evaluator performed an unsuccessful and successful logon of each type (SSH CLI, Web UI and Local Console) using bad and good credentials respectively, and observed that when providing correct information, the connection was successful, but providing incorrect information resulted in a denial of access.

Test 2 - There are no services available prior to login. The evaluator was able to observe the TOE displayed a banner to the user before login as identified in the tests performed in FTA\_TAB.1.

Test 3 - No functions were available to the administrator accessing the console with the exception of acknowledging the banner.

The TOE is not distributed, therefore Test 4 is not applicable.

### **2.3.6 X.509 CERTIFICATE VALIDATION (NDCPP22E:FIA\_X509\_EXT.1/REV)**

#### **2.3.6.1 NDCPP22E:FIA\_X509\_EXT.1.1/REV**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT\_TUD\_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA\_X509\_EXT.1.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3



certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

a) Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOE's 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)

The TOE validates certificates as part of the TLS Session establishment with a syslog server. The evaluator performed each of the following tests on this interface.

Test 1 -- The evaluator configured the TOE and a peer with valid certificates. The evaluator then attempted to make a connection between the peer devices using TLS protected syslog. A successful connection was made. The evaluator then configured a syslog server that presented a certificate chain with an invalid certification path by deleting an intermediate CA so that the certificate chain was invalid because of a missing certificate. The connection between the TOE and the syslog server was refused by the TOE.

Test 2 -- The evaluator used the TOE's TLS client (syslog) to attempt connections to a test server. The test server then presented a certificate during the TLS negotiation where the certificate was expired. The TOE rejected the connection.

Test 3 -- The evaluator used a test server to accept connection attempts from the TOE TLS client (syslog). The test server then presented a certificate during the TLS negotiation where the certificate was valid. A packet capture was obtained of this TLS negotiation which shows that the connection was successful. The evaluator revoked



certificates in the chain (individually) and attempted the same connection from the syslog client. The attempt after revoking the certificate was not successful.

Test 4 -- The evaluator configured a CRL revocation server to present a certificate that does not have the CRLsign Key Usage. The evaluator established a TLS session from the TOE TLS client such that the TOE receives a CRL signed by the invalid certificate and ensured that the TLS session was not negotiated successfully.

Test 5 -- The evaluator configured a test server to present a certificate that had a byte in the first eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 6 -- The evaluator configured a test server to present a certificate that had a byte in the last eight bytes modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 7 -- The evaluator configured a test server to present a certificate that had a byte in the public key of the certificate modified to the TOE. The evaluator then attempted to make a connection between the peer devices. When the TOE attempted to connect to the test server using syslog, the TOE rejected the connection.

Test 8 -- The TOE does not support ECDSA Certificates, therefore this test is not applicable.

### **2.3.6.2 NDcPP22E:FIA\_X509\_EXT.1.2/REV**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** The evaluator shall perform the following tests for FIA\_X509\_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA\_X509\_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA\_X509\_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation). For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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



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

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

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

Test 1: The evaluator configured a test syslog server to present a certificate chain containing a CA certificate lacking the basicConstraints extension. The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test server. The evaluator observed that the TOE rejected the connections.

Test 2: The evaluator configured a test server to present a certificate chain containing a CA certificate having the basicConstraints section but with the cA flag not set (i.e., FALSE). The evaluator then used the TOE TLS client (syslog) to attempt to connect to the test server. The evaluator observed that the TOE rejected the connection.

**Component TSS Assurance Activities:** The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place, and that the TSS identifies any of the rules for extendedKeyUsage fields (in FIA\_X509\_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied). It is expected that revocation checking is performed when a certificate is used in an authentication step and when performing trusted updates (if selected). It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected).

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

Section 6.3 states that the TOE supports the use of CRLs to determine the revocation status of certificates. Certificates are validated as part of the authentication process when they are presented to the TOE and when they are loaded into the TOE. The following fields are verified as appropriate:

- Expiration - certificate cannot be expired
- Common Name - Needs to be FQN device name or IP. Wildcards are allowed only for 1 level of sub-domain and not allowed for the main domain. URI is not supported as an identifier type.
- CA Field - must be true if CA certificate



- Key Usage - Need to have "Certificate Sign" in case of CA certificates and "Digital Signature" in case of identity certificates.
- X509v3 Extended Key Usage - Need to rightly indicate whether it is for use as "server" certificate or "client" certificate. If incorrect, connection is not allowed
- X509v3 CRL Distribution Points- Certificate must be valid per a current CRL.
- Subject Alternative Method - Not a mandatory attribute. If present, the values stored in this will take priority over the CN in Subject attribute.
- Basics Constraints - Attribute must be present and must have CA Field

**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 [CC-Guide] describes certificate validation as two parts. The first is validation of a certificate for import. This type of validation focuses upon the validity of a certificate with respect to validity dates, KeyUsages, ExtendedKeyUsages, Basic Constraints flags, CRL distribution points, certificate chain validation and other certificate characteristics beyond the Subject DN and/or Subject Alternate Name extensions. These checks are described in the section entitled "Importing an endpoint certificate".

