Assurance Activity Report

VID 10785

17-3625-R-0025 V1.1

September 13, 2017

Pulse Policy Secure Security Target

Version 1.0

September 5, 2017

Evaluated by:

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

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

TOE Evaluation Sponsor and Developer
Pulse Secure, LLC
2700 Zanker Road, Suite 200
San Jose, CA 95134

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

Evaluation Personnel:
Michael C. Baron
Ryan Day

Applicable Common Criteria Protection Profile
collaborative Protection Profile for Network Devices, Version 1.0, Feb. 27, 2015

Table of Contents

1 Overview

This document presents evaluation results of the Pulse Policy Secure against the collaborative Protection Profile for Network Devices, Version 1.0, February 27, 2015. This document contains a description of the assurance activities and associated results as performed by the UL Verification Services Inc., an accredited Common Criteria Testing Laboratory. This Evaluation was conducted with the oversight and guidance provided by the National Information Assurance Partnership and its contributors.

2 SFR Assurance Activities and Results

2.1 FAU_GEN.1 Audit Data Generation

Guidance Documentation

The evaluator shall check the guidance documentation and ensure that it lists all of the auditable events and provides a format for audit records. Each audit record format type must be covered, along with a brief description of each field. The evaluator shall check to make sure that every audit event type mandated by the cPP is described and that the description of the fields contains the information required in FAU_GEN1.2, and the additional information specified in the table of audit events.

The evaluator shall also make a determination of the administrative actions that are relevant in the context of the cPP. 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 security relevant with respect to the cPP. The evaluator may perform this activity as part of the activities associated with ensuring that the corresponding guidance documentation satisfies the requirements related to it.

Tests

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

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

2.1.1 TSS

None

2.1.2 Guidance

The evaluator examined the audit logs that were produced during the testing activities performed to verify all required logs were present, and that their format corresponded to the audit record format examples provided in the guidance documentation. The evaluator confirmed relevant audit records for all SFRs identified in [ST] Section 6.1.1.1 were present, and each audit record contained the minimum content requirements as required and described by FAU_GEN.1.2. [AGD] Section 6.1 provides a mapping of the FAU_GEN.1.2 requirements to the audit record format as generated by the TOE. In addition, the evaluator verified the TOE generated the additional audit record requirements as described in [PP] Section A.1 and B.1, for the appropriate optional and selection-based SFRs defined in the [ST].

2.1.3 Testing

The evaluator examined the audit logs that were produced during the testing activities performed to verify all required logs were present, and that their format corresponded to the audit record format examples provided in the guidance documentation. The evaluator confirmed relevant audit records for all SFRs identified in [ST] Section 6.1.1.1 were present, and each audit record contained the minimum content requirements as required and described by FAU_GEN.1.2. [AGD] Section 6.1 provides a mapping of the FAU_GEN.1.2 requirements to the audit record format as generated by the TOE. In addition, the evaluator verified the TOE generated the additional audit record requirements as described in [PP] Section A.1 and B.1, for the appropriate optional and selection-based SFRs defined in the [ST].

2.2 FAU_GEN.2 User identity association

This activity should be accomplished in conjunction with the testing of FAU_GEN.1.1.

2.2.1 TSS

None

2.2.2 Guidance

None

2.2.3 Testing

Test assurance activities for this element were performed during the testing assurance activities performed for FAU_GEN.1.1. The evaluator verified for audit events resulting from actions of identified users, the TSF associated each of the auditable events with the identity of the user that caused the event.

2.3 FAU_STG.1 Protected audit trail storage

TSS

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

Guidance Documentation

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

Tests

The evaluator shall perform the following tests:

a) Test 1: The evaluator shall access the audit trail as an unauthorized administrator and attempt to modify and delete the audit records. The evaluator shall verify that these attempts fail.

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

2.3.1 TSS

[ST] Section 7.1.2 “Audit Storage” states that 200MB of local storage space is allocated for locally stored audit data and that this can be configured up to 500MB.

[ST] Section 7.1.2 “Audit Storage” describes how local audit records are protected against unauthorized modification/deletion by a restrictive administrative interface.

[ST] Section 7.1.2 “Audit Storage” describes that the admin must be positively identified and authenticated to the TOE prior to deleting any audit records.

2.3.2 Guidance

[AGD] Section 5.13.4 provides the description of how the TSF protects the locally stored audit records from unauthorized modification/deletion. To modify/delete the locally stored audit records, an administrative user must be authenticated to the TSF.

2.3.3 Testing

The evaluator created a non-admin user account on the TOE and attempted to login to the administrative web GUI management portal to attempt to access the audit trail using this non-administrative account, then attempted to login to this administrative web GUI portal using a username and password that was not configured on the TOE. The evaluator verified both attempts failed to access locally stored audit records. As an unauthorized user, the evaluator used ‘wget’ in an attempt to retrieve web resources of the locally stored audit records as identified by URLs found during authorized access to the locally stored audit records. The evaluator verified the TOE prevents non-administrative and unauthorized users from accessing, viewing, modifying, or deleting the audit records stored locally on the TOE.

The evaluator then used an authorized administrative account and verified the admin could view and delete locally stored audit records. The TOE does not provide the ability to modify the locally stored audit records, even by an authenticated admin.

2.4 FAU_STG_EXT.1 Protected audit event storage

TSS

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

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

If the TOE complies with FAU_STG_EXT.2 the evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2 are correct when performing the tests for FAU_STG_EXT.1.3.

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

Guidance Documentation

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

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

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

Tests

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

a) 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).

b) 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)

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

2.4.1 TSS

[ST] Section 7.1.2 states that audit data are transferred to the external audit server via the Syslog protocol, utilizing the TLS Trusted Channel, according to RFCs 5424 and 5425 respectively.

[ST] Section 7.1.2 states that 200MB of local storage space is allocated for locally stored audit data, and that this can be configured up to 500MB.

[ST] Section 7.1.2 contains the statement of what happens when the local audit data store is full (log rotation feature). In addition, this section describes the 'low audit storage' warning audit record that is presented.

[ST] Section 7.1.2 describes that the admin must be authenticated to the TOE prior to deleting any audit records.

[ST] Section 7.1.2 details the behavior of the TOE when the storage space for audit data is full. Since the option 'overwrite previous audit record' was selected this description included an outline of the rule for overwriting audit data (the current/full log file is deleted; a new log file is created and audit logs are written to it until full) when the local storage space for audit data is full.

2.4.2 Guidance

[AGD] Section 5.13 describes the configuration items required to establish the Trusted Channel to the remote audit server. The following configuration steps need to be performed:

- Install a 'Trusted Server CA' used to authenticate the remote audit server's certificate. ([AGD] Section 5.9 contains the guidance on doing this.)
- Install a 'Client auth Certificate' which is sent to the remote audit server for TLS mutual authentication. ([AGD] Section 5.11.1 contains the guidance on doing this.)
- Install a 'Trusted Client CA' to validate the 'Client Auth. Certificate'. ([AGD] Section 5.8 contains the guidance on doing this.)

[AGD] Section 5..13 contains the remaining guidance for configuring the TOE to establish the Trusted Channel to the audit server. These steps include configuring the FQDN or the IP address of the remote audit server (needs to match the 'Common Name' field as configured for the audit server's certificate), configuring the communication protocol to use the TLS protocol (TLS as configured in [AGD] Section 5.12 "Enable NDcPP Mode"), and selecting the TOE's client certificate to send to the remote audit server for mutual authentication (client certificate as configured in [AGD] Section 5.8).

[AGD] Section 6.3 "Log Sent to Remote Audit Storage" contains the description of the relationship between the local and remote audit records. This section states that the logs sent to the remote audit server are copies of the local audit records except for the formatting. This section provides a table the maps the fields that are different between the local and remote audit records.

[AGD] Section 5.10 states that the TOE supports both RSA and ECC certificates. [AGD] Section 5.10.1 states that only RSA 2048 and 3072 are supported and that P-256 and P-384 curves are supported for ECC certificates.

2.4.3 Testing

The evaluator verified the TOE transmits all audit data to the remote audit server via the mutually authenticated, TLS Trusted Channel as defined in [ST] Section 6.1.7.1. The evaluator ran a packet capture using Wireshark while establishing a session between the TOE and remote audit server and performing an action that was known to generate a particular audit record. The evaluator verified the audit record generated was stored locally on the TOE and was transmitted to the remote audit server protected by TLS on the wire.

The evaluator verified the TSF performed the behavior as defined in [ST] Section 7.1.2 when the local audit storage was exhausted. The TSF uses a log rotation feature which divides the configured local audit storage into two files (an active file which the TSF writes audit records to until full; and an inactive file that contains the oldest stored audit records). For example, if the local audit record storage is configured at 200MB, the TSF would write 100MB of audit records to the active file, then create a new file which now becomes the 'active' file, and then writes new audit records to this new 'active' file until 100MB was reached. Once this second file is full, the TSF deletes the oldest audit record file, switches the current 'active' record file to the old position, creates a new 'active' audit record file and begins writing to it. This is the audit log rotation feature. The evaluator configured the local audit storage to 1MB (meaning it would take over 2MB to overwrite the oldest audit record stored), cleared the locally stored audit records, generated and noted the first new audit record that was locally stored, then generated slightly over 1MB of audit data and verified that the TSF generated an audit record warning of low local storage at 90% of 1MB, then the TSF generated an audit record warning that the local audit storage was full after 1MB of audit records were generated. This meant the first locally stored audit record file was full and writing of additional audit records was switched to the second audit record file. This process repeated a second and final time. Once a total of a little over 2MB of local audit record data was generated, the evaluator verified in the export file of the locally stored audit records that the TSF wrote over the first generated audit record (oldest log file), thus verifying the audit log rotation feature was operating as expected. This behavior was consistent with the description in [ST] Section 7.1.2.

2.5 FAU_STG_EXT.3 Display warning for local storage space

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

TSS

The evaluator shall examine the TSS to ensure that it details how the user is warned before the local storage for audit data is full.

Guidance Documentation

The evaluator shall also ensure that the guidance documentation describes how the user is warned before the local storage for audit data is full and how this warning is displayed or stored (since there is no guarantee that an administrator session is running at the time the warning is issued, it is probably stored in the log files). The description in the guidance documentation shall correspond to the description in the TSS.

Tests

The evaluator shall verify that a warning is issued by the TOE before the local storage space for audit data is full.

2.5.1 TSS

[ST] Section 7.1.2 states that an audit log warning in the audit logs and a notice in the HTTPs GUI is presented when the current audit file reaches 90% of the local audit storage capacity.

2.5.2 Guidance

[ST] Section 7.1.2 describes the warning for low local storage space for the audit records.

[AGD] Section 5.10.4 states that an audit record is generated when the local storage for the audit records is at 10% of the value set for the “Max Log Size” (i.e. 90% full).

[AGD] Section 6.2.20 provides an audit record format for the audit record that is generated when the local storage space for audit records is low.

2.5.3 Testing

During testing for FAU_STG_EXT.1.2 and FAU_STG_EXT.1.3, the evaluator verified that the TSF generated an audit record warning the local audit record storage space was running low.

2.6 FCS_CKM.1 Cryptographic Key Generation

TSS

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

Guidance Documentation

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this PP.

Tests

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

Key Generation for FIPS PUB 186-4 RSA Schemes

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

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

a) Random Primes:
   - Provable primes
   - Probable primes

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

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

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

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

and two ways to generate the cryptographic group generator g:

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

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

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

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

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

For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values

generated by the TSF with those generated from a known good implementation. Verification must also confirm

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

for each FFC parameter set and key pair.

2.6.1 TSS

[ST] Section 7.2.1 identifies RSA 2048 and 3072-bit keys and ECDSA p-256 and p-384 curves as being supported by the TOE for TLS client and TLS server authentication.

2.6.2 Guidance

[ST] Section 6.1.2.1 selected the following asymmetric cryptographic key generation schemes:

- RSA Schemes
  - [ST] Section 7.2.1 describes key generation for RSA 2048-bit and 3072-bit keys for TLS client and TLS server authentication for ciphersuites utilizing RSA algorithms.
- ECC Schemes
  - [ST] Section 7.2.1 describes key generation for ECDSA p-256 and p-384 keys for TLS client and TLS server authentication for ciphersuites utilizing ECHDE and ECDSA algorithms.

RSA and ECDSA asymmetric key-generation is performed during session establishment for TLS connections. [AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance for configuring TLS to use the key-generation schemes and associated key sizes and curves selected in the [ST]. This section instructs the admin to enable ‘NDcPP Mode’ and additional configuration boxes, which enables the TOE to use the TLS ciphersuites that are selected in the [ST]. These TLS server ciphersuites allow the following asymmetric key-generation schemes, which generate the following key sizes and curves:

- RSA
  - 2048 and 3072-bit key sizes
- ECDSA curves
  - secp256r1 and secp384r1

RSA and ECDSA asymmetric cryptographic key-generation also occurs when creating a Certificate Signing Request. [AGD] Section 5.10.1 provides instructions on generating a Certificate Request Message for requesting a signed certificate from a CA. The ‘Key Type’ and ‘Key Length’ fields allow the Security Administrator to generate an RSA or ECDSA certificate with 2048 or 3072-bit key sizes for RSA and secp256r1 or secp384r1 curves for ECDSA.

2.6.3 Testing

The evaluator reviewed the CAVP website and verified the certificates claimed in [ST] Section 7.2.2 were correct, matched the environment of the TOE, and covered all functions of the SFRs. These included the following CAVP certificate numbers for the TSF:

SFR	Description	Cert
FCS_CKM.1	RSA 2048/3072-bit Key Generation	2345
	ECDSA P-256/P-384 Key Generation	1026
FCS_CKM.2	Elliptic Curve Diffie-Hellman SP800-56A Key Agreement	1270
FCS_COP.1(1)	AES 128/256-bit CBC/GCM Encryption/Decryption	4334
FCS_COP.1(2)	RSA 2048/3072-bit Signature Generation, Signature Verification with SHA-1/256/384/512	2345
	ECDSA P-256/P-384 Signature Generation, Signature Verification	1026

SFR	Description	Cert
	with SHA-1/256/384/512	
FCS_COP.1(3)	SHA-1/256/384/512	3577
FCS_COP.1(4)	HMAC-SHA-1/256/384	2880
FCS_RBG_EXT.1	CTR_DRBG using AES 256	1384

2.7 FCS_CKM.2 Cryptographic Key Establishment

TSS

The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Guidance Documentation

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

Tests

Key Establishment Schemes

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

SP800-56A Key Establishment Schemes

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

Function Test

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

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

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

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

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

Validity Test

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

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

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

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following assurance activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS- OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following assurance activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

a) To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTS- OAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the

test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

b) The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM- KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

2.7.1 TSS

The [ST] claims RSA and Elliptic Curve based schemes are utilized for cryptographic key generation. Elliptic Curve schemes are used for key establishment.

Section 7.2.1 states that generation of 2048 and 3072 bit RSA keys are used for both client and server authentication purposes and for RSA key encapsulation when RSA-based ciphersuites are negotiated in TLS sessions (TRP and ITC)

[ST] Section 7.2.1 “Cryptographic Key Generation” states that ECC based key establishment protocols are negotiated when ECHDE based TLS ciphersuites have been negotiated.

In addition, [ST] Section 7.2.1 states that RSA based key encapsulation is utilized when RSA based TLS ciphersuites are negotiated.

2.7.2 Guidance

[ST] Section 6.1.2.2 selected the following cryptographic key establishment schemes:

- Elliptic Curve Based Schemes: [ST] Section 7.2.1 describes ECDHE key agreement for TLS client and TLS server session establishment.

[AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance for configuring TLS to use the key-establishment schemes and associated curves selected in the [ST]. This section instructs the admin to enable ‘NDcPP Mode’ and additional configuration boxes which enables the TOE to use the TLS ciphersuites that are selected in the [ST]. These TLS server ciphersuites allow the following key-establishment schemes, which generate the following curves:

- ECDHE curves
  - Secp256r1 and secp384r1

2.7.3 Testing

The evaluator verified the CAVP certificate for CVL Cert. #1270 was issued and was applicable to the TSF and the TOE’s OE.

2.8 FCS_CKM.4 Cryptographic Key Destruction

TSS

The evaluator shall check to ensure the TSS lists each type of plaintext key material and its origin and storage location.

The evaluator shall verify that the TSS describes when each type of key material is cleared (for example, on system power off, on wipe function, on disconnection of trusted channels, when no longer needed by the trusted channel per the protocol, etc.).

The evaluator shall also verify that, for each type of key, the type of clearing procedure that is performed (cryptographic erase, overwrite with zeros, overwrite with random pattern, or block erase) is listed. If different types of memory are used to store the materials to be protected, the evaluator shall check to ensure that the TSS describes the clearing procedure in terms of the memory in which the data are stored (for example, "secret keys stored on flash are cleared by overwriting once with zeros, while secret keys stored on the internal persistent storage device are cleared by overwriting three times with a random pattern that is changed before each write").

2.8.1 TSS

[ST] Section 7.2.1 states the following:

Persistent Keys:

- HTTPs/TLS Private Host Key
  - Stored in:
    - Hard Disk (Non-volatile)
      - Zerotization method:
        - Overwriting the file location with data from /dev/urandom three times. Each overwrite calls /dev/urandom ensuring that a different pseudo random pattern is used each time.
    - RAM (Volatile)
      - Zerotization method:
        - Writing zeros to the memory location three times and performing a read verify to ensure that the memory location was set to all zeros. If the read verify fails, the TSF repeats the zeroization process.