The discussion states that the EKU is not required but will be tested if present. It states that TLS Web Server Authentication is required for certificates intended for server use; the TLS Web Client Authentication is required for certificates intended for client use; and that other uses are permitted in addition to these.

This section also states that if the CRL Distribution Points (CRLDP) extension is present, revocation status will be checked.

The second form of validation involves checks to ensure the certificate being validated appropriately identifies the expected peer. These checks vary depending upon whether the TOE is a TLS client or a TLS server. The section entitled "Certificate validation during establishment of a trusted TLS channel" in chapter 7 of [CC-Guide] describes how the TOE validates certificates.

**Component Testing Assurance Activities:** None Defined

## 2.3.7 X.509 CERTIFICATE AUTHENTICATION (NDcPP22E:FIA\_X509\_EXT.2)

### 2.3.7.1 NDcPP22E:FIA\_X509\_EXT.2.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined



### 2.3.7.2 NDcPP22E:FIA\_X509\_EXT.2.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

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

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

Section 6.3 of the [ST] explains that the TOE uses X509v3 certificates for TLS communications. Certificates are validated as part of the authentication process when they are presented to the TOE and when they are loaded into the TOE. Administrators must install a trusted root CA certificate. The TOE must also install its own certificate which it will present to the syslog server during TLS negotiations. The certificate presented by a syslog peer and the certificate imported into the TOE must chain to this root CA certificate.

If the TOE cannot establish a connection with a revocation server to check the status of a certificate in question, the TOE does not accept the certificate as valid.

**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.

Chapter 3, the section entitled "Certificate Revocation List" in the [CC-Guide] indicates that the TOE uses CRL to perform certificate revocation checking. This section states that if the revocation server is not reachable, the TLS Session Establishment will be rejected.

**Component Testing Assurance Activities:** The evaluator shall perform the following test for each trusted channel:

The evaluator shall demonstrate that using a valid certificate that requires certificate validation checking to be performed in at least some part by communicating with a non-TOE IT entity. The evaluator shall then manipulate the environment so that the TOE is unable to verify the validity of the certificate, and observe that the action selected in FIA\_X509\_EXT.2.2 is performed. If the selected action is administrator-configurable, then the evaluator shall follow the guidance documentation to determine that all supported administrator-configurable options behave in their documented manner.





The evaluator first set up a valid connection between the TOE and a TLS syslog server. Then, the evaluator caused the revocation server for the cert presented by the syslog server to disconnect its network interface such that the TOE could no longer reach the revocation server. Attempts to connect the TOE to the syslog server failed as expected because the TOE could not reach the revocation server to determine the revocation status of the certificate presented by the syslog server.

### 2.3.8 X.509 CERTIFICATE REQUESTS (NDcPP22E:FIA\_X509\_EXT.3)

#### 2.3.8.1 NDcPP22E:FIA\_X509\_EXT.3.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.3.8.2 NDcPP22E:FIA\_X509\_EXT.3.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

**Component TSS Assurance Activities:** If the ST author selects 'device-specific information', the evaluator shall verify that the TSS contains a description of the device-specific fields used in certificate requests.

The requirement did not select "device-specific information" so no explanation is required.

**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.

Chapter 4, the section entitled "Creating a CSR request by using a 2048-bit RSA key" in the [CC-Guide] contains steps to generate a CSR for use by the TOE as a syslog client. The steps identify the openssl command that is used to generate a CSR which can be exported from the TOE. This command takes a number of arguments on the command line which specify the export location and prompts the user for remaining values used in the certificate. The values which the TOE prompts for are Country Name, State, Locality, Organization, Organizational Unit, Common Name, and Email Address. The description also indicates which of these values are required to be completed and which are optional.

**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 guidance to generate a certificate signing request on the TOE. The TOE is able to display the CSR to be signed by a CA. The evaluator used an OpenSSL command to view the CSR and verified that the CSR contained a public key.

Test 2 - Testing for both valid and invalid certificate paths was performed in FIA\_X509\_EXT.1.1.

## 2.4 SECURITY MANAGEMENT (FMT)

### 2.4.1 MANAGEMENT OF SECURITY FUNCTIONS BEHAVIOUR (NDcPP22E:FMT\_MOF.1/MANUALUPDATE)

#### 2.4.1.1 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.

The TOE is not distributed.

**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).

Chapter 7, the section entitled "Verifying secure updates" in the [CC-Guide] explains that Administrators need to manually apply QRadar secure updates when they are released. An operational system must be taken offline



during system update and returned to operational state afterward. This section contains instructions to perform an update which include an explanation that security updates are signed by the vendor using QRadar Security Keys and verified during the update installation. This section also states that if the signature verification fails, or initialization of the update installer fails for any reason an error message is presented and the update terminates. This section also provides instructions to determine the currently running QRadar version and security patch version.

This TOE is not distributed and the assurance activity states that there are no specific requirements for non-distributed TOEs.

**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.

The TOE restricts the ability to perform manual updates to Security Administrators with prior authentication. This has been tested in FIA\_UIA\_EXT.1 where the evaluator showed that the TOE does not offer any services through the administrative interfaces prior to authenticating the connecting user other than the display of a TOE warning banner. See Test Case FPT\_TUD\_EXT.1 for results of a successful update with prior authentication.

## **2.4.2 MANAGEMENT OF TSF DATA (NDCPP22E:FMT\_MTD.1/COREDATA)**

### **2.4.2.1 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 6.4 of the [ST] states that only security administrators can login to the TOE via its command line or Web GUI interfaces. This restricts access to all management functions including management of X.509 certificates.

**Component Guidance Assurance Activities:** The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the cPP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

If the TOE supports handling of X.509v3 certificates and provides a trust store, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to configure and maintain the trust store in a secure way. If the TOE supports loading of CA certificates, the evaluator shall review the guidance documentation to determine that it provides sufficient information for the administrator to securely load CA certificates into the trust store. The evaluator shall also review the guidance documentation to determine that it explains how to designate a CA certificate a trust anchor.

The section entitled "Logging in to QRadar" in [CC-Guide] states that No features of the QRadar console, aside from login, are available until after successful authentication.

The section entitled "Importing the root CA of the TLS client" provides instructions to load root CA certificates. The section entitled "x509 Certificate Management and Validation in QRadar" states that during import, verification is performed to confirm the certificate's compliance with QRadar's expectation for validity, usage and revocation status. These two sections include instructions with steps to indicate the CA certificate being imported is a "Root Certificate."

**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.

No separate testing for FMT\_MTD.1/CoreData is required.

### 2.4.3 MANAGEMENT OF TSF DATA (NDCPP22E:FMT\_MTD.1/CRYPTOKEYS)

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



Section 6.4 of [ST] states that only administrators can import certificates and configure cryptography on the TOE. An administrator may generate and delete SSH host keys as well as keys associated with a CSR. The administrator may also import keys associated with CA certificates, as well as identity certificates that are used by the TOE.

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

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

The TSS indicates that the TOE can import certificates, manage SSH host keys, and manage keys associated with a CSR. Each of these topics is discussed in [CC-Guide].

Certificate import is described in the section entitled "Importing the root CA of the TLS client" and "Importing the TLS server certificates". The deletion of certificates is described within Chapter 7 section entitled "Deleting a CA root/anchor certificate."

The section entitled "Configuring the SSHD\_config file" states that HostKeys will be generated automatically when the SSHd service is restarted. To regenerate Host Keys, delete /etc/ssh/ssh\_host\_\* key files (rm /etc/ssh/ssh\_host\_\*) and restart SSHd with systemctl restart sshd.

Keys associated with a CSR are generated and deleted as described by the instructions found in the section entitled "Creating a CSR request by using a 2048-bit RSA key".

**Component Testing Assurance Activities:** The evaluator shall try to perform at least one of the related actions (modify, delete, generate/import) without prior authentication as security administrator (either by authentication as a non-administrative user, if supported, or without authentication at all). Attempts to perform related actions without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to manage cryptographic keys can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This attempt should be successful.

The evaluator first confirmed that there is an admin account configured on the TOE. An admin account is the only account able to perform any cryptographic keys changes (modify, delete, generate/import) on the TOE. The evaluator then logged out of the admin account and verified that without prior authentication, no cryptographic keys changes can be performed on the TOE. The evaluator then provided the correct credentials to the admin account and logged onto the TOE using that account. The evaluator then verified that with prior authentication, cryptographic keys changes can be performed on the TOE.

#### **2.4.4 SPECIFICATION OF MANAGEMENT FUNCTIONS - PER TD0631 (NDcPP22E:FMT\_SMF.1)**



#### 2.4.4.1 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.

Section 6.4 explains the TOE provides the ability for security administrators to securely configure and manage the TOE. Administrators can connect to the Command Line Interface (CLI) via either a remote SSH connection or using a locally connected terminal. The CLI is used for initial configuration, creating CSR requests, performing updates, reviewing local audit logs, and remediation of AIDE integrity check discrepancies. Administrators perform a majority of administrative tasks using an HTTPS/TLS protected communication channel offering a Web-based GUI.