- Syslog/TLS Private Client Key
  - Stored in:
    - Hard Disk (Non-volatile)
      - Zerotization method:
        - Overwriting the file location with data from /dev/urandom three times. Each overwrite calls /dev/urandom ensuring that a different pseudo random pattern is used each time.
    - RAM (Volatile)
      - Zerotization method:
        - Writing zeros to the memory location three times and performing a read verify to ensure that the memory location was set to all zeros. If the read verify fails, the TSF repeats the zeroization process.
- HAWK Secret Key
  - Stored in:
    - Hard Disk (Non-volatile)
      - Zerotization method:
        - Overwriting the location of the previous key with the new value of the key or instructing the filesystem to destroy the abstraction representing the key (deletion of the config file).
    - RAM (Volatile)
      - Zerotization method:
        - Writing zeros to the memory location three times and performing a read verify to ensure that the memory location was set to all

zeros. If the read verify fails, the TSF repeats the zeroization process.

Ephemeral Keys:

- TLS Session Keys (HTTPS/TLS Trusted Path and TLS Trusted Channel to Syslog):
  - Stored in:
    - RAM (Volatile)
      - Zerotization method:
        - Writing zeros to the memory location three times and performing a read verify to ensure that the memory location was set to all zeros. If the read verify fails, the TSF repeats the zeroization process.
- DRBG State:
  - Stored in:
    - RAM (Volatile)
      - Zerotization method:
        - Writing zeros to the memory location three times and performing a read verify to ensure that the memory location was set to all zeros. If the read verify fails, the TSF repeats the zeroization process.

2.8.2 Guidance

None

2.8.3 Testing

None

2.9 FCS_COP.1(1) Cryptographic Operation (AES Data Encryption/Decryption)

Tests

AES-CBC Known Answer Tests

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

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

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

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

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

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

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

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

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

AES-CBC Multi-Block Message Test

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

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

AES-CBC Monte Carlo Tests

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

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

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

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

AES-GCM Test

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

128 bit and 256 bit keys

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

b) **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.

c) **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.

2.9.1 TSS

None

2.9.2 Guidance

None

2.9.3 Testing

The evaluator verified the CAVP certificate for AES Cert. #4334 was issued and was applicable to the TSF and the TOE's OE.

2.10 FCS_COP.1(2) Cryptographic Operation (Signature Generation and Verification)

Tests

ECDSA Algorithm Tests

ECDSA FIPS 186-4 Signature Generation Test

For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and

S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.

ECDSA FIPS 186-4 Signature Verification Test

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

(message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

Signature Generation Test

The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages.

The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

Signature Verification Test

The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

The evaluator shall use these test vectors to emulate the signature verification test using the corresponding parameters and verify that the TOE detects these errors.

2.10.1 TSS

None

2.10.2 Guidance

None

2.10.3 Testing

The evaluator verified the CAVP certificates for RSA Cert. #2345 and ECDSA Cert. #1026 were issued and were applicable to the TSF and the TOE's OE.

2.11 FCS_COP.1(3) Cryptographic Operation (Hash Algorithm)

TSS

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

Guidance Documentation

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

Tests

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

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

Short Messages Test – Bit-oriented Mode

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

Short Messages Test – Byte-oriented Mode

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

Selected Long Messages Test – Bit-oriented Mode

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

Selected Long Messages Test – Byte-oriented Mode

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

Pseudorandomly Generated Messages Test

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

2.11.1 TSS

[ST] Section 7.2.2 contains Table 11 which itemizes each hashing algorithm specified in FCS_COP.1.1(3) (SHA-1/256/384/512) and their usage in association with other TSF cryptographic functions, including HMAC, digital signatures, file integrity checking and password obfuscation. SHA usage in HMAC is further described in [ST] Section 7.2.2, Table 12.

2.11.2 Guidance

[ST] Section 6.1.2.6, FCS_COP.1.1(3) lists the hashing algorithms utilized by the TSF.

[AGD] Section 8 “Hash Functions” provides information regarding the configuration of the [ST] selected hashing algorithms. This section states that TLS uses appropriate hashing function (either SHA-1, SHA-256, SHA-384 or SHA-512) based on the negotiated ciphersuite in the TLS connections. File integrity checks utilize the SHA-256 hash for self-tests during boot-up; no configuration is required. The Security Administrator’s password is obfuscated using SHA-256; no configuration is required.

The NDcPP ciphersuites contain hashing algorithms available for use by the TOE once configured. [AGD] Section 5.12 “Enable NDcPP Mode” contains the instructions for enabling “NDcPP Mode” which configures the [ST] selected ciphersuites for TLS and TLS Protocol versions on the TOE for both the “inbound” TLS connections (TLS Server configuration) and the “outbound” TLS connections (TLS Client configuration).

2.11.3 Testing

The evaluator verified the CAVP certificate for SHS Cert. #3577 was issued and was applicable to the TSF and the TOE's OE.

2.12 FCS_COP.1(4) Cryptographic Operation (Keyed Hash Algorithm)

TSS

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

Tests

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

2.12.1 TSS

Table 12 in [ST] Section 7.2.2 specifies the hash function utilized, key length, block size and output MAC length (digest), for each HMAC function. The stated hash algorithms are consistent with FCS_COP.1.1(3). The stated HMAC algorithms are consistent with FCS_COP.1.1(4).

2.12.2 Guidance

None

2.12.3 Testing

The evaluator verified the CAVP certificate for HMAC Cert. #2880 was issued and was applicable to the TSF and the TOE's OE.

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

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

Tests

The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the guidance documentation contains appropriate instructions for configuring the RNG functionality.

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

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input,

nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

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

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

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

Personalization string: The length of the personalization string must be <= seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

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

2.13.1 TSS

This work unit is presented in the [ENT] document. Please see the rationale in [ENT] for further details.

2.13.2 Guidance

None

2.13.3 Testing

The evaluator verified the CAVP certificate for DRBG Cert. #1384 was issued and was applicable to the TSF and the TOE's OE.

2.14 FCS_HTTPS_EXT.1 HTTPS Protocol

**FCS_HTTPS_EXT.1 was modified according to "TD0125: NIT Technical Decision for Checking validity of peer certificates for HTTPS servers".*

TSS

The evaluator shall check that the TSS describes how peer authentication is implemented when HTTPS protocol is used.

Tests

The evaluator shall perform the following tests:

a) Test 1: The evaluator shall attempt to establish an HTTPS connection with a web server, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

Other tests are performed in conjunction with the TLS evaluation activities.

Certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1, and the evaluator shall perform the following test:

b) Test 2: If 'the peer presents a valid certificate during handshake' is selected in FCS_HTTPS_EXT.1.3, then certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1 if HTTPS is used for FTP_TRP.1 or FTP_ITC.1.

2.14.1 TSS

ST] Section 7.2.3 states that when the TOE acts as a TLS client (TOE to Pulse One management server), the TSF will only establish the connection if the peer certificate (Pulse One x509 cert) is successfully authenticated according to the x509 authentication rule set described in [ST] Section 7.3.5.

When acting as a TLS server (Admin to TOE connection), peer certificates are not implemented. The administrative web GUI provides an admin authentication portal using username/password credentials only.

2.14.2 Guidance

None

2.14.3 Testing

Test 1:

Testing for FCS_TLSS_EXT.1.1 – Test 1, satisfied the requirements for FCS_HTTPS_EXT.1 – Test 1. In FCS_TLSS_EXT.1.1 – Test 1, the evaluator used the custom TLS testing tool to establish a TLS connection to the HTTPS/TLS Server TSF. The TLS testing tool ran through each ciphersuite selected in [ST] Section 6.1.2.12 and verified a successful session establishment using each of the mandatory and selected ciphersuites. These included the following ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

Testing for FCS_TLSS_EXT.1.1 – Test 2, 3 and 4, ran packet captures during performance of those tests. These tests verify that the protocol on the wire was negotiated as TLS.

Test 2:
The testing requirements for FCS_HTTPS_EXT.1 - Test 2 were revised by 'TD0125: NIT Technical Decision for Checking validity of peer certificates for HTTPS servers'. This test is not applicable since the [ST] did not select 'the peer presents a valid certificate during handshake'.

2.15 FCS_TLSC_EXT.1 Extended: TLS Client Protocol

FCS_TLSC_EXT.1.1

TSS

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.

Guidance Documentation

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

Tests

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 in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

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

Test 3: The evaluator shall use a client to send a key exchange message in the TLS connection that does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE either sends an alert after receiving the client's ChangeCipherSpec message or Finished message; or terminates the connection after receiving the client's ChangeCipherSpec message or Finished message.

**Test 3 above was modified by "TD0143: NIT Technical Decision for Failure testing for TLS session establishment in NDcPP and FWcPP".*

Test 4: The evaluator shall configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS_TLSS_EXT.1.1 or FCS_TLSS_EXT.2.1 can be used as a substitute for this test.

Test 5: The evaluator 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 (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.

b) 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.

c) 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.

d) Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message. 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 5d was modified by "TD0165: NIT Technical Decision for Sending the ServerKeyExchange message when using RSA".*

e) Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.

f) Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

FCS_TLSC_EXT.1.2

TSS

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 Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

Guidance Documentation

The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Tests

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

a) Test 1: The evaluator shall present 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. The evaluator shall verify that the connection fails.

b) Test 2: 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.

c) Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contains the SAN extension. The evaluator shall verify that the connection succeeds.

d) Test 4: 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.

e) Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

    1) 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) 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. 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.come) and verify that the connection fails.

f) Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

g) Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

FCS_TLSC_EXT.1.3

Tests

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates, and show that the certificate is not validated and the trusted channel is not established.

FCS_TLSC_EXT.1.4

TSS

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

Guidance Documentation

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

Tests

Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

2.15.1 TSS

FCS_TLSC_EXT.1.1

[ST] Section 6.1.2.10 includes the ciphersuite selection for FCS_TLSC_EXT.1. [ST] Section 6.1.2.11 includes the ciphersuite selection for FCS_TLSC_EXT.2.

[ST] Section 7.2.4 describes the behavior the TLS Client TSF and lists the ciphersuites consistently with [ST] Sections 6.1.2.10 and 6.1.2.11.

FCS_TLSC_EXT.1.2

[ST] Section 7.2.4 describes the behavior the TLS Client TSF. This section states that certificate pinning is not supported by the TOE. In addition, this section states that the TOE validates the incoming server certificate using the DNS name or IP address of the server, in addition to the validity checks described in [ST] Section 7.3.4.

[ST] Section 7.3.5 further describes the certificate authentication process of the TSF when a server certificate is presented to the TOE's TLS Client TSF. This section states that the TSF verifies the server certificate in the following manner:

- When the server is specified using a domain name, the TSF verifies that the domain name matches a Subject Alternative Name DNS Name field in the certificate using exact or wildcard matching specified in Section 3.1 of RFC 2818. If the certificate does not contain any Subject Alternative Name fields, the TSF matches the domain name against the Common Name in the certificate.
- When the server is specified using an IP address, the TSF verifies that the IP address exactly matches a Subject Alterative Name IP Address field in the certificate using the rules specified in Section 3.1 of RFC 2818. If the certificate does not contain any Subject Alternative Name fields, the TSF matches the IP address against the Common Name in the certificate.

FCS_TLSC_EXT.1.4

[ST] Section 7.2.4 states that secp256r1 and secp384r1 Elliptic Curves are supported by the TLS Client TSF and are configurable by the Security Administrator (via configuration of the supported ciphersuites).

The curves described are consistent with the curves selected for FCS_TLS_EXT.1.4 in [ST] Section 6.1.2.10.

2.15.2 Guidance

[AGD] Section 5.12 “Enable NDcPP Mode” provides guidance on configuring the TOE to use the TLS version and ciphersuites selected and mandated by FCS_TLSC_EXT.2.1 in [ST] Section 6.1.2.11. [AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance on configuring TLS to accept only TLS v1.1 and v1.2.

[ST] Section 7.3.5 describes the fields in the peer certificate that are compared to the reference identifiers. The fields are checked against the reference identifiers depending on how the TOE was configured. If the remote audit server destination was configured with a FQDN, then the TSF compares the SAN (Subject Alternative Name) DNS field in the peer certificate to the FQDN configured for the remote audit server. If the peer certificate does not contain a SAN field, then the comparison is made against the FQDN in the ‘Common Name’ field of the peer certificate. Similarly, if the remote audit server destination was configured with an IP address, the TSF compares the configured IP address to the SAN IP field or to the Common Name field, if no SAN IP field exists in the peer certificate. All of this checking is performed automatically after the remote audit server destination is configured by the Security Admin (i.e., with a FQDN or an IP address).

[AGD] Section 5.17 provides the guidance for configuring the host name of the Pulse One management server that the TOE’s TLS client will connect to.

The TSS does not indicate that the Supported Elliptic Curves Extension must be configured.

2.15.3 Testing

Note: FCS_TLSC_EXT.2 tested the Trusted Channel TSF between the TOE and Pulse One remote management server. Acting in place of the Pulse One management server for testing was either OpenSSL’s s_server utility or UL’s custom TLS tool. The descriptions below indicate which tool was utilized for each particular test performed.

FCS_TLSC_EXT.1.1:

Test 1:

Using OpenSSL’s s_server utility, the evaluator verified the TOE established a TLS session utilizing each of the ciphersuites listed in [ST] Section 6.1.2.10. This included the following ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

An RSA 2048-bit server certificate was used when negotiating RSA based ciphersuites. An ECDSA P-384 server certificate was utilized when negotiating ECDSA based ciphersuites. The OpenSSL s_server utility shows the connections being established successfully and receiving at least one audit record from the TOE. Packet captures verify protocol session establishment and parameters utilized.

Test 2:

Objective 1: This test was satisfied through testing of FCS_TLSC_EXT.1.1 – Test 1. Test 1 is described immediately above. This test verified the TOE accepts a server certificate with ‘TLS Server Authentication’ purpose in the extendedKeyUsage field and that the connection was successfully established.

Objective 2: The evaluator verified the TOE rejected a server certificate that does not have the ‘TLS Server Authentication’ purpose in the extendedKeyUsage field by presenting the compromised certificate

to the TSF using the OpenSSL s_server tool and configuring the TOE's Pulse One remote management TSF to attempt to connect to s_server instead of the actual Pulse One remote management server. Packet captures confirm that s_server sent the compromised certificate and that the TSF denied the connection attempt by sending a fatal alert message.

Test 3:

Using a custom TLS testing tool, the evaluator verified the TOE rejected a TLS connection attempt with a server that sent a server certificate that does not match the server-selected ciphersuite. Packet captures confirm the custom TLS testing tool choose an RSA based ciphersuite in the server_hello message, but then sent an ECDSA certificate in the server's 'certificate' message. The TSF sent a TCP FIN segment, denying the establishment of the TLS session.

Test 4:

Using a custom TLS testing tool, the evaluator verified the TOE did not establish a TLS connection using the TLS_NULL_WITH_NULL_NULL ciphersuite. The test tool sent the null ciphersuite in the server_hello message; the TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings.

Test 5:

Objective a): Using the custom TLS testing tool, the evaluator verified the client rejected non-supported TLS versions by attempting to establish a session using TLS version 1.3. The TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings.

Objective b): Using the custom TLS testing tool, the evaluator verified the TOE rejected a TLS connection attempt when using a modified nonce in the Server_Hello handshake message. The TSF sent a fatal alert message for a 'decrypt error' immediately after receiving the server's 'server key exchange' message. The TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings. For greater test coverage, the evaluator attempted the test using a ciphersuite with RSA key exchange and verified the server rejected (sent a TCP FIN segment) after receiving the client's 'client key exchange' message. Packet captures confirm these findings.

Objective c): Using the custom TLS testing tool, the evaluator verified the TOE rejected a connection attempt with a server-selected ciphersuite that was not present in the client_hello message sent from the TOE. The TSF sent a fatal alert message after receiving the server's 'server_hello' message. Packet captures confirm these findings.

Objective d): Using the custom TLS testing tool, the evaluator verified the TOE rejected a TLS connection attempt with a modified signature block in the Server's 'Key Exchange' message when using an ECDHE ciphersuite. The TSF sent a fatal alert message after receiving the Server Key Exchange message. Packet captures confirm these findings.

Objective e): Using the custom TLS testing tool, the evaluator verified the TOE sent a fatal alert upon receipt of a modified Server Finished handshake message, and the TOE did not send application data. The TSF sent an encrypted fatal alert message after receiving the encrypted server_finished message, then immediately sent a TCP FIN segment. Packet captures confirm these findings.

Objective f): Using the custom TLS testing tool, the evaluator verified that the TSF denied the connection when sending a garbled message from the Server after the Server issued the ChangeCipherSpec message. The TSF sent an encrypted fatal alert message after receiving the garbled message from the test tool, then immediately sent a TCP RST,ACK segment. Packet captures confirm these findings.

FCS_TLSC_EXT.1.2:

Test 1:

Using OpenSSL's s_server utility, the evaluator verified the TOE denied a server certificate with a reference identifier (CN field) that did not match the reference identifier as configured on the TOE. The evaluator configured the server certificate with a CN of "*badexpectedidentifier.server.local"* (a SAN was not included in the certificate) while the TOE was configured with an expected identifier of

"itsupport.server.local". The evaluator verified the TSF sent a fatal alert message and did not establish the session. Packet captures confirm these findings.

Test 2:

Using OpenSSL's s_server utility, the evaluator verified the TOE denied a server certificate with a reference identifier in the CN field that matched the expected identifier configured on the TOE, but with an identifier in the SAN that did not match the configured reference identifier. This was performed for both "dNSName" and "iPAddress" SAN types (RFC 5280 nomenclature). For each session attempt, the evaluator verified the TSF sent a fatal alert message after receiving the server's 'certificate' message. Packet captures confirm these findings for both session attempts with the two different certificates.

Test 3:

Testing for FCS_TLSC_EXT.1.1 Extended: TLS Client Protocol – Test 1 satisfied the objectives for this test. The certificate utilized in FCS_TLSC_EXT.1.1 – Test 1 contained a reference identifier that matched the TOE configured expected identifier, and did not contain a SAN extension. The evaluator verified the session established successfully. Packet captures confirm these findings.

Test 4:

Using OpenSSL's s_server utility, the evaluator verified the TOE successfully established a connection using a server certificate with a reference identifier in the CN field that did not match the expected identifier configured on the TOE, but with an identifier in the SAN extension that did match the configured expected identifier on the TOE. The configured expected identifier of the TOE was itsupport.server.local; the certificate utilized contained a CN of "badcn.server.local" with an SAN of "itsupport.server.local". The evaluator verified that the tool sent the correct certificate with the bad CN but good SAN and verified the session established successfully. Packet captures confirm these findings.

Test 5:

Objective 1a: Using OpenSSL's s_server utility, the evaluator verified the TOE did not support wildcard certificates with wildcards that are NOT in the left-most domain component of the CN identifier. The certificate utilized contained a CN of "itsupport.*.local" with no SAN. The evaluator verified in the packet capture the TSF sent a fatal alert message after receiving the compromised certificate. Packet captures confirm these findings.

Objective 1b: Using OpenSSL's s_server utility, the evaluator verified the TOE did not support wildcard certificates with wildcards that are NOT in the left-most domain component of the SAN extension. The certificate utilized contained a SAN of "itsupport.*.local" with no CN. The evaluator verified in the packet capture the TSF sent a fatal alert message after receiving the compromised certificate. Packet captures confirm these findings.

Objective 2a: Using OpenSSL's s_server utility, the evaluator verified the TOE supported wildcard certificates with a wildcard that was in the left-most domain component of the CN field. The certificate utilized contained a CN of "*.server.local" with no SAN. The evaluator verified in the packet capture that the session established successfully. Packet captures confirm these findings.

Objective 2b: Using OpenSSL's s_server utility, the evaluator verified the TOE supported wildcard certificates with a wildcard that was in the left-most domain component of the CN field. The certificate utilized contained a SAN of "*.server.local" with no CN. The evaluator verified in the packet capture the session established successfully. Packet captures confirm these findings.

Objective 3a: Using OpenSSL's s_server utility, the evaluator verified the connection attempt failed when the s_server utility sent a cert with a wildcard in the left-most domain component for the CN field when the TOE configured expected identifier was configured without a left-most label. The certificate contained a CN of "*.server.local". The TOE configured expected identifier was "server.local" which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings.

Objective 3b: Using OpenSSL's s_server utility, the evaluator verified the connection attempt failed when the s_server utility sent a cert with a wildcard in the left-most domain component for the CN field when the TOE configured expected identifier was configured without a left-most label. The certificate contained a SAN extension of "*.server.local". The TOE configured expected identifier was "server.local" which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings.

Objective 4a: Using OpenSSL's s_server utility, the evaluator verified the connection attempt failed when the server sent a cert with a wildcard in the left-most label for the CN field and the TOE configured expected identifier was configured with two left-most labels. The certificate contained a CN field of "*.server.local". The TOE configured expected identifier was "foo.itsupport.server.local" which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings.

Objective 4b: Using OpenSSL's s_server utility, the evaluator verified that the connection attempt failed when the server sent a cert with a wildcard in the left-most label for the SAN extension and the TOE configured expected identifier was configured with two left-most labels. The certificate contained a SAN extension of "*.server.local". The TOE configured expected identifier was "foo.itsupport.server.local" which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings.

Test 6:

This test was not performed because the TOE did not support URI or Service Name reference identifiers.

Test 7:

This test was not performed because pinned certificates were not supported by the TOE. [ST] Section 7.2.4 states that the TOE does not support the pinning of certificates.

FCS_TLSC_EXT.1.3:

Test 1:

Objective 1: Using OpenSSL's s_server utility, the evaluator verified validation of a certificate without a valid certification path failed. The evaluator sent a server certificate that was signed by 'Intermediate CA – G', which was not installed/trusted on the TOE. The TSF sent a fatal alert message, denying the establishment of the session. A packet capture confirms these findings.

Objective 2: Using OpenSSL's s_server utility, the evaluator verified validation of a certificate with a valid certificate path succeeded once the trust chain was installed on the TOE. The evaluator sent a server certificate that was signed by 'Intermediate CA – G', which was installed/trusted on the TOE. The TSF successfully established the session and encrypted application data was sent. A packet capture confirms these findings.

Objective 3: Using OpenSSL's s_server utility, the evaluator verified that validation did not succeed after deleting one of the certificates in the trust chain. The evaluator deleted the 'Intermediate CA - G' from the TOE's "Trusted Server CAs" certificate store and attempted a connection to the OpenSSL s_server utility which sent the same server certificate signed by the 'Intermediate CA – G'. The TSF sent a fatal alert message, denying the establishment of the session. A packet capture confirms these findings.

FCS_TLSC_EXT.1.4:

Test 1:

Using the custom TLS tool, the evaluator verified the TOE did not negotiate the P-192 curve. A packet capture shows the tool sending the secp192r1 ECDH server parameters in the server's key exchange message and then shows the TSF sending a fatal alert message, denying the establishment of the session.

2.16 FCS_TLSC_EXT.2 Extended: TLS Client Protocol with authentication

FCS_TLSC_EXT.2.1

TSS

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.

Guidance Documentation

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

Tests

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 in an attempt to discern the ciphersuite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).

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

Test 3: The evaluator shall send a server certificate in the TLS connection that the does not match the server-selected ciphersuite (for example, send a 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 configure the server to select the TLS_NULL_WITH_NULL_NULL ciphersuite and verify that the client denies the connection. Test 2 in FCS_TLSS_EXT.1.1 or FCS_TLSS_EXT.2.1 can be used as a substitute for this test.

Test 5: The evaluator 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 (for example 1.3 represented by the two bytes 03 04) and verify that the client rejects the connection.

b) 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.

c) 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.

d) Modify the signature block in the Server's Key Exchange handshake message, and verify that the client rejects the connection after receiving the Server Key Exchange message. 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 5d was modified by "TD0165: NIT Technical Decision for Sending the ServerKeyExchange message when using RSA".*

e) Modify a byte in the Server Finished handshake message, and verify that the client sends a fatal alert upon receipt and does not send any application data.

f) Send a garbled message from the Server after the Server has issued the ChangeCipherSpec message and verify that the client denies the connection.

FCS_TLSC_EXT.2.2

TSS

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 Common Name, DNS Name, URI Name, Service Name, or other application-specific Subject Alternative Names) and whether IP addresses and wildcards are supported. The evaluator shall ensure that this description identifies whether and the manner in which certificate pinning is supported or used by the TOE.

Guidance Documentation

The evaluator shall verify that the AGD guidance includes instructions for setting the reference identifier to be used for the purposes of certificate validation in TLS.

Tests

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

a) Test 1: The evaluator shall present 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. The evaluator shall verify that the connection fails.

b) Test 2: 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.

c) Test 3: The evaluator shall present a server certificate that contains a CN that matches the reference identifier and does not contains the SAN extension. The evaluator shall verify that the connection succeeds.

d) Test 4: 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.

e) Test 5: The evaluator shall perform the following wildcard tests with each supported type of reference identifier:

    1) 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) 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. 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.come) and verify that the connection fails.

f) Test 6: [conditional] If URI or Service name reference identifiers are supported, the evaluator shall configure the DNS name and the service identifier. The evaluator shall present a server certificate containing the correct DNS name and service identifier in the URIName or SRVName fields of the SAN and verify that the connection succeeds. The evaluator shall repeat this test with the wrong service identifier (but correct DNS name) and verify that the connection fails.

g) Test 7: [conditional] If pinned certificates are supported the evaluator shall present a certificate that does not match the pinned certificate and verify that the connection fails.

FCS_TLSC_EXT.2.3

Tests

Test 1: The evaluator shall demonstrate that using a certificate without a valid certification path results in the function failing. Using the administrative guidance, the evaluator shall then load a certificate or certificates needed to validate the certificate to be used in the function, and demonstrate that the function succeeds. . If the certificate is validated and a trusted channel is established, the test passes. The evaluator then shall delete one of the certificates, and show that the certificate is not validated and the trusted channel is not established.

FCS_TLSC_EXT.2.4

TSS

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

Guidance Documentation

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

Tests

Test 1: The evaluator shall configure the server to perform an ECDHE key exchange in the TLS connection using a non-supported curve (for example P-192) and shall verify that the TOE disconnects after receiving the server's Key Exchange handshake message.

FCS_TLSC_EXT.2.5

TSS

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

Guidance Documentation

The evaluator shall verify that the AGD guidance required per FIA_X509_EXT.2.1 includes instructions for configuring the client-side certificates for TLS mutual authentication.

Tests

Test 1: The evaluator shall perform the following modification to the traffic:

a) Configure the server to require mutual authentication and then modify a byte in a CA field in the Server's Certificate Request handshake message. The modified CA field must not be the CA used to sign the client's certificate. The evaluator shall verify the connection fails.

2.16.1 TSS

FCS_TLSC_EXT.2.1

[ST] Section 7.2.4 contains the supported versions of TLS and the list of ciphersuites supported by the TOE, and is consistent with those that are specified in FCS_TLSC_EXT.2.1.

FCS_TLSC_EXT.2.2

[ST] Section 7.2.4 describes the behavior the TLS Client TSF. This section states that certificate pinning is not supported by the TOE. In addition, this section states that the TOE validates the incoming server

certificate using the DNS name or IP address of the server, in addition to the validity checks described in [ST] Section 7.3.4.

[ST] Section 7.3.5 further describes the certificate authentication process of the TSF when a server certificate is presented to the TOE's TLS Client TSF. This section states that the TSF verifies the server certificate in the following manner:

- When the server is specified using a domain name, the TSF verifies that the domain name matches a Subject Alternative Name DNS Name field in the certificate using exact or wildcard matching specified in Section 3.1 of RFC 2818. If the certificate does not contain any Subject Alternative Name fields, the TSF matches the domain name against the Common Name in the certificate.
- When the server is specified using an IP address, the TSF verifies that the IP address exactly matches a Subject Alterative Name IP Address field in the certificate using the rules specified in Section 3.1 of RFC 2818. If the certificate does not contain any Subject Alternative Name fields, the TSF matches the IP address against the Common Name in the certificate.

FCS_TLSC_EXT.2.4

[ST] Section 7.2.4 states that secp256r1 and secp384r1 Elliptic Curves are supported by the TLS Client TSF and are configurable by the Security Administrator. The curves described are consistent with the curves selected for FCS_TLS_EXT.1.4 in [ST] Section 6.1.2.10.

FCS_TLSC_EXT.2.5

[ST] Section 7.3.5, paragraph five, describes the use of client-side certificates for TLS mutual authentication. The TLS client certificate is presented to the remote syslog server for authentication and validation.

2.16.2 Guidance

[AGD] Section 5.12 "Enable NDcPP Mode" contains the guidance on enabling the "NDcPP Mode" which configures the TOE to allow the [ST] selected TLS ciphersuites and TLS Protocol versions on the TOE for both the "inbound" TLS connections (TLS Server configuration) and the "outbound" TLS connections (TLS Client configuration).

[ST] Section 7.3.5 describes the fields in the peer certificate that are compared to the reference identifiers. The fields that are checked against the reference identifiers depend on how the TOE was configured. If the remote audit server destination was configured with a FQDN, then the TSF compares the SAN (Subject Alternative Name) DNS field in the peer certificate to the FQDN configured for the remote audit server. If the peer certificate does not contain a SAN field, then the comparison is made against the FQDN in the 'Common Name' field of the peer certificate. Similarly, if the remote audit server destination was configured with an IP address, the TSF compares the configured IP address to the SAN IP field or to the Common Name field, if no SAN IP field exists in the peer certificate. All of this checking is performed automatically after the remote audit server destination is configured by the Security Admin (i.e., with a FQDN or an IP address).

The TSS does not indicate that the Supported Elliptic Curves Extension must be configured.

TLS mutual authentication is utilized for the connection between the TOE's TLS client and the TLS server of the remote audit server. [AGD] Section 5.11 "Configure Secure Channel to Syslog Server" provides guidance on configuring the mutually authenticated trusted channel between the TOE and the remote audit server. A trusted server CA needs to be imported as per the instructions in [AGD] Section 5.11.1, a "client auth certificate" needs to be imported as per the instructions in [AGD] Section 5.11.1, and a trusted client CA needs to be imported as per the instructions in [AGD] Section 5.8. [AGD] Section 5.13.4 "Configure Syslog Server Parameters" then requires the admin to select the 'client auth certificate' that was imported as per the instructions in [AGD] Section 5.11.1, for the 'Client Certificate' parameter when configuring the remote audit server. This allows the TSF to present the TOE's client certificate to

authenticate to the remote audit server. [AGD] Section 5.10.1 provides the guidance for establishing the fields of an x.509v3 certificate.

2.16.3 Testing

Note: For all FCS_TLSC_EXT.2 testing, the testing tools utilized (OpenSSL s_server & custom TLS tool) were configured to request a client certificate for each test performed. FCS_TLSC_EXT.2 tested Trusted Channel TSF between the TOE and Remote Audit Server.

FCS_TLSC_EXT.2.1:

Test 1:

Using OpenSSL's s_server utility, the evaluator verified the TOE established a TLS session utilizing each of the ciphersuites listed in [ST] Section 6.1.2.11. This included the following ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA
- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

An RSA 2048-bit server certificate was used when negotiating RSA based ciphersuites. An ECDSA P-384 server certificate was utilized when negotiating ECDSA based ciphersuites. The OpenSSL s_server utility shows the connections being established successfully and receiving at least one audit record from the TOE.

Test 2:

Objective 1: This test was satisfied through testing of FCS_TLSC_EXT.2.1 – Test 1. Test 1 is described immediately above. This test verified the TOE accepted a server certificate with 'TLS Server Authentication' purpose in the extendedKeyUsage field and that the connection was successfully established.

Objective 2: The evaluator verified the TOE rejected a server certificate that did not have the 'TLS Server Authentication' purpose in the extendedKeyUsage field by presenting the compromised certificate to the TSF using the OpenSSL s_server tool and configuring the TOE's remote auditing TSF to attempt to connect to s_server instead of the IT Support's syslog audit server. Packet captures confirm that s_server sent the compromised certificate and that the TSF denied the connection attempt by sending a fatal alert message.

Test 3:

Using a custom TLS testing tool, the evaluator verified the TOE rejected a TLS connection attempt with a server that sent a server certificate that does not match the server-selected ciphersuite. Packet captures confirm that the custom TLS testing tool choose an ECDSA based ciphersuite in the server_hello message, but then sent an RSA certificate in the server's 'certificate' message. The TSF sent a TCP RST segment, denying the establishment of the TLS session. No application data was sent.

Test 4:

Using a custom TLS testing tool, the evaluator verified that the TOE did not establish a TLS connection using the TLS_NULL_WITH_NULL_NULL ciphersuite. The test tool sent the null ciphersuite in the server_hello message; the TSF sent a fatal alert message, denying the establishment of the session. A packet capture confirms these findings.

Test 5:

Objective a): Using the custom TLS testing tool, the evaluator verified that the client rejected non-supported TLS versions by attempting to establish a session using TLS version 1.3. The TSF sent a fatal alert message, denying the establishment of the session. Packet captures confirm these findings.

Objective b): Using the custom TLS testing tool, the evaluator verified that the TOE rejected a TLS connection attempt when using a modified nonce in the Server_Hello handshake message. The TSF sent a fatal alert message after receiving the server's 'server key exchange' message, denying the establishment of the session. Packet captures confirm these findings. For greater test coverage, the evaluator attempted the test using a ciphersuite with RSA key exchange and verified that the server rejected (sent a TCP FIN segment) the connection after receiving the client's 'client key exchange' message. Packet captures confirm these findings.

Objective c): Using the custom TLS testing tool, the evaluator verified that the TOE rejected a connection attempt with a server-selected ciphersuite that was not present in the client_hello message sent from the TOE. The TSF sent a fatal alert message after receiving the server's 'server_hello' message which chose a ciphersuite not present in the client_hello message. Packet captures confirm these findings showing the ciphersuite list provided by the client and the server selected ciphersuite. The Server chose the TLS_RSA_WITH_AES_128_GCM_SHA256 ciphersuite which was not on the list provided in the client_hello.