Section 6.4 of [ST] also lists the management functions offered by the TOE. These functions correspond to those required by FMT\_SMF.1 and were observed by the evaluator during testing. The specific management capabilities defined in the [ST] include:

- Ability to administer the TOE locally and remotely;
- Ability to configure the access banner;
- Ability to configure the session inactivity time before session termination or locking;



- Ability to update the TOE, and to verify the updates using [digital signature] capability prior to installing those updates;
- Ability to configure the authentication failure parameters for FIA\_AFL.1;
- Ability to configure audit behavior (e.g. changes to storage locations for audit; changes to behaviour when local audit storage space is full),
- Ability to manage the cryptographic keys,
- Ability to configure the cryptographic functionality,
- Ability to configure thresholds for SSH rekeying,
- Ability to configure the reference identifier for the peer,,
- Ability to manage the TOE's trust store and designate X509.v3 certificates as trust anchors,
- Ability to import X509v3 certificates to the TOE's trust store,
- Ability to set the time which is used for time-stamp.

**Component Guidance Assurance Activities:** See TSS Assurance Activities

The TOE is compliant with all requirements in the ST as identified in this report.

**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 TOE security functions are identified in the guidance documentation and have been tested as documented throughout this AAR.

## **2.4.5 RESTRICTIONS ON SECURITY ROLES (NDcPP22E:FMT\_SMR.2)**

### **2.4.5.1 NDcPP22E:FMT\_SMR.2.1**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### **2.4.5.2 NDcPP22E:FMT\_SMR.2.2**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### **2.4.5.3 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 6.4 of [ST] explains that the TOE supports only a Security Administrator role. The TOE supports local administration via a CLI or a locally attached network device. Remote administration is provided through a CLI protected by SSH or through a web GUI protected by HTTPS/TLS. The majority of administrative functionality is available only through the web GUI.

**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.

Chapter 2, the section entitled "Logging in to QRadar" in the [CC-Guide] indicates that the TOE can be managed using 1) a CLI that is accessed through a local console or via SSH 2) the Web GUI which is protected by TLS/HTTPS. The Guidance assurance activities associated with FCS\_SSHS\_EXT.1, FCS\_TLSC\_EXT.1 and FCS\_TLSS\_EXT.1 are identify relevant sections of [CC-Guide] that contain steps which provide instructions to configure the cryptographic capabilities of the TOE's SSH and TLS servers.

**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.

The TOE offers a Console CLI, SSH-protected CLI and WebUI. Testing throughout the evaluation covered these interfaces and addressed features available on each interface (e.g., Identification and Authentication was tested on all three interfaces; Console, SSH, and WebUI). The TOE's configuration operations were performed using either interface interchangeably throughout the course of testing.

## 2.5 PROTECTION OF THE TSF (FPT)

### 2.5.1 PROTECTION OF ADMINISTRATOR PASSWORDS (NDcPP22E:FPT\_APW\_EXT.1)

#### 2.5.1.1 NDcPP22E:FPT\_APW\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.5.1.2 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 6.5 of the [ST] explains administrator passwords are stored SHA-512 hashed and salted. No interface is offered by the TOE to present cleartext passwords.

**Component Guidance Assurance Activities:** None Defined

**Component Testing Assurance Activities:** None Defined

## **2.5.2 PROTECTION OF TSF DATA (FOR READING OF ALL PRE-SHARED, SYMMETRIC AND PRIVATE KEYS) (NDCPP22E:FPT\_SKP\_EXT.1)**

### **2.5.2.1 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 6.5 of the [ST] states that pre-shared keys, symmetric keys, and private keys stored by the TOE are AES encrypted and never presented back to administrators as plaintext. The AES key protecting other keys is stored in a password protected PKCS12 file and is never presented or displayed in any tools or UI. The password to access the PCKS12 file is used by the TOE to decrypt the AES key, which allows all other keys to become accessible to the TOE software.

**Component Guidance Assurance Activities:** None Defined

**Component Testing Assurance Activities:** None Defined

## **2.5.3 RELIABLE TIME STAMPS - PER TD0632 (NDCPP22E:FPT\_STM\_EXT.1)**

### **2.5.3.1 NDCPP22E:FPT\_STM\_EXT.1.1**





**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### 2.5.3.2 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 6.5 of the [ST] explains that the TOE uses the system time obtained from the hardware for use in audit records, session timeouts, scheduling integrity validation and during certificate validation. The hardware-based time value is considered reliable because it can be set and modified only by an administrator.