Objective d): Using the custom TLS testing tool, the evaluator verified the TOE rejected a TLS connection attempt with a modified signature block in the Server's 'Key Exchange' message when using an ECDHE ciphersuite. The TSF sent a fatal alert message after receiving the Server Key Exchange message. Packet captures confirm these findings.

Objective e): Using the custom TLS testing tool, the evaluator verified that the TOE sent a fatal alert upon receipt of a modified Server Finished handshake message, and that the TOE did not send application data. The TSF sent an encrypted fatal alert message after receiving the encrypted server_finished message, then immediately sent a TCP FIN segment, denying the establishment of the session. Packet captures confirm these findings.

Objective f): Using the custom TLS testing tool, the evaluator verified that the TSF denied the connection when sending a garbled message from the Server after the Server issued the ChangeCipherSpec message. The TSF sent an encrypted fatal alert message after receiving the garbled message from the test tool, then immediately sent a TCP RST,ACK segment, denying the establishment of the session. Packet captures confirm these findings.

FCS_TLSC_EXT.2.2:

Test 1:

Using OpenSSL's s_server utility, the evaluator verified the TOE denied a server certificate with a reference identifier (CN field) that did not match the reference identifier as configured on the TOE. The evaluator configured the server certificate with a CN of "*badexpectedidentifier.server.local"* (a SAN was not included in the certificate) while the TOE was configured with an expected identifier of *"itsupport.server.local".* The evaluator verified that the TSF sent a TCP FIN,ACK segment, denying the establishment of the session. Packet captures confirm these findings.

Test 2:

Using OpenSSL's s_server utility, the evaluator verified the TOE denied a server certificate with a reference identifier in the CN field that matched the expected identifier configured on the TOE, but with an

identifier in the SAN that did not match the configured reference identifier. This was performed for both “dNSName” and “iPAddress” SAN types (RFC 5280 nomenclature). For each session attempt, the evaluator verified the TSF sent a TCP FIN,ACK segment after receiving the server’s ‘certificate’ message, effectively denying the establishment of the session. Packet captures confirm these findings for both session attempts with the two different certificates.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

Test 3:

Testing for FCS_TLSC_EXT.2.1 Extended: TLS Client Protocol – Test 1 satisfied the objectives for this test. The certificate utilized in FCS_TLSC_EXT.2.1 – Test 1 contained a reference identifier that matched the TOE configured expected identifier, and did not contain a SAN extension. The evaluator verified the session established successfully. Packet captures confirm these findings.

Test 4:

Using OpenSSL’s s_server utility, the evaluator verified the TOE successfully established a connection using a server certificate with a reference identifier in the CN field that did not match the expected identifier configured on the TOE, but with an identifier in the SAN extension that did match the configured expected identifier on the TOE. The configured expected identifier of the TOE was “itsupport.server.local”; the certificate utilized contained a CN of “badcn.server.local” with an SAN of “itsupport.server.local”. The evaluator verified the tool sent the correct certificate with the bad CN but good SAN and verified the session established successfully as s_server received audit record data from the TOE. Packet captures confirm these findings.

Test 5:

Objective 1a: Using OpenSSL’s s_server utility, the evaluator verified the TOE did not support wildcard certificates with wildcards that are NOT in the left-most domain component of the CN field. The certificate utilized contained a CN of “itsupport.*.local” with no SAN. The evaluator verified in the packet capture the TSF sent a TCP FIN,ACK segment after receiving the compromised certificate. Packet captures confirm these findings.

Objective 1b: Using OpenSSL’s s_server utility, the evaluator verified the TOE did not support wildcard certificates with wildcards that are NOT in the left-most domain component of the SAN extension. The certificate utilized contained a SAN of “itsupport.*.local” with no CN. The evaluator verified in the packet capture the TSF sent a TCP FIN,ACK segment after receiving the compromised certificate, effectively denying the establishment of the session. Packet captures confirm these findings.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

Objective 2a: Using OpenSSL’s s_server utility, the evaluator verified the TOE supported wildcard certificates with a wildcard that was in the left-most domain component of the CN field. The certificate utilized contained a CN of “*.server.local” with no SAN. The evaluator verified in the packet capture the session established successfully. Packet captures confirm these findings.

Objective 2b: Using OpenSSL’s s_server utility, the evaluator verified the TOE supported wildcard certificates with a wildcard that was in the left-most domain component of the CN field. The certificate utilized contained a SAN of “*.server.local” with no CN. The evaluator verified in the packet capture the session established successfully. Packet captures confirm these findings.

Objective 3a: Using OpenSSL’s s_server utility, the evaluator verified the connection attempt failed when the s_server utility sent a cert with a wildcard in the left-most domain component for the CN field when the TOE configured expected identifier was configured without a left-most label. The certificate contained a CN of “*.server.local”. The TOE configured expected identifier was “server.local” which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a TCP FIN,ACK segment after receiving the server certificate, effectively denying the establishment of the session. Packet captures confirm these findings.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

Objective 3b: Using OpenSSL's s_server utility, the evaluator verified the connection attempt failed when the s_server utility sent a cert with a wildcard in the left-most domain component for the CN field when the TOE configured expected identifier was configured without a left-most label. The certificate contained a SAN extension of "*.server.local". The TOE configured expected identifier was "server.local" which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a TCP FIN,ACK segment, denying the establishment of the session. Packet captures confirm these findings.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

Objective 4a: Using OpenSSL's s_server utility, the evaluator verified the connection attempt failed when the server sent a cert with a wildcard in the left-most label for the CN field and the TOE configured expected identifier was configured with two left-most labels. The certificate contained a CN field of "*.server.local". The TOE configured expected identifier was "foo.itsupport.server.local" which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a TCP FIN,ACK segment, denying the establishment of the session. Packet captures confirm these findings.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

Objective 4b: Using OpenSSL's s_server utility, the evaluator verified the connection attempt failed when the server sent a cert with a wildcard in the left-most label for the SAN extension and the TOE configured expected identifier was configured with two left-most labels. The certificate contained a SAN extension of "*.server.local". The TOE configured expected identifier was "foo.itsupport.server.local" which the TOE correctly resolved to the IP address of 10.1.2.100 (IT Support VM). The TSF sent a TCP FIN,ACK segment, denying the establishment of the session. Packet captures confirm these findings.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

Test 6:

This test was not performed because the TOE did not support URI or Service Name reference identifiers.

Test 7:

This test was not performed because pinned certificates were not supported by the TOE. [ST] Section 7.2.4 states that the TOE does not support the pinning of certificates.

FCS_TLSC_EXT.2.3:

Test 1:

Objective 1: Using OpenSSL's s_server utility, the evaluator verified validation of a certificate without a valid certification path failed. The evaluator sent a server certificate that was signed by 'Intermediate CA – G', which was not installed/trusted on the TOE. The TSF sent a TCP FIN,ACK segment, denying the establishment of the session. A packet capture confirms these findings.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

Objective 2: Using OpenSSL's s_server utility, the evaluator verified validation of a certificate with a valid certificate path succeeded once the trust chain was installed on the TOE. The evaluator sent a server certificate that was signed by 'Intermediate CA – G', which was installed/trusted on the TOE. The TSF successfully established the session and encrypted application data (audit records) was sent. A packet capture confirms these findings.

Objective 3: Using OpenSSL's s_server utility, the evaluator verified validation did not succeed after deleting one of the certificates in the trust chain. The evaluator deleted the 'Intermediate CA - G' from the TOE's "Trusted Server CAs" certificate store and attempted a connection to the OpenSSL s_server utility

which sent the same server certificate signed by the 'Intermediate CA – G'. The TSF sent a TCP FIN,ACK segment, denying the establishment of the session. A packet capture confirms these findings.

Note: the TSF sent the TCP FIN,ACK segment after the server sent the encrypted server_finished message; however, no application data was sent by the TSF.

FCS_TLSC_EXT.2.4:

Test 1:

Using the custom TLS tool, the evaluator verified the TOE did not negotiate the P-192 curve. A packet capture shows the tool sending the secp192r1 ECDH server parameters in the server's key exchange message and then shows the TSF sending a fatal alert message, denying the establishment of the session.

FCS_TLSC_EXT.2.5:

Test 1:

Using the custom TLS Test tool, the evaluator verified the TOE did not establish a TLS session with a modified CA field in the server's certificate. The tool sent a modified CA Distinguished Name (DN) in the server's certificate_request message (the modified DN was not of the CA that signed the client's certificate). The TSF sent an encrypted alert message after receiving the server's encrypted finished message.

2.17 FCS_TLSS_EXT.1 Extended: TLS Server Protocol

FCS_TLSS_EXT.1.1

TSS

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.

Guidance Documentation

The evaluator shall also 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).

Tests

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 in an attempt 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 use a client to send a key exchange message in the TLS connection that the does not match the server-selected ciphersuite (for example, send an ECDHE key exchange while using the TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite or send a RSA key exchange while using one of the ECDSA ciphersuites.) The evaluator shall verify that the TOE disconnects after the receiving the key exchange message.

**This test was modified by "TD0143: NIT Technical Decision for Failure testing for TLS session establishment in NDcPP and FWcPP".*

Test 4: The evaluator shall perform the following modifications to the traffic:

**This test was modified by "TD0151: NIT Technical Decision for FCS_TLSS_EXT Testing - Issue 1 in NDcPP v1.0".*

a) **This Test Was Removed by TD0151**

b) **This Test Was Removed by TD0151**

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

d) After generating a fatal alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, send a Client Hello with the session identifier from the previous test, and verify that the server denies the connection.

e) Send a garbled message from the client after the client has issued the ChangeCipherSpec message and verify that the Server denies the connection.

FCS_TLSS_EXT.1.2

TSS

The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions.

Guidance Documentation

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

Tests

**This test was revised by "TD0156: NIT Technical Decision for SSL/TLS Version Testing in the NDcPP v1.0 and FW cPP v1.0".*

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

FCS_TLSS_EXT.1.3

TSS

The evaluator shall verify that the TSS describes the key agreement parameters of the server key exchange message.

Guidance Documentation

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

Tests

**This test was revised by "TD0155: NIT Technical Decision for TLSS tests using ECDHE in the NDcPP v1.0".*

The evaluator shall attempt establishing connection using each claimed key establishment protocol (RSA, DH, ECDHE) with each claimed parameter (RSA key size, Diffie-Hellman parameters, supported curves) as selected in FCS_TLSS_EXT.1.3 (or FCS_TLSS_EXT.2.3). For example, determining that the RSA key size matches the claimed size is sufficient to satisfy this test. The evaluator shall ensure that each supported parameter combination is tested.

Note that this testing can be accomplished in conjunction with the other testing activities.

2.17.1 TSS

FCS_TLSS_EXT.1.1

[ST] Section 7.2.5, paragraph one and two, list the TOE supported TLS Server ciphersuites. The list is identical to that of FCS_TLSS_EXT.1.1 in [ST] Section 6.1.2.12.

FCS_TLSS_EXT.1.2

[ST] Section 7.2.5, paragraph one, states that if the TOE receives a Client_Hello message requesting TLSv1.0 or earlier, the TSF sends a fatal handshake_failure message and closes the connection.

FCS_TLSS_EXT.1.3

[ST] Section 7.2.5, paragraph four states that when the TSF selects an EDHE ciphersuite, it sends the client secp256r1 or secp384r1 key agreement parameters. When an RSA-based ciphersuite is negotiated, 2048, or 3072 bit RSA key parameters are utilized.

2.17.2 Guidance

[AGD] Section 5.12 “Enable NDcPP Mode” provides guidance on configuring the TOE to use the TLS version and ciphersuites selected and mandated by FCS_TLSC_EXT.2.1 in [ST] Section 6.1.2.11.

[AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance on configuring TLS to accept only TLS v1.1 and v1.2.

[AGD] Section 5.12 contains the guidance for configuring the TOE to use all of the allowed TLS ciphersuites for the TLS server, which contain the key establishment algorithms and key/curve sizes. The key establishment parameters that are generated by the TSF depend on the configured TLS server certificate for the TOE. If the server certificate is configured with an RSA key, the TSF supports all the [ST] allowed ciphersuites with RSA authorization. When a ciphersuite with RSA key exchange is utilized, the TSF generates 2048 or 3072 bit parameters. If a ciphersuite with ECHDE key agreement is utilized, the TSF generates secp256r1 or secp384r1 curve parameters (depending on which of these two curves are supported by the TLS client). If the server certificate was configured with ECDSA key, the TSF supports all the [ST] allowed ciphersuites with ECDSA authorization.

[AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance for configuring TLS to use the key-establishment schemes and associated key sizes and curves selected in [ST] Section 6.1.2.12 for FCS_TLSS_EXT.1.3. This section instructs the admin to enable ‘NDcPP Mode’ and additional configuration boxes which enables the TOE to use the TLS ciphersuites that are selected in the [ST]. These TLS server ciphersuites allow the following key-establishment schemes, which generate the following key sizes and curves:

- RSA
  - 2048 and 3072-bit key sizes
- ECDHE curves
  - Secp256r1 and secp384r1

2.17.3 Testing

Note: FCS_TLSS_EXT.1 tested the Trusted Channel TSF between the TOE and a Remote Administrative User.

FCS_TLSS_EXT.1.1:

Test 1:

Using the custom TLS tool, the evaluator verified the TOE successfully established a TLS session with each of the ciphersuites listed in the [ST] Section 6.1.2.12. Testing included the following ciphersuites:

- TLS_RSA_WITH_AES_128_CBC_SHA
- TLS_RSA_WITH_AES_256_CBC_SHA

- TLS_DHE_RSA_WITH_AES_128_CBC_SHA
- TLS_DHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
- TLS_RSA_WITH_AES_128_CBC_SHA256
- TLS_RSA_WITH_AES_256_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

When attempting to establish RSA based ciphersuites, an RSA 2048-bit server certificate was configured on the TOE. An ECDSA P-384 server certificate was utilized when negotiating ECDSA based ciphersuites.

Test 2:

Objective 1: Using the OpenSSL s_client utility, the evaluator attempted to negotiate a TLS session using a list of ciphersuites that were not configured on the TOE and verified in the output of the s_client utility that the TLS server denied the connection attempt. The following ciphersuites were sent in the client_hello:

- CAMELLIA128-SHA
- PSK-AES128-CBC-SHA
- ECDHE-RSA-RC4-SHA
- ECDH-RSA-RC4-SHA
- RC4-SHA
- RC4-MD5
- PSK-RC4-SHA
- EDH-RSA-DES-CBC-SHA
- EDH-DSS-DES-CBC-SHA
- DES-CBC-SHA

Objective 2: Using the custom TLS tool, the evaluator attempted to negotiate a TLS session using the TLS_NULL_WITH_NULL_NULL ciphersuite and verified in the output of the s_client utility and in a packet capture, that the TLS server denied the connection attempt. The TSF sent a fatal alert after receiving the client_hello message, denying the establishment of the session.

Test 3:

Using the custom TLS tool, the evaluator verified that the TOE terminated the connection when a Client_Key_Exchange message was sent that did not match the server-selected ciphersuite. The TLS tool shows the server selecting the AES128-SHA256 ciphersuite. The tool then sends ECDH parameters in the client_key_exchange message. The TSF sent a fatal alert, denying the connection attempt.

Test 4:

Test c): Utilizing the custom TLS tool, the evaluator verified the server rejected the connection and did not send any application data when a byte in the Client's finished handshake message was modified. The packet capture shows the TSF sending a fatal alert message after receiving the client's encrypted finished message.

Test d): Utilizing the custom TLS tool, the evaluator verified the server denied the connection after attempting to resume a TLS session using the session ID gained from a previous failed handshake terminated by a fatal alert from the server. The test tool attempts to establish an initial session with the

server but forced a fatal alert and attempts to establish a session with the server using the session ID gleaned from the previous failed attempt. The packet capture shows the TSF sending a fatal alert message during the first connection attempt then the next attempt at a connection is terminated by the test tool immediately after the server sent an encrypted client_finished message. The evaluator observed that the server_hello's all contained a session ID length of zero (0). This shows that the server did not support session resumption. To verify the TOE did not support session caching, the evaluator used the OpenSSL's s_client –reconnect feature which tests for TLS Server session caching by attempting 5 connections to the server after an initial connection to grab the server's first session ID, attempting to use the session ID from the first sever_hello message. Each TLS handshake in the output of the command required full TLS handshake which included the client key_exchange message. This means no session was resumed since a successfully resumed session would skip any key exchange messages and utilize the same key parameters negotiated in the first session. Each server_hello contained a session ID length of zero (0).

Test e): Utilizing the custom TLS tool, the evaluator verified the server denied the connection after the client sent a garbled messaged after the client already sent the change_cipher_spec message. The packet capture shows the TSF sending a fatal alert message after receiving the client's garbled message.

FCS_TLSS_EXT.1.2:

Test 1:

Using the custom TLS tool, the evaluator verified the TOE's TLS Server did not negotiate TLSv1.0 or below (i.e., SSL 3.0, SSL2.0, SSL1.0). The test tool attempted to negotiate a session for each protocol version of TLSv1.0 and below. Each session attempt failed. The evaluator then used OpenSSL s_client and verified that the TSF negotiated TLSv1.1, and TLSv1.2 in separate session/connection attempts. A packet capture supports these findings. In addition, the output of OpenSSL s_client supports these findings.