**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.

Chapter 7, the section entitled "Configuring system time" in the [CC-Guide] contains instructions to allow time to be set manually by the administrator using the Web GUI and the command line interface.

**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 configure the time on the TOE using both the WebUI and CLI procedures to set time. The evaluator read the time from the TOE using the date command.

Test 2: The TOE does not claim support for an NTP server

Test 3: The TOE does not rely upon an underlying Virtual System.

## 2.5.4 TSF TESTING (NDCPP22E:FPT\_TST\_EXT.1)

### 2.5.4.1 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 6.5 of the [ST] states the TOE runs a suite of self-tests during power-on. These tests include cryptographic known answer tests exercising the algorithms listed in Section 6.2 of the ST. A failure of any of the power-up self-tests panics the module. The only recovery is to reboot. If a reboot cannot solve the problem, the system may be compromised, and QRadar must be reinstalled on the system.



For TSF integrity checking, the TOE uses Advanced Intrusion Detection Environment (AIDE) tool to verify the integrity of TOE software.

AIDE (<https://aide.github.io/>) is a utility that creates a database of files on the system, and then uses that database to ensure file integrity and detect system intrusions. After installation and configuration the administrator creates an AIDE baseline database and schedules the integrity check to run at configured intervals using cron. In the case where the AIDE tool identifies differences in the database the Administrator will review the identified system changes. If all changes are approved the administrator will commit the new database, otherwise the integrity of the device will be restored through a factory re-installation.

These tests together ensure that the TOE continues to operate correctly.

In the event a self-test fails, the administrator is expected to make a system image of the appliance for later investigation, and then reinstall the system. Since the TOE provides services to other network devices, availability is an important consideration in its design. To manage the time needed during startup, the AIDE tool is not run during TOE initialization, but rather is scheduled on a daily basis. This allows the TOE to return to providing network services quicker and to perform the TSF integrity checking more frequently. Guidance suggests that the AIDE tool be scheduled on a daily basis, however, administrators can schedule it more frequently if desired.

**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.

Chapter 7, the section entitled "QRadar self-test" in the [CC-Guide] explains that failure of any self-test during the start-up process requires the admin to restart the system and allow the tests to run again. If the failure persists, the system may be compromised, and QRadar must be reinstalled. This section explains that the TOE software integrity checks are performed using the Advanced Intrusion Detection Environment (AIDE) tool. The AIDE tool compares the current TOE software to a known good baseline. The AIDE tool generates a report indicating the status of TOE software each time it executes, this report indicates if a TOE software integrity check occurred.

Additionally, the section entitled "Editing scripts to configure QRadar in STIG environments" in the [CC-Guide] contains information about the TOE software integrity checks and steps the administrator should follow to ensure these tests run frequently and to review the tool output.

**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 verified that the TOE performs the software/firmware integrity tests and algorithm tests during start up. The output of the tests indicate that they were successful.

### 2.5.5 TRUSTED UPDATE (NDCPP22E:FPT\_TUD\_EXT.1)

#### 2.5.5.1 NDCPP22E:FPT\_TUD\_EXT.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.5.5.2 NDCPP22E:FPT\_TUD\_EXT.1.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.5.5.3 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 6.5 of the [ST] states that the TOE UI presents the currently running version upon request. Given the TOE UI can be either a CLI or WebUI, the administrator can query the currently installed TOE version using commands on the CLI or via screens provided through the WebUI. It also explains that IBM signs product updates using a 2048-bit RSA key. The IBM public keys is installed as part of the TOE. An update is delivered as an executable with an embedded payload. The embedded payload is a SHA-256 signed package. When executed, the payload is verified and if valid it is extracted. Upon failure of the signature check the extracted payload is deleted; upon success there is an indicator that installation can proceed.

**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 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 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.

Chapter 7, the section entitled "Verifying secure updates" in the [CC-Guide] explains that security updates are signed by the vendor using QRadar Security Keys and verified during the update installation. This section also states that if the installation fails, an error with details is logged and the update process is terminated. This section also provides instructions to determine the currently running QRadar version and security patch version.

**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: Prior to performing an update, the evaluator verified the TOE version. The evaluator then followed guidance to install a valid update to the TOE. Upon successful installation, the evaluator verified the TOE version once again and confirmed that the version after the successful update was changed as expected.

Test 2: The evaluator attempted to update the TOE using three illegitimate updates (an update that has been modified with a hex editor, an update that is unsigned, and an update signed with an invalid signature). During each attempt, the TOE checked the integrity of the update, determined that it was invalid, and correctly rejected it. The TOE rejected all the modified updates and the product version did not change.

Test 3: This test is not applicable as the TOE does not support published hash to authenticate firmware/software updates.

## 2.6 TOE ACCESS (FTA)

### 2.6.1 TSF-INITIATED TERMINATION (NDCPP22E:FTA\_SSL.3)

#### 2.6.1.1 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 6.6 of [ST] explains that for remote sessions, the TOE provides an inactivity timeout mechanism that is configurable by an administrator. The timeout mechanism applies to the HTTPS-protected Web GUI and to the SSH-protected CLI interface.

**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.

Chapter 7, the section entitled "Configuring the inactivity time period" in the [CC-Guide] explains that the administrator can configure the number of minutes of inactivity after which IBM QRadar logs the user out and terminates the session.

**Component Testing Assurance Activities:** For each method of remote administration, the evaluator shall perform the following test:

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a remote interactive session with the TOE. The evaluator then observes that the session is terminated after the configured time period.

The evaluator followed the guidance to configure the session timeout periods for SSH and HTTPS remote sessions and confirmed that the session was terminated after the configured time period. The inactivity time period was configured for periods of 3 minutes and 5 minutes.

## **2.6.2 USER-INITIATED TERMINATION (NDcPP22E:FTA\_SSL.4)**

### **2.6.2.1 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 6.6 of [ST] explains that the TOE provides the function to logout (or terminate) both local and remote user sessions as directed by the user. It explains that all administrative users can issue a logout from a remote administrative session using either the SSH, or the Web GUI interface. The RESTAPI interface is not an interactive interface, but rather every RESTAPI invocation is atomic and each is authenticated. Administrators can also logout from a local console session.



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

Chapter 2, the section entitled "Logging out of QRadar" in the [CC-Guide] includes a note that explains that a user can log out of an active session using the logout or exit commands.

**Component Testing Assurance Activities:** For each method of remote administration, the evaluator shall perform the following tests:

- a) Test 1: The evaluator initiates an interactive local session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.
- b) Test 2: The evaluator initiates an interactive remote session with the TOE. The evaluator then follows the guidance documentation to exit or log off the session and observes that the session has been terminated.

Test 1: The evaluator logged in to the local console and then typed in the command "logout". The evaluator observed that the session ended and a login prompt was presented.

Test 2: The evaluator repeated this test using an SSH session and a Web UI session. The evaluator observed that the session ended and the connection was terminated.

### **2.6.3 TSF-INITIATED SESSION LOCKING (NDcPP22E:FTA\_SSL\_EXT.1)**

#### **2.6.3.1 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 6.6 of [ST] states that the TOE terminates local sessions that have been inactive for an administrator-configured period of time. After termination, administrative authentication is required to access any of the administrative functionality of the TOE.

**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.

Chapter 7, the section entitled "Configuring the local session timeout period" in the [CC-Guide] contains instructions to configure the inactivity timeout period for console sessions. This section explains that on the expiry of the idle timeout, the user is logged out of sessions automatically.



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

a) Test 1: The evaluator follows the guidance documentation to configure several different values for the inactivity time period referenced in the component. For each period configured, the evaluator establishes a local interactive session with the TOE. The evaluator then observes that the session is either locked or terminated after the configured time period. If locking was selected from the component, the evaluator then ensures that reauthentication is needed when trying to unlock the session.

The evaluator followed the guidance to configure the idle timeout periods for the Local Console session and confirmed that the session was terminated after the configured time period. The inactivity time period was configured for periods of 3 minutes and 5 minutes.

## **2.6.4 DEFAULT TOE ACCESS BANNERS (NDCPP22E:FTA\_TAB.1)**

### **2.6.4.1 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 6.6 of the [ST] states the TOE presents a warning banner to the user during login actions prior to completing authentication on the Web GUI, the SSH interface and the local console.

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

Chapter 7, the section entitled "Adding or editing a QRadar login message" in the [CC-Guide] provides the instructions that an administrator can use to configure the message that users see before they log in. This section also explains that the banner is displayed on the local console and remote SSH logins.

**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.



The evaluator configured a banner and verified that the banner was displayed appropriately for console, SSH CLI and Web UI logins. The TOE also provides a banner for HTTPS connections prior to login.

## 2.7 TRUSTED PATH/CHANNELS (FTP)

### 2.7.1 INTER-TSF TRUSTED CHANNEL - PER TD0639 (NDcPP22E:FTP\_ITC.1)

#### 2.7.1.1 NDcPP22E:FTP\_ITC.1.1

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.7.1.2 NDcPP22E:FTP\_ITC.1.2

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

#### 2.7.1.3 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 6.7 of the [ST] states communication with an external audit server (i.e., outbound) will utilize mutually authenticated TLS. The external audit server would be expected to accept TLS communication initiated by the TOE.