FCS_TLSS_EXT.1.3:

Test 1:

Objective 1): Using the custom TLS tool, the evaluator verified the TOE's TLS Server negotiates the appropriate elliptic curves. The appropriate curves negotiated are the following:

- P-256
- P-384

Objective 2): Using the custom TLS tool, the evaluator verified the TOE's TLS Server negotiated DHE 256-bit key exchange.

2.18 FIA_PMG_EXT.1 Password Management

Guidance Documentation

The evaluator shall examine the guidance documentation to determine that it provides guidance to security administrators on the composition of strong passwords, and that it provides instructions on setting the minimum password length.

Tests

The evaluator shall perform the following tests.

a) Test 1: The evaluator shall compose passwords that either meet the requirements, or fail to 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, rule characteristics, and a minimum length listed in the requirement are supported, and justify the subset of those characters chosen for testing.

2.18.1 TSS

None

2.18.2 Guidance

[AGD] Section 5.2 provides guidance on configuring a minimum password-length requirement.

[AGD] Section 4.2 contains guidance for the creation of strong passwords.

2.18.3 Testing

Test 1:

Objective 1): The evaluator verified the TOE supported passwords composed of the allowed alphanumeric and special characters as described in [ST] Section 6.1.3.1 for both the remote and local console authentication mechanisms. The evaluator configured an administrative password with the following characters:

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&()*

The evaluator verified admin could login to the TOE's local console and administrative web GUI using this configured password.

Objective 2): The evaluator verified the TOE did not support passwords composed of disallowed alphanumeric and special characters as described in [ST] Section 6.1.3.1 for both the remote and local console authentication mechanisms. The evaluator attempted to configure an administrative password with the following characters:

ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789!@#$%^&()?+=\/*

The evaluator verified the attempt was unsuccessful.

Objective 3): The evaluator verified the TOE supported and enforced the minimum password requirements as set (to a minimum of 15 characters) for both remote and local console authentication. The evaluator configured an administrative password with less than 15 characters; however, verified the admin could not authenticate to the local console or to the remote web GUI administrative interface using the 14 character password.

2.19 FIA_UIA_EXT.1 User Identification and Authentication

TSS

The evaluator shall examine the TSS to determine that it describes the logon process for each logon method (local, remote (HTTPS, SSH, etc.)) supported for the product. This description shall contain information pertaining to the credentials allowed/used, any protocol transactions that take place, and what constitutes a "successful logon".

Guidance Documentation

The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps (e.g., establishing credential material such as pre- shared keys, tunnels, certificates, etc.) to logging in are described. For each supported the login method, the evaluator shall ensure the guidance documentation provides clear instructions for successfully logging on. If configuration is necessary to ensure the services provided before login are limited, the evaluator shall determine that the guidance documentation provides sufficient instruction on limiting the allowed services.

Tests

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

a) Test 1: The evaluator shall use the guidance documentation to configure the appropriate credential supported for the login method.

For that credential/login method, the evaluator shall show that providing correct I&A information results in the ability to access the system, while providing incorrect information results in denial of access.

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

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

2.19.1 TSS

[ST] Section 7.3.2 describes the logon process for each logon method supported for the TOE. The TOE utilizes HTTPs for a secure remote administration web GUI. Admins can authenticate via this management interface using a username/password combination. Successful logon occurs when there is a match in the credentials.

The TOE also provides a local serial CLI console. Admins can authenticate via username/password combination.

2.19.2 Guidance

[ST] Section 7.3.2 states that the TOE provides a local serial console and a remote web GUI management interface for local and remote administrative management of the TOE, respectively. The local serial console and remote web GUI management interfaces both require a username/password combination credential to log in.

[AGD] Section 4.2 provides the instructions for establishing the administrator username and password credentials for both the local console and remote web GUI management interface. These credentials are created at initial system installation and setup time (via the local serial console).

2.19.3 Testing

Test 1:

Objective 1): The evaluator verified administrative authentication to the TOE via the local console was:

- Successful with correct username/password credentials
- Denied with a correct username but an incorrect password
- Denied with a non-configured username and a random password

Objective 2): The evaluator verified administrative authentication to the TOE via the remote web administrative GUI was:

- Successful with correct username/password credentials
- Denied with a correct username but an incorrect password
- Denied with a non-configured username and a random password

Test 2:

Using the nmap networking scanning utility, the evaluator verified the services that are available through the configured “internal” Ethernet port interface are restricted to those listed in [ST] Section 6.1.3.2. Note, the services listed in [ST] Section 6.1.3.2 and even services not listed in that section, but utilized for the Trusted Channel and Trusted Path, are acceptable services available remotely. These include the TLS Web Server hosted at port 443 (Trusted Path) and TCP ports 11122 and 11123 for unevaluated security functionality communication to firewalls.

TCP port 80 was found to be open; however, using a custom python script, the evaluator verified port 80 was a HTTP 301 permanent redirect to https: or port 443 (Trusted Path). The evaluator verified there were no UDP services open.

Test 3:

The evaluator verified that only the following services were available to the local console prior to authentication:

- Display the warning banner in accordance with FTA_TAB.1

This is consistent with the services available prior to authentication to the local console as provided in [ST] Section 6.1.3.2.

2.20 FIA_UAU_EXT.2 Password-based Authentication Mechanism

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

2.21 FIA_UAU.7 Protected Authentication Feedback

Tests

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

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

2.21.1 TSS

None

2.21.2 Guidance

None

2.21.3 Testing

Test 1:

The evaluator verified that no feedback was provided when entering authentication information on the local console. A screen capture verifies no characters were echoed back to the local console when attempting authentication to the local console.

2.22 FIA_X509_EXT.1 X.509 Certificate Validation

TSS

The evaluator shall ensure the TSS describes where the check of validity of the certificates takes place. The evaluator ensures the TSS also provides a description of the certificate path validation algorithm.

The evaluator shall ensure the TSS describes when the check of validity of the certificates takes place. 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 sufficient to verify the status of a X.509 certificate only when it's loaded onto the device.

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).

Tests

NOTE: This test was updated by multiple TDs:
**The paragraph below was added by 'TD0093: NIT Technical Decision for FIA_X509_EXT.1.1 Requirement in NDcPP':*
"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 device. 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 paragraph below was added by "TD0117: NIT Technical Decision for FIA_X509_EXT.1.1 Requirement in NDcPP":*

"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's loaded onto the device.

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:

**Tests 1a and 1b were updated by "TD0187: NIT Technical Decision for Clarifying FIA_X509_EXT.1 test 1":*

a) *Test 1a: The evaluator shall load a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the certificate to be used in the function, and shall use this chain to demonstrate that the function succeeds.*
Test 1b: The evaluator shall then delete one of the certificates in the chain (i.e. the root CA certificate or other intermediate certificate, but not the end-entity certificate), and show that the function fails.

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

c) Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the TOE certificate and revocation of the TOE intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails.

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.)

** FIA_X509_EXT.1.2 was modified by'TD0093: NIT Technical Decision for FIA_X509_EXT.1.1 Requirement in NDcPP'.*

The evaluator shall perform the following tests for FIA_X509_EXT.1.2. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules.

The goal of the following tests it to verify that the TOE accepts only certificates that have been marked as CA certificates by using basicConstraints with the CA flag set to True (and implicitly 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 four certificates: a self-signed root CA certificate, two intermediate CA certificates, 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).

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).

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).

2.22.1 TSS

[ST] Section 7.3.4 states that the validity of the certificates is checked by the following:

- Checks that the cert is not listed in the applicable CRLs
- The certificate chain is valid
    - Checking validity of the 'valid from' and 'valid to' dates
    - Checking if each certificate is signed by a CA
    - Each CA has the basicConstraints 'CA=TRUE' flag is set

Each certificate is signed by a certificate in the cert chain or by a trusted root CA that has been installed in the TSF

2.22.2 Guidance

[ST] Section 6.1.3.7 selected the Common Name, Organization, Organizational Unit and Country fields which are available for configuration for a X.509 Certificate Request Message.

[AGD] Section 5.10.1 provides instructions on generating a Certificate Request Message for requesting a signed certificate from a CA, and establishing the following fields in the CSR:

- Common Name
- Organization Name
- Locality
- State
- Country
- Key Type
- Key Length

2.22.3 Testing

FIA_X509_EXT.1

Test 1a/1b:

These tests were satisfied by the testing performed for FCS_TLSC_EXT.1.3 Objectives 1, 2 and 3, and by testing for FCS_TLSC_EXT.2.3 Objectives 1, 2 and 3.

Test 2:

The evaluator verified an expired certificate did not allow authentication to the TOE. The evaluator created an expired certificate to be presented by the IT Support VM via OpenSSL's s_server utility on port 6514 (Pulse One Trusted Channel), then configured the TOE to attempt to connect to s_server. The evaluator verified the function failed. S_server presented the expired certificate to the TSF; the TSF sent a TCP FIN,ACK segment, denying the establishment of the session. A packet capture confirms these findings.

Test 3:

Objective 1): The evaluator verified the TSF properly handled a peer certificate that was revoked (using a CRL only). Using OpenSSL's s_server, the evaluator attempted to establish a TLS session with the remote auditing Trusted Channel TSF of the TOE. S_server presented the server certificate that was revoked by the signing intermediate CA (Intermediate CA - C). 'Intermediate CA – C's' CRL was hosted and reachable by the TOE. The TOE downloaded the appropriate CRLs when validating the trust chain of the presented certificate. In doing so, the TSF correctly validated the presented server certificate, indicating that the certificate was revoked in the CRL. The TSF sent a fatal alert message, denying the establishment of the session.

Objective 2): The evaluator verified the TSF properly handled an intermediate CA that was revoked (using a CRL only). Using OpenSSL's s_server, the evaluator attempted to establish a TLS session with the remote auditing Trusted Channel TSF of the TOE. S_server presented a valid server certificate that was signed by a revoked intermediate CA (Intermediate CA - C). 'Intermediate CA – C' was revoked by the 'root CA'. The 'root CA's CRL was hosted and reachable by the TOE. The TOE downloaded the appropriate CRLs when validating the trust chain of the presented certificate. In doing so, the TSF correctly validated the Intermediate CA - C, indicating the certificate was revoked in the 'root CA' CRL. The TSF sent a fatal alert message, denying the establishment of the session.

Note: The TOE implemented CRL caching. Since the lowest time period for CRL updates was 1 hour, the evaluator forced the TOE to download the updated CRLs by deleting all CAs (including intermediate CAs and the root CA) in the trust chain of the peer certificate, and then reinstalled them.

Objective 3): The evaluator verified the TSF successfully installed a valid 'Device Certificate' certificate when the trust chain was installed. The evaluator verified installation of a valid root CA, intermediate CA, and end entity certificate was successful.

Objective 4): The evaluator verified the TSF rejected the attempt to install a revoked 'Device Certificate'. The evaluator created a new server certificate that was signed by an intermediate CA, then revoked the newly created certificate using the CRL of the intermediate CA that signed the new certificate. The new CRL was generated, hosted, and accessible by the TOE. The evaluator attempted to install the revoked certificate onto the TOE and verified the attempt failed.

Objective 5): The evaluator verified the TSF rejected the attempt to install a revoked intermediate CA. The evaluator created a new intermediate CA signed by the root CA and revoked the new intermediate CA using the root CA CRL. The new CRL was generated, hosted, and accessible by the TOE. The evaluator attempted to install the revoked intermediate CA onto the TOE and verified that the attempt failed.

Test 4:

The evaluator verified the TSF failed to validate a CRL that was signed by a CA that did not have the 'CRL Sign' keyUsage bit set. The evaluator created an intermediate CA signed by the root CA. The new

intermediate CA had the 'Certificate Sign' keyUsage bit set but not the 'CRL Sign' keyUsage bit set. Generated a CRL from the new intermediate CA and hosted the CRL. The TOE could reach the CRL. The evaluator then created a new end-entity certificate signed by the new intermediate CA (the one without the 'CRL Sign' keyUsage bit set) and attempted to install the new end-entity certificate onto the TOE. The evaluator verified the attempt failed due to failed validation of the CRL. The audit records show that the certificate was rejected because it could not be validated.

Test 5:

The evaluator verified the TOE did not validate a certificate that had a byte within the first 8 bytes of the certificate, modified. The evaluator created a new root CA certificate, exported the certificate in PEM format, changed the first two characters of the certificate (after the headers) and saved the modified cert. The evaluator then attempted to install the root CA into the TOE and verified that the attempt failed.

Test 6:

The evaluator verified the TSF failed to validate a certificate with a modified byte in the last byte of the certificate. The evaluator exported a previously created certificate used during FCS_TLSC_EXT.1 testing. The evaluator converted the certificate from PEM to HEX and modified the last byte (two characters), converted the modified HEX back into PEM format, replaced the header and footer and saved the certificate as "Modified_Last_Byte.crt". The evaluator used OpenSSL's s_server utility to present the modified certificate to the remote audit Trusted Channel TSF. The trust chain was installed on the TOE for the modified certificate. The evaluator verified the TSF denied the session establishment attempt. The TSF sent a fatal alert message. A packet capture supports the findings.

Test 7:

The evaluator verified the TSF failed to validate a certificate that had a modified public key. The evaluator exported a previously created certificate used during FCS_TLSC_EXT.1 testing. The evaluator converted the certificate from PEM to HEX and modified 3-bytes in the public-key portion of the certificate from '88A6DE' to 'C0FFEE'; converted the modified HEX back into PEM format, replaced the header and footer and saved the certificate as "Modified_Public_Key.crt". The evaluator used the custom TLS tool to present the modified certificate to the remote audit Trusted Channel TSF. The trust chain was installed on the TOE for the modified certificate. The evaluator verified the TSF denied the session establishment attempt. The TSF sent a fatal alert message. A packet capture supports the findings.

FIA_X509_EXT.1.2

Test 1:

The evaluator verified the TSF did not validate the certificate path of a CA certificate that did not contain the CA=True basicConstraints field. The evaluator created a new intermediate CA signed by an intermediate CA in the installed trust chain in the TOE. The new intermediate CA did not contain the basicConstraints field. The evaluator then attempted to import the new intermediate CA into the TOE and verified the attempt failed to validate the intermediate CA.

Test 2:

The evaluator verified that the TSF fails to validate the certificate path when the CA that signed the TOE's certificate had the CA flag in the basicConstraints extension set to 'FALSE'. The evaluator created a new intermediate CA signed by 'Intermediate CA – A'. The evaluator attempted to import the new intermediate CA into the TOE and verified the TSF rejected the intermediate CA because the CA flag in the basicConstraints extension set to 'FALSE'. Screen captures show the TSF rejecting the certificate and giving the reason.

Test 3:

The evaluator verified the TSF accepted a certificate with a valid certification path using intermediate CAs. The evaluator created a new intermediate CA signed by 'Intermediate CA – A'. The new intermediate CA contained the CA flag in the basicConstraints extension set to 'TRUE'. The evaluator verified this new intermediate CA was installed successfully on the TOE. In addition, the evaluator

generated a new end entity certificate from the new intermediate CA and verified this certificate was successfully installed into the TOE.

2.23 FIA_X509_EXT.2 X.509 Certificate Authentication

TSS

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

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

Tests

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

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

2.23.1 TSS

[ST] Section 7.3.4 describes how/when the TOE chooses which certificates to use.

[ST] Section 7.3.5, last paragraph, states that if the TOE cannot contact the CLR server or the server does not respond, the TSF logs the failure and considers the certificate valid.

[ST] Section 7.3.4 describes the X.509 certificate validation process. The TOE checks the certificate identifiers are valid for the syslog server, then performs the validity steps as described in [ST] Section 7.3.4. If any of the validity checks fail, the connection is not established.

The TOE establishes a Trusted Channel between itself and a remote syslog server and the Pulse One remote management server.

2.23.2 Guidance

None

2.23.3 Testing

Test 1:

Objective 1: The evaluator verified the TSF accepted a valid certificate when the CRL is downloadable for both the TLSC_EXT.1 and TLSC_EXT.2 Trusted Channels. Using the 'wget' program on the IT Support VM, the evaluator verified all relevant CRLs were accessible in the environment. Utilizing the certificate chain created during FCS_TLSC_EXT.1 testing, the evaluator verified a Trusted Channel session between the remote audit server TSF and OpenSSL's s_server utility was successfully established. Audit data was sent to s_server and a packet capture supports these findings. The evaluator repeated this for the Pulse One remote management server Trusted Channel TSF and verified that the session establishment was successful.

Objective 2: The evaluator verified the TSF accepted a valid certificate when the CRL was not accessible/downloadable in the testing environment for both the TLSC_EXT.1 and TLSC_EXT.2 Trusted Channels. Using the 'wget' program on the IT Support VM, the evaluator verified that all relevant CRLs were inaccessible in the testing environment. Utilizing the certificate chain created during FCS_TLSC_EXT.1 testing, the evaluator verified that a Trusted Channel session between the remote audit server TSF and OpenSSL's s_server utility was successfully established. Audit data was sent to s_server and a packet capture supports these findings. The evaluator repeated this for the Pulse One remote management server Trusted Channel TSF and verified that the session establishment was successful.

2.24 FIA_X509_EXT.3 Extended: X509 Certificate Requests

TSS

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

Guidance Documentation

The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request Message. 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 certificate request message.

Tests

The evaluator shall perform the following tests:

a) Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a certificate request message. The evaluator shall capture the generated message and ensure that it conforms to the format specified. The evaluator shall confirm that the certificate 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 certificate response message without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate response message, and demonstrate that the function succeeds. The evaluator shall then delete one of the certificates, and show that the function fails.