**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.



Chapter 6, the sections entitled "Adding forwarding destinations to QRadar" and "Configuring routing rules for event forwarding" in the [CC-Guide] includes steps to configure the TOE to connect to an external syslog server using TLS to protect the communication pathway. These steps include a note that states that the QRadar console will reconnect TLS pathways after an unintentional disconnect.

**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.

The TOE utilizes TLS to protect communications with an external audit server (syslog server).

The evaluator established a successful TLS connection between the TOE and the external audit server (the TOE initiated the connection [Test 2]) and began a packet capture. Within the new TLS session [Test 1], the evaluator observed encrypted application data traffic between the TOE and the test server [Test 3]. Next the evaluator initiated the physical disruption [Test 4] between the TOE and the remote audit server for roughly 30 seconds. Upon reconnection the evaluator observed that the TOE initiated a new TLS negotiation to establish a new TLS session between the TOE and the Gossamer test server. The evaluator also observed audit records appearing on the syslog server after the connection was repaired. The packet capture shows that upon reconnection the TOE does initiate a new TLS handshake.

The evaluator repeated the above steps using a duration of 2 minutes. Upon reconnection, the evaluator observed that the TOE initiated a new TLS negotiation to establish a new TLS session between the TOE and the Gossamer test server. The evaluator also observed audit records appearing on the syslog server. The packet capture shows that upon reconnection the TOE does initiate new TLS handshake.

## **2.7.2 TRUSTED PATH - PER TD0639 (NDCPP22E:FTP\_TRP.1/ADMIN)**

### **2.7.2.1 NDCPP22E:FTP\_TRP.1.1/ADMIN**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### **2.7.2.2 NDCPP22E:FTP\_TRP.1.2/ADMIN**

**TSS Assurance Activities:** None Defined

**Guidance Assurance Activities:** None Defined

**Testing Assurance Activities:** None Defined

### **2.7.2.3 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 6.7 of the [ST] states the TOE provides multiple methods of remote administration. A command line interface is available remotely via an SSH protected channel. Additionally, an HTTPS/TLS connection is available which presents a Web GUI administrative interface and the RESTAPI interface. Administrators must initiate the connection to the TOE from a remote network entity.

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

chapter 1, the section entitled "Supported Cipher Suites" in the [CC-Guide] states that during the STIG hardening, QRadar is configured to use TLS protocol version 1.2 and to use the supported cipher suites only. It also lists all supported cipher suites configured at the time of initial configuration.

Chapter 2, the section entitled "Logging in to Qradar" in the [CC-Guide] details authentication methods for each type of remote administrator.

Chapter 3, the section entitled "Configuring SSH to use public key authentication only" in the [CC-Guide] details configuring the TOE for SSH public key and the section entitled "Configuring the SSHD\_config file" describes how to modify the SSH config file for algorithm selection.

The Web GUI (protected with TLS) and the SSH interfaces are the only remote administrative interfaces identified by the Security Target and by evaluators during testing.

**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.

The TOE supports administration from a local console, ssh, and web GUI.

During the course of testing for the FCS\_TLSS\_EXT.1 and FCS\_SSH\_EXT.1 requirements, the evaluator produced results demonstrating successful HTTPS/TLS and SSH connections. In each case, the following steps were repeated on both the GUI and the CLI.



- a) The evaluator initiated a packet capture of traffic between a remote administrative workstation and the TOE.
- b) The evaluator connected to the TOE and performed a login using an administrator account
- c) The evaluator then terminated the connection by 'Logout' and terminated the packet capture.
- d) The evaluator verified via the packet capture that the connections were successful and that the channel data was encrypted and not sent in plaintext.





### 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 programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 5.

The EAs presented in this section address the CEM work units ADV\_FSP.1-1, ADV\_FSP.1-2, ADV\_FSP.1-3, and ADV\_FSP.1-5.

The EAs are reworded for clarity and interpret the CEM work units such that they will result in more objective and repeatable actions by the evaluator. The EAs in this SD are intended to ensure the evaluators are consistently performing equivalent actions.

The documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any required supplementary information required by the cPP: no additional 'functional specification' documentation is necessary to satisfy the EAs. The interfaces that need to be evaluated are also identified by reference to the EAs listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any required supplementary information defined in the cPP rather than as a separate list specifically for the purposes of CC evaluation. The direct identification of documentation requirements and their assessment as part of the EAs for each SFR also means that the tracing required in ADV\_FSP.1.2D (work units ADV\_FSP.1-4, ADV\_FSP.1-6 and ADV\_FSP.1-7) is treated as implicit and no separate mapping information is required for this element.

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'.

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.