2.24.1 TSS

[ST] Section 6.1.3.7 did not select "device-specific information", therefore, this Work Unit is not applicable.

2.24.2 Guidance

2.24.3 Testing

Test 1:

The evaluator verified the TSF generated CSR conforms to the format specified in [ST] Section 6.1.3.7 for FIA_X509_EXT.3.1. The evaluator checked the details of the Certificate Signing Request message using OpenSSL's 'req' utility and verified that it contained the Public-key, Common Name, Organization, Locality, State, Country, Key Type and Key Length information.

Test 2:

Objective 1: The evaluator verified the TOE did not import a signed CSR without the CA that signed the CSR configured as trusted on the TOE. The evaluator created a new root CA and did not install this new

root CA onto the TOE. The evaluator signed the CSR using this untrusted root CA and attempted to install the signed CSR on the TOE. The evaluator verified that the attempt failed.

Objective 2: The evaluator verified the TOE successfully imported the signed CSR when the CA that signed the CSR was configured as trusted on the TOE. The evaluator utilized the new root CA to sign the CSR. The evaluator installed the new root CA into the TOE as trusted and attempted to install the signed CSR on the TOE. The evaluator verified the attempt was successful.

Objective 3: This objective was unable to be performed on the TOE since once the signed CSR is successfully imported for the appropriate CSR, the CSR is no longer available on the TOE.

2.25 FMT_MOF.1(1)/Trusted Update

Tests

The evaluator shall try to perform the update using a legitimate update image without prior authentication as security administrator (either by authentication as a user with no administrator privileges or without user authentication at all – depending on the configuration of the TOE). This test should fail.

The evaluator shall try to perform the update with prior authentication as security administrator using a legitimate update image. This test should pass. This test case should be covered by the tests for FPT_TUD_EXT.1 already.

2.25.1 TSS

None

2.25.2 Guidance

None

2.25.3 Testing

Test 1:

Objective 1: The evaluator verified an unauthenticated user and an authenticated non-administrative user cannot update the TOE. Testing for FMT_MOF.1(1)/AdminAct, FMT_MOF.1(1)/Audit, and FMT_MTD.1/AdminAct Management of TSF Data showed there were no interfaces for authenticated non-administrative users to attempt to update the TOE's software/firmware, satisfying testing for Objective 1. These tests found no interface for an unauthenticated user, nor for an authenticated non-administrative user to perform actions reserved for the security administrative user.

Objective 2: The evaluator verified an authenticated administrative user could update the TOE. Objective 2 is satisfied by the tests performed for FPT_TUD_EXT.1 Trusted Update – TEST 1.

2.26 FMT_MOF.1(1)/Audit

Tests

The evaluator shall try to modify all parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity 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). This test should fail.

The evaluator shall try to modify all parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as security administrator. The effects of the modifications should be confirmed.

The evaluator does not necessarily have to test all possible values of all parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity but at least one allowed value per configurable parameter.

2.26.1 TSS

None

2.26.2 Guidance

None

2.26.3 Testing

Test 1:

Objective 1: The evaluator verified an unauthenticated user could not modify the parameters for configuration of the transmission protocol for the transmission of audit data to an external IT entity. This test is satisfied by the testing performed for FMT_MOF.1(1)/AdminAct Objective 1.

Objective 2: The evaluator verified an authenticated, non-administrative user could not modify the parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity. This test is satisfied by the testing performed for FMT_MOF.1(1)/AdminAct Objective 2.

Objective 3: The evaluator verified an authenticated administrative user could modify the parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity. The evaluator logged into the TOE with administrative credentials and modified multiple parameters for the transmission of the audit data including the following:

- 'Type' parameter from TLS to TCP and to UDP
- 'Client Certificate' parameter from 'psa3000.server.local' to 'Select Client Cert'

2.27 FMT_MOF.1(2)/Audit

Tests

The evaluator shall try to modify all parameters for configuration of the handling of audit data 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). This test should fail. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

The evaluator shall try to modify all parameters for configuration of the handling of audit data with prior authentication as security administrator. The effects of the modifications should be confirmed. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2.

The evaluator does not necessarily have to test all possible values of all parameters for configuration of the handling of audit data but at least one allowed value per configurable parameter.

2.27.1 TSS

None

2.27.2 Guidance

None

2.27.3 Testing

Test 1:

Objective 1: The evaluator verified non-administrative users and users that have not authenticated to the TOE were not able to modify parameters for configuration of the handling of audit data. Unauthenticated users have no interface to configure/modify the parameters for the configuration of the handling of audit data as revealed in testing for FAU_STG.1. For authenticated non-administrative users, the evaluator attempted to login to the administrative web GUI with user level credentials. The evaluator verified non-administrative users could not authenticate to the administrative web GUI. In addition, the evaluator logged into the TOE's 'user' web portal and attempted to access the following URL's that were discovered during authorized administrative access to the local audit records and configuration:

- https://psa3000.server.local/dana/log/logviewer.cgi?type=events
- https://psa3000.server.local/dana/log/log.cgi?type=events

The evaluator verified access was denied.

Objective 2: The evaluator verified administrative users authenticated to the TOE were able to modify all parameters for configuration of the handling of audit data. The evaluator logged into the TOE via the administrative web GUI and navigated to the configuration page for the 'events' logging. The evaluator modified the 'Max Log Size' parameter and verified the TSF accepted the configuration. In addition, the evaluator verified the administrative user was able to enable/disable specific event types within the auditing configuration.

2.28 FMT_MOF.1(1)/AdminAct

Tests

The evaluator shall try to perform at least one of the related actions 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). These attempts should fail.

The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. These attempts should succeed.

2.28.1 TSS

None

2.28.2 Guidance

None

2.28.3 Testing

Test 1:

Objective 1: The evaluator verified an unauthenticated user could not modify the remote audit logging TSF configuration. This objective was satisfied by the testing performed for FAU_STG.1 Test 1. This testing found there were no interfaces for a non-administrative user, nor for an unauthenticated user to modify the behavior of the TOE Security Functions.

Objective 2: The evaluator verified an authenticated, non-administrative user could not modify the remote audit logging TSF configuration. This objective was satisfied by the testing performed for FAU_STG.1 Test 1. This testing found that there were no interfaces for a non-administrative user, nor for an unauthenticated user to modify the behavior of the TOE Security Functions.

Objective 2: The evaluator verified an authenticated administrative user could modify the remote audit logging TSF configuration. The evaluator logged into the TOE using administrative credentials and changed the following parameters in the configuration of the remote auditing TSF:

- Server Name/IP
- Type
- Client Certificate

The evaluator confirmed the configuration changes were successful.

2.29 FMT_MTD.1 Management of TSF Data

TSS

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

Guidance Documentation

The evaluator shall review the guidance documentation to determine that each of the TSF-data-manipulating functions implemented in response to the requirements of the c PP is identified, and that configuration information is provided to ensure that only administrators have access to the functions.

2.29.1 TSS

[ST] Section 7.6 describes the TOE access warning banner that is presented before user authentication to the TOE.

[ST] Section 7.4 contains the description of the management of the TSF data. This section states that the TSF restricts access management functions to users that have been authenticated to the TOE.

2.29.2 Guidance

The TSF-data-manipulating functions are defined as the commands and actions an administrator can issue/take to configure the TOE into the CC evaluated configuration. These commands and actions, while not explicitly listed in [AGD] as “TSF-data-manipulating functions”, are identified in [AGD] as the guidance/instructions provided to configure the TSF.

[ST] Section 7.4 states that the TSF restricts access to the management functions (TSF-data-manipulating functions) to users that have been identified, authenticated and authorized with the Security Administrator role.

[AGD] Section 4.2 provides the initial configuration guidance for configuring the administrative credentials.

2.29.3 Testing

None

2.30 FMT_MTD.1/AdminAct Management of TSF Data

Tests

The evaluator shall try to perform at least one of the related actions 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). This test should fail.

The evaluator shall try to perform at least one of the related actions with prior authentication as security administrator. This test should pass.

2.30.1 TSS

None

2.30.2 Guidance

None

2.30.3 Testing

Test 1:

Objective 1: The evaluator verified an unauthenticated user could not modify/delete/generate/import cryptographic keys. This objective was satisfied by the testing performed for FAU_STG.1 Test 1, which found there were no interfaces for a non-administrative user, nor for an unauthenticated user to modify the behavior of the TOE Security Functions.

Objective 2: The evaluator verified an authenticated, non-administrative user could not modify/delete/generate/import cryptographic keys. This objective was satisfied by the testing performed for FAU_STG.1 Test 1, which found that there were no interfaces for a non-administrative user, nor for an unauthenticated user to modify the behavior of the TOE Security Functions.

Objective 3: The evaluator verified an authenticated administrative user could generate cryptographic keys. This objective was satisfied by testing for FIA_X509_EXT.3 Extended: X.509 Certificate Requests – TEST 1.

2.31 FMT_SMF.1 Specification of Management Functions

The security management functions for FMT_SMF.1 are distributed throughout the cPP and are included as part of the requirements in FTA_TAB.1, FTA_SSL.3, FTA_SSL.4, FMT_MOF.1(1)/TrustedUpdate, FMT_MOF.1(2)/TrustedUpdate (if included in the ST), FIA_X509_EXT.2.2 & FPT_TUD_EXT.2.2 (if included in the ST and if they include an administrator-configurable action), FMT_MOF.1(1)/Audit, FMT_MOF.1(2)/Audit, FMT_MOF.1.1(1)/AdminAct, FMT_MOF.1.1(2)/AdminAct and FMT_MOF.1/LocSpace (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.

2.32 FMT_SMR.2 Restrictions on security roles

Guidance Documentation

The evaluator shall review the guidance documentation to ensure that it contains instructions for administering the TOE both locally and remotely, including any configuration that needs to be performed on the client for remote administration.

Tests

In the course of performing the testing activities for the evaluation, the evaluator shall use all supported interfaces, although it is not necessary to repeat each test involving an administrative action with each interface. The evaluator shall ensure, however, that each supported method of administering the TOE that conforms to the requirements of this cPP be tested; for instance, if the TOE can be administered through a local hardware interface; SSH; and TLS/HTTPS; then all three methods of administration must be exercised during the evaluation team's test activities.

2.32.1 TSS

None

2.32.2 Guidance

[ST] Section 7.3.2 states that the TOE provides a local serial console and a remote web GUI management interface for local and remote administrative management of the TOE, respectively.

[AGD] Section 2.1 lists the physical interfaces that are used for management of the TOE. These include the serial port for local administration, and the internal and external ports which can both be used for remote administration. [AGD] Section 4.4 provides instructions for configuring networking for these interfaces.

[AGD] Section 4.3 describes how the admin can connect to the HTTPS/TLS web GUI for remote management. [AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance for configuring the TLS cryptographic engine which is necessary for the establishment of the Trusted Path between both the TOE the Security Administrative user. [AGD] Section 5.10 “Device Certificates” contains the guidance on configuring the TOE’s X.509v3 server certificate to present to the incoming TLS Client for authentication of the server. [AGD] Section 5.9 provides the guidance to import and trust the CA that signed the TOE’s server certificate for validation purposes.

[AGD] Section 5.3 provides guidance for configuring the local console for administrative access. [AGD] Section 4.2 provides guidance on connecting to the local console with a client terminal.

2.32.3 Testing

The evaluator verified the TOE could be administered locally (via local serial console, RS-232) and remotely (via HTTPS web GUI) by exercising some TSF configuration through each interface. Testing for FIA_UIA_EXT.1 User Identification and Authentication – TEST 1, Objective 1 and Objective 2 satisfied this test.

2.33 FPT_SKP_EXT.1 Protection of TSF Data (for reading of all symmetric keys)

TSS

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

2.33.1 TSS

[ST] Section 7.5.2 states that there is no interface provided to allow any user to view any passwords, pre-shared keys, symmetric keys and private keys.

[ST] Section 7.2.1 describes where private keys and ephemeral keys are stored. Persistent private keys are stored on internal hard disk drives in plaintext. These include the HTTPs/TLS private host key and the Syslog/TLS private client key. These keys are loaded into RAM when they are required for use by the TSF and subsequently zeroized when no longer needed. The HAWK secret key is also stored on internal hard disk drives and zerioized in the same manner.

[ST] Section 7.5.1, states that passwords are obfuscated using the SHA-256 algorithm, and that there is no interface to view the password hashes.

The TOE does not make use of Pre-Shared keys.

2.33.2 Guidance

None

2.33.3 Testing

None

2.34 FPT_APW_EXT.1 Protection of Administrator Passwords

TSS

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

2.34.1 TSS

[ST] Section 7.5.1 states that the TOE does not store plaintext passwords, and that each user's password is stored as a SHA-256 hash, and that there are no interfaces to view the password hashes.

2.34.2 Guidance

None

2.34.3 Testing

None

2.35 FPT_TST_EXT.1 TSF testing

TSS

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

Guidance Documentation

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

Tests

Future versions of this cPP will mandate a clearly defined minimum set of self tests. But also for this version of the cPP 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.

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 verify that the self tests described above are either carried out during initial start-up and that the developer has justified any deviation from this (if applicable).

2.35.1 TSS

[ST] Section 7.5.3 details the self-tests that are run by the TOE. These tests are performed at power-on. These tests include BIOS checks, cryptographic library functionality tests, and firmware checks. This section provides an outline of what each test consists of. The third from the last paragraph of this section provides an argument as to why the tests are sufficient to demonstrate that the TSF is operating correctly.

2.35.2 Guidance

[ST] Section 6.1.5.3 states the following TSF testing is performed by the TOE during initial start-up (on power on):

- BIOS checks
- Cryptographic library functionality test
- Firmware integrity checks

[AGD] Section 6.2.24 describes the log messages generated for successful and failed self-tests that occur during boot-up.

[AGD] Section 8 “Self-Test” describes the self-tests that are automatically run on the TOE during boot-up and the possible errors that can occur for each test block. These self-tests include:

- BIOS Checks
  - If these checks fail, the TOE will not boot-up. The admin is instructed to shutdown the TOE and contact Pulse Secure for support.
- Cryptographic Library Tests
  - If these checks fail, the cryptographic TSF will not start and audit record will be generated. The admin is instructed to shutdown the TOE and contact Pulse Secure for support.
- File integrity checks
  - If these checks fail, the TOE will continue to boot, and an audit record will be generated stating “Failed filesystem integrity check”. The admin is instructed to shutdown the TOE and contact Pulse Secure for support.

2.35.3 Testing

The evaluator verified the TOE performed the following self-tests:

- Verification of the integrity of the firmware and executable software of the TOE.
- Verification of the correct operation of the cryptographic functions necessary to fulfill any of the SFRs.

The evaluator authenticated to the local console as an administrator and proceeded to reboot the TOE. After the completed its boot up, the evaluator reviewed the audit records to verify the self-checks were performed. [AGD] states during a successful self-test, the audit records will not contain the following audit records:

- *Critical SYS31161 <current timestamp> - ive - [127.0.0.1] System()[] - Failed filesystem integrity check*
- *Critical ERR30967 <current timestamp> - ive - [127.0.0.1] System()[] – Unable to set FIPS mode for web server*

The evaluator confirmed these audit records were not in the local audit storage after successful boot up.

2.36 FPT_TUD_EXT.1 Trusted Update

TSS

**TSS Assurance Activity was modified by "TD0154: NIT Technical Decision for Versions of TOE Software in the NDcPP v1.0 and FW cPP v1.0".*

**TSS Assurance Activity was modified by "TD0094: NIT Technical Decision for validating a published hash in NDcPP".*

The evaluator shall verify that the TSS describes all TSF software update mechanisms for updating the system 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 ST author indicates that a certificate-based mechanism is used for software update digital signature verification, the evaluator shall verify that the TSS contains a description of how the certificates are contained on the device. The evaluator also ensures that the TSS (or guidance documentation) describes how the certificates are installed/updated/selected, if necessary.

**AGD Assurance Activity was modified by "TD0094: NIT Technical Decision for validating a published hash in NDcPP".*

Guidance Documentation

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.

Tests

**ATE Assurance Activity was modified by "TD0094: NIT Technical Decision for validating a published hash in NDcPP".*

The evaluator shall perform the following tests:

> Test 1: The evaluator performs the version verification activity to determine the current version of the product as well as the most recently installed version (should be the same version before updating). 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.
>
> Test 2 (if digital signatures are used): 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 gap between the installation of an update and a required reboot or activation to execute the updated code, 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.

Test 2 (if published hash is verified on the TOE): If the published hash is provided to the TOE by the Security Administrator and the verification of the hash value over the update file(s) against the published hash is performed by the TOE, then the evaluator shall perform the following tests. The evaluator first confirms that no update is pending and then performs the version verification activity to determine the current version of the product, verifying that it is different from the version claimed in the update(s) to be used in this test.

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

2. The evaluator uses a legitimate update and tries to perform verification of the hash value without storing the published hash value on 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 a gap between the installation of an update and a required reboot or activation to execute the updated code, 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.

The evaluator shall perform the Tests 1 and 2 for all methods supported (manual updates, automatic checking for updates, automatic updates).

If the verification of the hash value over the update file(s) against the published hash is not performed by the TOE, Test 2 shall be skipped.

2.36.1 TSS

[ST] Section 7.5.4 describes the Trusted Update process. This section states that the TOE performs an RSA 2048 SHA-256 digital signature verification of the update package using the Pulse Secure firmware update Public-Key. If the signature verification step fails, the TOE presents the user with an error message and discards the update. If the signature verification step is successful, the TOE installs the candidate update. The candidate updates are obtained by downloading them from the vendor's website.

The TOE does not utilize certificates for the Trusted Update process.

2.36.2 Guidance

[ST] Section 7.5.4 describes an RSA 2048-bit/SHA-256 digital signature verification of the update. This section states that if the signature verification check is successful, the TSF installs the update. If the signature verification check is unsuccessful, the TSF presents the user with an error message and discards the update.

[AGD] Section 4.7 describes software updates to the TOE. This section states that "The verification of the authenticity of the software package is performed by digital signature verification."

2.36.3 Testing

Test 1:

Objective 1: The evaluator verified the currently executing TOE firmware/software. The evaluator authenticated to the administrative web GUI and navigated to the system maintenance > platform configuration page and verified the current running software version.

Objective 2: The evaluator verified a legitimate update could successfully be installed onto the TOE. The evaluator authenticated to the administrative web GUI and navigated to the system maintenance > Upgrade/Downgrade configuration page, uploaded a legitimate update package, and verified the installation was successful.

Objective 3: The evaluator verified the admin could query the TOE's software/firmware version to verify the reported version matches the legitimate update version. After installing the legitimate update package, the evaluator authenticated to the administrative web GUI and navigated to the system maintenance > platform configuration page and verified the current running software version matched the legitimate update package version.

Test 2:

Objective 1: The evaluator verified the TOE does not accept update packages that are corrupt. The evaluator authenticated to the administrative web GUI and navigated to the system maintenance > Upgrade/Downgrade configuration page, uploaded a corrupted update package, and verified the installation was not successful.

Objective 2: The evaluator verified the TOE does not accept update packages that are unsigned. The evaluator authenticated to the administrative web GUI and navigated to the system maintenance > Upgrade/Downgrade configuration page, uploaded an update package that was not signed, and verified the installation was not successful.

Objective 3: The evaluator verified that the TOE does not accept update packages that are signed by an invalid signature. The evaluator authenticated to the administrative web GUI and navigated to the system maintenance > Upgrade/Downgrade configuration page, uploaded an update package that was signed with an invalid signature, and verified the installation was not successful.

2.37 FPT_STM.1 Reliable Time Stamps

TSS

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

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

Tests

The evaluator shall perform the following tests:

a) Test 1: The evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

b) Test 2: If the TOE supports the use of an NTP server; the evaluator shall use the guidance documentation to configure the NTP client on the TOE, and set up a communication path with the NTP server. The evaluator will observe that the NTP server has set the time to what is expected. If the TOE supports multiple protocols for establishing a connection with the NTP server, the evaluator shall perform this test using each supported protocol claimed in the guidance documentation.

If the audit component of the TOE consists of several parts with independent time information, then the evaluator shall verify that the time information between the different parts are either synchronized or that it is possible for all audit information to relate the time information of the different part to one base information unambiguously.

2.37.1 TSS

[ST] Section 7.5.5 discusses reliable time stamps. The TOE uses the system time for timestamp audit log records, to determine user session timeouts, to determine certificate validity, and for HAWK authentication. This sections states that these operations do not require accuracy finer than 1 second. The system time must be updated by the security administrator once a month to maintain accuracy.

2.37.2 Guidance

None

2.37.3 Testing

Test 1:

The evaluator verified the administrator could manually set the time and verified the time was set as configured. The evaluator authenticated to the administrative web GUI, navigated to the page for configuration of the Date and Time, manually configured the date and time, saved the changes, and then navigated to the System Overview page to verify that the Date and Time was set as configured.

Test 2:

The evaluator verified the TOE could be configured to use NTP as its reliable source of time and the TOE's time was successfully configured to the NTP server time. The evaluator authenticated to the administrative web GUI, navigated to the page for configuration of the Date and Time, configured the date and time to use NTP with a symmetric key, saved the changes, and then navigated to the System Overview page to verify the Date and Time was set as configured and was the same time as displayed by the IT Support VM's system time (ran timedatectl utility on the VM and compared its output to the TOE's system time). The evaluator ran a packet capture during this process and verified NTP updates were being passed between the TOE and the NTP server (IT Support VM). The evaluator verified in the audit records the TOE updated its system time based on NTP updates.

2.38 FTA_SSL_EXT.1 TSF-initiated Session Locking

Tests

The evaluator shall perform the following test:

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

2.38.1 TSS

None

2.38.2 Guidance

None

2.38.3 Testing

Test 1:

The evaluator verified that the local console session was terminated after the configured idle timeout period for 3 different time period settings. The evaluator authenticated to the administrative web GUI, navigated to the Administrator's Session Options configuration page and configured the 'idle timeout' value to 5 minutes and saved the changes. Then, configured the timer bash script for 300 seconds, authenticated to the local console and immediately started the timer script. The evaluator verified in the audit records and using the bash script timer that the TSF terminated the local console session after an idle period of 5 minutes. The evaluator verified this functionality for 5, 8 and 10 minute periods.

2.39 FTA_SSL.3 TSF-initiated Termination

Tests

The evaluator shall perform the following test:

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

2.39.1 TSS

None

2.39.2 Guidance

None

2.39.3 Testing

Test 1:

The evaluator verified that the web GUI session idle timeout configuration was enforced for 3 different time values. The evaluator authenticated to the administrative web GUI, navigated to the Administrator's Session Options configuration page and configured the 'idle timeout' value to 5 minutes and saved the changes, then logged out of the session. The evaluator then logged back into the administrative web GUI while immediately starting a digital timer. The evaluator did not perform any actions on the administrative web GUI for the length of the configured session inactivity time period plus a couple of seconds extra to allow for a buffer between clicking start on the timer and actual authentication to the TSF. After the configured time period elapsed, the evaluator attempted to navigate to a different page within the administrative web GUI. The evaluator verified the attempt to navigate the session failed and the session was terminated by the TSF. The TOE presented a page saying the session timed out. The evaluator verified in the audit records and using a digital timer the TSF terminated the local console session after an idle period of 5 minutes. The evaluator verified this functionality for 5, 8 and 10 minute periods.

2.40 FTA_SSL.4 User-initiated Termination

Tests

The evaluator shall perform the following tests:

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

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

2.40.1 TSS

None

2.40.2 Guidance

None

2.40.3 Testing

Test 1:

The evaluator verified the session was terminated after manually logging out of the local console session. The evaluator authenticated to the local console using administrative credentials then entered '11' and hit 'enter' to 'exit the serial console session'. The evaluator verified in the audit records and in the local console session that the session was terminated and the evaluator was presented again with a login prompt for the local console.

Test 2:

The evaluator verified the administrator could establish a remote administrative session to the TOE via the web GUI and could manually terminate the session. The evaluator authenticated to the administrative web GUI then clicked in the top right hand corner 'down-arrow' icon then clicked 'Logout'. The evaluator verified in the audit records and on the screen that the session was terminated.

2.41 FTA_TAB.1 Default TOE Access Banners

TSS

The evaluator shall check the TSS to ensure that it details each method of access (local and remote) available to the administrator (e.g., serial port, SSH, HTTPS).

Tests

The evaluator shall also perform the following test:

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

2.41.1 TSS

[ST] Section 7.6 states that administrative users have the remote web UI and local console CLI available to them, as described in [ST] Section 7.3.2. The referenced section (7.3.2) details the protocol for the remote web UI (HTTPs), and that the local console CLI is a serial connection.

2.41.2 Guidance

None

2.41.3 Testing

Test 1:

Objective 1: The evaluator verified the Notice and Consent Warning Message was displayed before authentication to the local console administrative interface. The evaluator opened a terminal session (using Putty.exe) to the local console of the TOE, pressed ‘any key’ and was prompted with the Warning and Consent Banner configured during setup of the TOE.

Objective 2: The evaluator verified the Notice and Consent Warning Message was displayed before authentication to the local console administrative interface. The evaluator navigated to the home page of the administrative web GUI and was immediately prompted with the Warning and Consent Banner configured during setup of the TOE.

2.42 FTP_ITC.1 Inter-TSF trusted channel

TSS

The evaluator shall examine the TSS to determine that, for all communications with authorized IT entities identified in the requirement, each communications mechanism is identified in terms of the allowed protocols for that IT entity. The evaluator shall also confirm that all protocols listed in the TSS are specified and included in the requirements in the ST.

Guidance Documentation

The evaluator shall confirm that the guidance documentation contains instructions for establishing the allowed protocols with each authorized IT entity, and that it contains recovery instructions should a connection be unintentionally broken.

Tests

The 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: The evaluators shall, for each protocol associated with each authorized IT entity tested during test 1, the connection is physically interrupted. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

Further assurance activities are associated with the specific protocols.

2.42.1 TSS

[ST] Section 7.7.1 states that communications with the remote syslog server (trusted IT entity) is performed using syslog over TLS as described in [ST] Sections 7.1.2 and 7.2.4.

[ST] Section 7.7.1 states that HTTPS/TLS is utilized for establishing a Trusted Channel connection to the Pulse One management server. This channel uses the HAWK authentication protocol for mutual authentication (identification of the TOE to the Pulse One Server; whereas the Pulse One server provides an x509 cert for assured identification).

[ST] Section 7.1.2 states that syslog over TLS is performed according to RFC 5425.

[ST] Section 7.2.4 specifies the TLS protocol in terms of the [PP] requirements placed on the selections made for FCSS_TLSC_EXT.2. These include, but are not limited to, listing the acceptable versions of TLS, listing the acceptable-mandatory and acceptable-optional TLS ciphersuites, describing the use of elliptic curve algorithms and listing their acceptable NIST curves, and a statement of the denial of non-accepted TLS/SSL versions.

2.42.2 Guidance

[ST] Section 6.1.7.1 states that TLS is utilized to provide a trusted channel to the audit server and HTTPS/TLS to the Pulse One management server (authorized IT entities).

[AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance for configuring the TLS cryptographic engine which is necessary for the establishment of the Trusted Channel between both the TOE and the syslog server, and between the TOE and the Pulse One management server.

[AGD] Section 5.9 “Import Trusted Server CA” provides the guidance to import and trust the CA that signed the syslog server’s certificate and the Pulse One server’s certificate for validation/authentication purposes.

[AGD] Section 5.17 “Integrate with Pulse One (Optional)” contains the guidance to configure the TOE with a registration code that is generated by the Pulse One management server and inputted into the TOE to allow Pulse One to authenticate the TOE for mutual authentication of the Trusted Channel. The TOE authenticates the Pulse One management server using Pulse One’s X.509v3 certificate that is presented in the TLS handshake.

[AGD] Section 5.11 states that TLS connections automatically attempt to reconnect if the connection is unintentionally broken.

2.42.3 Testing

Test 1:

Objective 1: The evaluator verified the Trusted Channel communication with the remote audit server could be successfully established using TLSC_EXT.2.x. This objective was performed during the FAU_STG_EXT.1 - Protected audit event storage Objective 1 test.

Objective 2: The evaluator verified the Trusted Channel communication with the remote audit server could be successfully established using TLSC_EXT.2.x. This test was performed during the FCS_TLSC_EXT.1.1 – Test 1.

Test 2:

The evaluator verified the TOE could initiate HTTPS sessions as a TLS client. This test was performed during the FCS_HTTPS_EXT.1 - HTTPS Protocol Test 1.

Test 3:

Objective 1: The evaluator verified the Trusted Channel connection between The TOE and the Pulse One Management server was encrypted with TLS. The evaluator established a session between the TOE and the Pulse One management server with a packet capture machine running wireshark. The TOE, Pulse One management server and the packet capture machine were each connected to a 8-port hub. The packet capture shows the connection successfully negotiating a TLS session between the TOE and Pulse One management server. The evaluator verified after the session was established, application data was sent encrypted.

Objective 2: The evaluator verified that the Trusted Channel connection between The TOE and the remote audit server is encrypted with TLS. The evaluator established a session between the TOE and the syslog audit server running on the IT Support VM with a packet capture running on the Kali VM. Objective 2 was satisfied by the testing for FAU_STG_EXT.1 - Protected audit event storage.

Test 4:

Objective 1: The evaluator verified audit data was not sent in the clear upon reconnection when the connection between the TOE and remote audit server was physically interrupted. The evaluator started a packet capture on the Kali VM and established the TLS session between the TOE and syslog audit server, then used a custom python script to force the TOE to generate an audit record. The evaluator physically unplugged the TOE's Ethernet cable from the testing environment then ran the custom python script again to attempt to have the TOE generate an audit record. The evaluator plugged the TOE's Ethernet cable back into the testing environment, waited 1 minute then ran the custom python script again to force the TOE to generate an audit record. The evaluator stopped the packet capture and verified tht in a search of the capture's contents the following string was not present in the clear:

- "PulseSecure"

In addition, the evaluator set a visual filter in the packet capture to only show syslog packets. The evaluator verified no syslog packets were sent in the clear.

Objective 2: The evaluator verified HTTP data was not sent in the clear upon reconnection when the connection between the TOE and Pulse One Management server was physically interrupted. The evaluator started a packet capture on the packet capture machine then authenticated to the administrative web GUI of the TOE and navigated to the Pulse One configuration page. The evaluator clicked on 'Renegotiate Credential' to force communication between the TOE and the Pulse One remote management server. Once the button was clicked, the evaluator immediately unplugged the Ethernet cable between the TOE and the Pulse One management server. The evaluator clicked on the 'Renegotiate Credential' button again, and then plugged the Ethernet cable back in and clicked 'Renegotiate Credential' again. The evaluator stopped the packet capture and verified there was no http traffic in the clear during the entirety of testing for objective 2 using a filter in the wireshark packet capture.

2.43 FTP_TRP.1 Trusted Path

TSS

The evaluator shall examine the TSS to determine that the methods of remote TOE administration are indicated, along with how those communications are protected. The evaluator shall also confirm that all

protocols listed in the TSS in support of TOE administration are consistent with those specified in the requirement, and are included in the requirements in the ST.

Guidance Documentation

The evaluator shall confirm that the guidance documentation contains instructions for establishing the remote administrative sessions for each supported method.

Tests

The evaluator shall perform the following tests:

a) Test 1: The evaluators shall ensure that communications using each specified (in the guidance documentation) remote administration method is tested during the course of the evaluation, setting up the connections as described in the guidance documentation and ensuring that communication is successful.

b) Test 2: 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: The evaluators shall ensure that, for each protocol associated with each authorized IT entity tested during test 1, the connection is physically interrupted. The evaluator shall ensure that when physical connectivity is restored, communications are appropriately protected.

Further assurance activities are associated with the specific protocols.

2.43.1 TSS

[ST] Section 7.7.2 states that HTTPs/TLS provides the Trusted Path for remote administration.

[ST] Section 7.2.3 states that HTTPs is performed according to RFC 2818.

[ST] Section 7.2.5 describes the TLS server protocol.

The protocols listed in the TSS for the Trusted Path are consistent with those specified for FTP_TRP.1

2.43.2 Guidance

[ST] Section 6.1.7.2 states that the TOE is capable of using TLS/HTTPS to provide a Trusted Path between the TOE and remote administrative users. There are no other evaluated means of remote administration of the TOE.

[AGD] Section 5.12 “Enable NDcPP Mode” contains the guidance for configuring the TLS cryptographic engine. The admin clicks the box “Turn on NDcPP Mode” which configures TLS to utilize the TLS version and ciphersuites that are selected in the [ST]. Additionally the admin must click on “Use 2048bit Diffie-Hellman key exchange” box and uncheck the “Enable support for SSL legacy renegotiation” box, to maintain the CC Evaluated configuration for TLS. These actions configure for the TLS Server and TLS Client cryptographic engines of the TOE.

[AGD] Section 5.10 “Device Certificates” contains the guidance on configuring the TOE’s X.509v3 server certificate to present to the incoming TLS Client for authentication of the server.

[AGD] Section 5.9 provides the guidance to import and trust the CA that signed the TOE’s server certificate for validation purposes.

2.43.3 Testing

Test 1:

The evaluator verified the remote administration method (web GUI | HTTPS) was tested during the course of the evaluation and that establishing a remote connection for administrative management was successful. The evaluator stated a packet capture then proceeded to authenticate to the administrative web GUI. After successful authentication to the TOE, the evaluator stopped the packet capture and verified the session was protected using TLS.

Test 2:

N/A – The Trusted Path via the Web GUI cannot be initiated by the TOE.

Test 3:

The evaluator verified the channel data was not sent in clear-text. The evaluator started a packet capture and proceeded to authenticate to the administrative web GUI. The evaluator then navigated to the system local user's configuration section and configured a new user with the following credentials:

- Username: plaintextdatahere
- Password: abcdefghhijklmnopqrstuvwxyz

The evaluator saved the changes, stopped the packet capture, and searched the packet capture for the clear-text string of the password. The evaluator verified the clear-text data was not present in the packet captures.