As identified throughout this AAR, the [CC-Guide] provides instructions for configuring the TOE's cryptographic security functions. The [CC-Guide] provides instructions for configuring the cryptographic algorithms and parameters used for the evaluated configuration. The [CC-Guide] is clear that no other cryptographic configuration has been evaluated or tested. The section entitled "Supported cipher suites" instructs administrators to configure TLS to support only the ciphers that are required to complete the evaluation. There are also warnings and notes throughout the [CC-Guide] regarding use of functions that are and are not allowed in the evaluated configuration. There are also specific settings identified that must be enabled or disabled in order to remain CC compliant. The process for updating the TOE is described above in NDcPP22e:FPT\_TUD\_EXT.1.

### 3.2.2 PREPARATIVE PROCEDURES (AGD\_PRE.1)

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

It is noted that specific requirements for Preparative Procedures are defined in [SD] for distributed TOEs as part of the Evaluation Activities for FCO\_CPC\_EXT.1 and FTP\_TRP.1(2)/Join.

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



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

In addition, the evaluator performs the EAs specified below.

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

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

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

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

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

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

The preparative procedures must

- a) include instructions to provide a protected administrative capability; and
- b) identify TOE passwords that have default values associated with them and instructions shall be provided for how these can be changed.

The evaluation team had the following documents to use when configuring the TOE:

- IBM QRadar Version 7.5, Common Criteria for NIAP Revision 1.0

The completeness of this documentation is addressed by its 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.

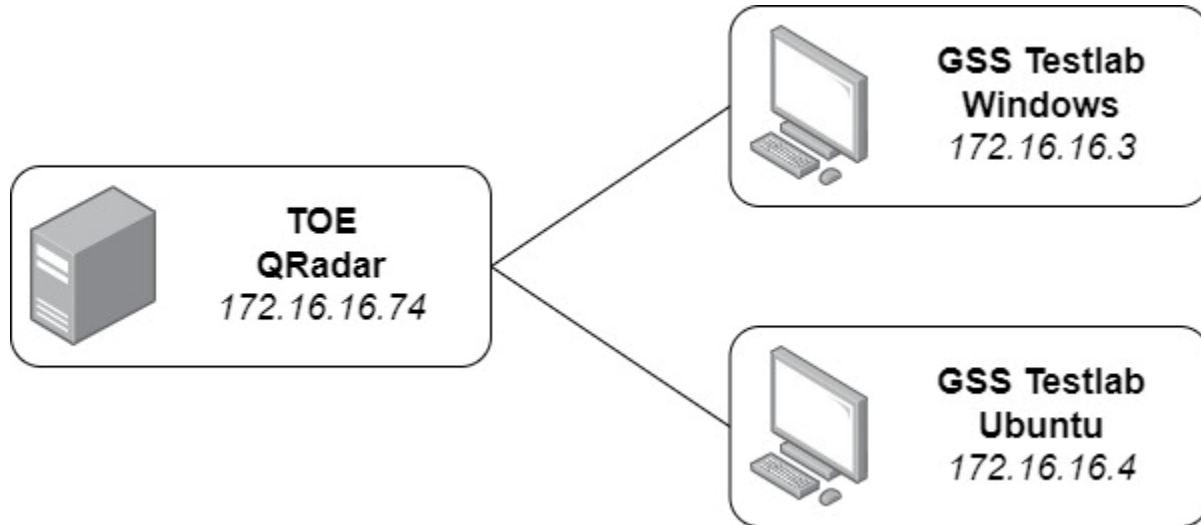


Figure 1 Test Network

**TOE Platforms:**

- IBM QRadar Security Intelligence Platform Version 7.5 running on RedHat Enterprise Linux v7.9.

**Supporting Products:**

- None

**Supporting Software:**

The Gossamer Test servers utilized both a windows and Ubuntu environment. The Windows supporting software included the following.

- Windows 10.0
- Wireshark several versions including 3.0.6 and 4.0.4
- Windows SSH Client - Putty version 0.71, 0.73 & 0.74 (used to connect to device console and SSH)
- Nmap version 7.92 (Windows)

The Gossamer Test servers with an Ubuntu environment acted as platforms to initiate testing. The test servers also acted as a syslog server.



- OpenSSL 1.0.2g-fips
- Nmap version 7.01 (Linux)
- Openssh client version 8.2p1
- Big Packet Putty, Openssh-client version 7.2
- Rsyslog version 8.16.0
- Tcpdump version 4.9.3
- Libpcap version 1.7.4
- Stunnel 5.30

### 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 components<sup>7</sup> 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 June 8, 2023. The search was conducted with the following search terms: "QRadar", "Openssl", "seim", "ibmfipsjce", "TLS", "SSH", "Dell 3128C", "Xeon Gold 5118".