Test 4:

The evaluator verified that the HTTPS/TLS connection renegotiates a TLS session after a physical network interruption, without sending HTTP traffic in the clear during the process. The evaluator started a packet capture then logged into the administrative web GUI of the TOE. The evaluator physically removed the Ethernet cable between TOE and the test environment. The evaluator attempted to navigate to a new page within the administrative GUI and waited 30 seconds for the attempt to timeout, then plugged the TOE back into the test environment. The evaluator waited a couple of seconds then clicked on "Try Again" in the web browser, waited for the page to reload, then stopped the packet capture. The evaluator verified in the packet capture HTTP data was not sent in the clear using a visual filter for HTTP, and also verified a new TLS session was established.

3 SAR Assurance Activities and Results

3.1 ASE: Security Target Evaluation

3.1.1 ASE_CCL.1.8C

The evaluator shall check that the statements of security problem definition in the PP and ST are identical.

Result:

[ST] Section 2.3 “Conformance to Security Packages” contains the statement regarding conformance to Security Packages. This section does not claim conformance to any security packages.

3.1.2 ASE_CCL.1.9C

The evaluator shall check that the statements of security objectives in the PP and ST are identical.

Result:

[ST] Section 1.3.1 “TOE Product Type” identifies the TOE as a Network Device.

[PP] Section 1.2 “TOE Overview” indicates that compliant TOEs are Network Devices.

The evaluator determined that the TOE type as described in [ST] Section 1.3.1 is consistent with the compliant TOE types as described in the [PP].

3.1.3 ASE_CCL.1.10C

The evaluator shall check that the statements of security requirements in the ST include all the mandatory SFRs in the cPP, and all of the selection-based SFRs that are entailed by selections made in other SFRs (including any SFR iterations added in the ST). The evaluator shall check that if any other SFRs are present in the ST (apart from iterations of SFRs in the cPP) then these are taken only from the list of optional SFRs specified in the cPP (the cPP will not necessarily include optional SFRs, but may do so). If optional SFRs from the cPP are included in the ST then the evaluator shall check that any selection-based SFRs entailed by the optional SFRs adopted are also included in the ST.

Result:

[ST] Section 3 contains the statements of security problem definitions of the TOE.

[PP] Section 3 “Security Problem Definition” contains the statements of security problem definitions of the TOE.

The evaluator determined that the statements of security problem definitions in [PP] Section 3 “Security Problem Definition” are identical to the statements of security problem definitions in [PP] Section 3 “Security Problem Definition”.

3.2 ADV: Development

3.2.1 ADV_FSP.1

The evaluator shall check 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 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 documents to be examined for this assurance component in an evaluation are therefore the Security Target, AGD documentation, and any supplementary information required by the cPP for aspects such as entropy analysis or cryptographic key management architecture1: no additional “functional specification” documentation is necessary to satisfy the Evaluation Activities. The interfaces that need to be evaluated are also identified by reference to the assurance activities listed for each SFR, and are expected to be identified in the context of the Security Target, AGD documentation, and any supplementary information required by 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 Evaluation Activities for each SFR also means that the tracing required in ADV_FSP.1.2D is treated as implicit, and no separate mapping information is required for this element.

However, if the evaluator is unable to perform some other required Evaluation Activity 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’.

Result:

[ASU] Cell C3 lists the Web Interface, Serial Console, and Pulse ONE TSFI interfaces.

[ASU] Cell C4 and C6 state that the “internal”, “external” and “management” labeled ports are Ethernet ports which provide access/connectivity to the Web Management and Pulse ONE TSFI. In addition, the “Console” labeled port is an RJ-45 form factor port providing local console access.

[ASU] Cell E7 provides the parameters for the described interfaces.

[ASU] Cell C6 and E6 describe the purpose and method of use for each TSFI. Configuration of all the TSF can be administered through the Web Interface TSFI.

3.3 AGD: Guidance Documents

3.3.1 Operational User Guidance (AGD_OPE.1)

The evaluator shall check the requirements below are met by the guidance documentation.

Guidance documentation shall be 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.

Guidance documentation must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

The contents of the guidance documentation will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

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 generation of the hash/digital signature.

c) The TOE will likely contain security functionality that does not fall in the scope of evaluation under this cPP. The guidance documentation shall make it clear to an administrator which security functionality is covered by the Evaluation Activities.

Result:

[ASU] provides information on how guidance and preparative procedure documentation is identified and distributed to administrators of the TOE. [ASU] states that CC customers are notified of the guidance documentation and where to download via an email notification. [ASU] states that all documentation will be available to the customer via download from the vendor's website (www.pulsesecure.net). In addition, [ASU] states that every appliance is shipped with a Registration Card, EULA, and Tech Publication. The Registration Card contains the link to Pulse Secure's Techpub website to download CC documentation (guidance).

[AGD] Section 5.12 "Enable NDcPP Mode" provides guidance on configuring the TOE's TLS/HTTPS (Trusted Path) and TLS (Trusted Channel) cryptographic engines. This section includes enabling the 'NDcPP Mode' on the TOE which configures the TLS cryptographic engine for both the TLS server and TLS client of the TOE. Enabling the 'NDcPP Mode' configures the [ST] provisioned TLS ciphersuites and TLS protocol versions. The admin is instructed to check additional boxes to allow for 2048-bit DHE key-exchange, to disable SSL legacy renegotiation support, and to remove specified ciphersuites from the list of supported ciphersuites.

[AGD] Sections 5.8, 5.9, and 5.10 describe the certificate requirements for the TLS engines.

[ST] only claims one Operational Environment for the TOE. [ST] Section 1.3.4 lists the hardware and software components that the TOE requires to be present in the operational environment. These include the following components:

- Local Serial Console
  - RS-232
- TLS Client
  - Internet Explorer 11, Google Chrome 50, or Firefox 38
  - Supporting TLSv1.1 and/or TLSv1.2
  - Supporting at least one of the following ciphersuites:
    - TLS_RSA_WITH_AES_128_CBC_SHA
    - TLS_RSA_WITH_AES_256_CBC_SHA
    - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
    - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
    - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
    - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
    - TLS_RSA_WITH_AES_128_CBC_SHA256
    - TLS_RSA_WITH_AES_256_CBC_ SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
    - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
    - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- Syslog Server
  - Conformant with RFC 5424 (Syslog Protocol)
  - Supporting Syslog over TLS (RFC 5425)
  - Acting as a TLSv1.1 and/or TLSv1.2 server
  - Supporting Client Certificate authentication

    - Supporting at least one of the following cipher suites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_256_CBC_ SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- DNS Server
    - Conformant with RFC 1035
- NTP Server
    - [AGD] Section 4.6 states that the NTP server must be conformant with RFC 5905.
- CRL Server
    - Conformant to RFC 5280
- [For the MAG-SM160 and MAG-SM360 platforms only] Chassis, one of:
    - MAG6610
    - MAG6611
- Pulse One management server (Optional)
    - Pulse One v2.0 management server.

[ST] Section 1.2 identifies the TOE as:

- Pulse Policy Secure;
- Running the following software:
    - Pulse Policy Secure v5.3R4.10;
- Running on the following hardware platforms:
    - PSA300
    - PSA3000
    - PSA5000
    - PSA7000c
    - PSA7000f
    - MAG2600
    - MAG4610
    - MAG-SM160
    - MAG-SM360

[AGD] contains the guidance instructions for the each of the platforms listed above.

[AGD] Section 4.7 “Software Updates” contains the guidance for updating the TOE software. Software is verified via digital signature verification performed by the TOE.

[AGD] Section 4.7 includes the instructions for obtaining the update package itself, by downloading it from Pulse Secure’s Licensing and Download Center onto a trusted computer. Further guidance directs the admin to upload the trusted update package to the TOE and to click ‘Install’ to initiate the update process.

[AGD] Section 6.2.18 provides audit records for the successful and failed attempts at updating the TOE.

[ST] Section 1.4.2 describes the logical boundary of the TOE as being the security functions that are implemented exclusively by the TOE. [ST] Section 1.3.3 provides a summary of those security functions.

[ST] Section 1.4.2.8 lists the unevaluated functionality of the TOE.

[ST] Section 1.4.2.9 lists the excluded functionality of the TOE. [AGD] Section 1.6 provides guidance on disabling the following excluded functionality:

- DMI Agents ([AGD] Section 1.6.1)
- SNMP Traps ([AGD] Section 1.6.2)
- External Authentication Servers ([AGD] Section 1.6.3)

[AGD] Section 1.2 states that guidance does not provide configuration settings for the features of the TOE that are unevaluated.

3.3.2 Preparative Procedures (AGD_PRE.1)

The evaluator shall check the requirements below are met by the preparative procedures.

The contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

Preparative procedures shall be 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 contents of the preparative procedures will be verified by the Evaluation Activities defined below and as appropriate for each individual SFR in section 2 above.

In addition to SFR-related Evaluation Activities, the following information is also required.

Preparative procedures must include a description of how the adminstrator 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).

Preparative procedures must be provided for every Operational Environment that the product supports as claimed in the Security Target and must adequately address all platforms claimed for the TOE in the Security Target.

The preparative procedures must include

a) instructions to successfully install the TSF in each Operational Environment; and

b) instructions to manage the security of the TSF as a product and as a component of the larger operational environment; and

c) instructions to provide a protected administrative capability.

Result:

[ASU] provides information on how guidance and preparative procedure documentation is identified and distributed to administrators of the TOE. [ASU] states that CC customers are notified of the guidance documentation and where to download via an email notification. [ASU] states that all documentation will be available to the customer via download from the vendor's website (www.pulsesecure.net). In addition, [ASU] states that every appliance is shipped with a Registration Card, EULA, and Tech Publication. The Registration Card contains the link to Pulse Secure's Techpub website to download CC documentation (guidance).

The [ST] only claims one operational environment for the TOE. [ST] Section 1.3.4 lists the hardware and software components that the TOE requires to be present in the operational environment (some

components are optional and are identified as such in the list below). These include the following components:

- Syslog server
- DNS Server
- Local console
- NTP server
- Web browser (TLS Client)
- CRL Server
- Pulse One v2.0 Management Server (Optional)
- [For the MAG-SM160 and MAG-SM360 platforms only] Chassis, one of:
    - MAG6610
    - MAG6611

[AGD] Section 1.5 contains a description of the components in the Operational Environment. These include the following:

- Local Serial Console
    - RS-232
- TLS Client
    - Internet Explorer 11, Google Chrome 50, or Firefox 38
    - Supporting TLSv1.1 and/or TLSv1.2
    - Supporting at least one of the following ciphersuites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_256_CBC_ SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- Syslog Server
    - Conformant with RFC 5424 (Syslog Protocol)
    - Supporting Syslog over TLS (RFC 5425)
    - Acting as a TLSv1.1 and/or TLSv1.2 server
    - Supporting Client Certificate authentication
    - Supporting at least one of the following cipher suites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_256_CBC_ SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256

        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- DNS Server
    - Conformant with RFC 1035
- NTP Server
    - [AGD] Section 4.6 states that the NTP server must be conformant with RFC 5905.
- CRL Server
    - Conformant to RFC 5280
- [For the MAG-SM160 and MAG-SM360 platforms only] Chassis, one of:
    - MAG6610
    - MAG6611
- Pulse One management server (Optional)
    - Pulse One v2.0 management server.

[AGD] Section 9 “Security Objectives for the Operational Environment” contains the description of how the administrator verifies that the operational environment can fulfil its role to support the security objectives as listed in [ST] Section 4.1, for the Operational Environment.

[ST] only claims one Operational Environment for the TOE. [ST] Section 1.3.4 lists the hardware and software components that the TOE requires to be present in the operational environment. These include the following components:

- Local Serial Console
    - RS-232
- TLS Client
    - Internet Explorer 11, Google Chrome 50, or Firefox 38
    - Supporting TLSv1.1 and/or TLSv1.2
    - Supporting at least one of the following ciphersuites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_256_CBC_ SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- Syslog Server
    - Conformant with RFC 5424 (Syslog Protocol)
    - Supporting Syslog over TLS (RFC 5425)
    - Acting as a TLSv1.1 and/or TLSv1.2 server
    - Supporting Client Certificate authentication
    - Supporting at least one of the following cipher suites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_256_CBC_ SHA256

        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- DNS Server
    - Conformant with RFC 1035
- NTP Server
    - [AGD] Section 4.6 states that the NTP server must be conformant with RFC 5905.
- CRL Server
    - Conformant to RFC 5280
- [For the MAG-SM160 and MAG-SM360 platforms only] Chassis, one of:
    - MAG6610
    - MAG6611
- Pulse One management server (Optional)
    - Pulse One v2.0 management server.

[ST] Section 1.2 identifies the TOE as:

- Pulse Policy Secure;
- Running the following software:
    - Pulse Policy Secure v5.3R4.10;
- Running on the following hardware platforms:
    - PSA300
    - PSA3000
    - PSA5000
    - PSA7000c
    - PSA7000f
    - MAG2600
    - MAG4610
    - MAG-SM160
    - MAG-SM360

[ST] Section 1.4.1, Table 2, list multiple platforms for the TOE. The MAG-SM160 and MAG-SM360 platforms are required to be installed in one of the following chassis: MAG6610 or MAG6611. [HW5] Section “Installing the MAG-SM160 Kit” on page 32 and Section “Installing the MAG-SM360 Kit” on page 33, contains the guidance on installing the MAG-SM160 and MAG-SM360 platforms into the MAG6610 and/or MAG6611 chassis, respectively.

[AGD] contains the preparative procedures for the each of the platforms listed above.

[ST] only claims one Operational Environment for the TOE. [AGD] contains the instructions to install the TSF in the Operational Environment. The sections which contain the specific guidance for each TSF that is implemented or supported by the Operational Environment are listed throughout the assurance activities in this document.

[ST] Section 1.3.4 contains the list of IT entities that are required (or optional) to be present in the Operational Environment of the TOE to support the TSF. The list below contains those components that are required (or specifically identified as optional) in the Operational Environment for secure operation of the TOE, in addition, the list below provides the sections within the guidance documentation for which installation instructions can be found for each component:

- Local Serial Console
    - RS-232
- TLS Client

    - Internet Explorer 11, Google Chrome 50, or Firefox 38
    - Supporting TLSv1.1 and/or TLSv1.2
    - Supporting at least one of the following ciphersuites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_256_CBC_ SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- Syslog Server
    - Conformant with RFC 5424 (Syslog Protocol)
    - Supporting Syslog over TLS (RFC 5425)
    - Acting as a TLSv1.1 and/or TLSv1.2 server
    - Supporting Client Certificate authentication
    - Supporting at least one of the following cipher suites:
        - TLS_RSA_WITH_AES_128_CBC_SHA
        - TLS_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
        - TLS_RSA_WITH_AES_128_CBC_SHA256
        - TLS_RSA_WITH_AES_256_CBC_ SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
        - TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
        - TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
        - TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- DNS Server
    - Conformant with RFC 1035
- NTP Server
    - [AGD] Section 4.6 states that the NTP server must be conformant with RFC 5905.
- CRL Server
    - Conformant to RFC 5280
- [For the MAG-SM160 and MAG-SM360 platforms only] Chassis, one of:
    - MAG6610
    - MAG6611
- Pulse One management server (Optional)
    - Pulse One v2.0 management server.

[AGD] Section 4 provides guidance on the initial configuration to install the TOE in the Operational Environment.

[AGD] Section 3 provides guidance on ensuring secure acceptance of the TOE.

[AGD] Section 4 and 5 together contain the instructions to configure the TOE to provide a Trusted Path for remote administration over TLS.

3.4 ATE: Tests

3.4.1 Independent Testing (ATE_IND.1)

The evaluator shall examine the TOE to determine that the test configuration is consistent with the configuration under evaluation as specified in the ST.

The evaluator shall examine the TOE to determine that it has been installed properly and is in a known state.

The evaluator shall prepare a test plan that covers all of the testing actions for ATE_IND.1 in the CEM and in the SFR-related Evaluation Activities. While it is not necessary to have one test case per test listed in an Evaluation Activity, the evaluator must show in the test plan that each applicable testing requirement in the SFR-related Evaluation Activities is covered.

The test plan identifies the platforms to be tested, and for any platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no affect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary.

The test plan describes the composition and configuration of each platform to be tested, and any setup actions that are necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the functionality by the TOE and its platform. This also includes the configuration of any cryptographic engine to be used (e.g. for cryptographic protocols being evaluated).

The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives, and the expected results.

The test report (which could just be an updated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure, so that a fix was then installed and then a successful re-run of the test was carried out, then the report would show a “fail” result followed by a “pass” result (and the supporting details), and not just the “pass” result.

Result:

The evaluator completed all functional testing elements and determined that the TOE passed functional testing. Further details on this element are shown in individual sections above.

3.5 AVA: Vulnerability Assessment

3.5.1 Vulnerability Survey (AVA_VAN.1)

The evaluator shall document their analysis and testing of potential vulnerabilities with respect to this requirement. This report could be included as part of the test report for ATE_IND, or could be a separate document.

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.5. The evaluator shall then perform vulnerability analysis in accordance with Appendix A.4. The results of the analysis shall be documented in the report according to Appendix A.5.

Result:

The evaluation team performed a vulnerability assessment and penetration testing based on an initial port scan of the TOE. This comprehensive port scan identified any and all open ports and acquired all possible identifying information from the TOE. This information was compared to those services listed in the [ST], and used as input into the public domain search. (This step was performed several times. For additional information, see the Evaluation Technical Report.)

Based on the output from the port scan, CVEdetails.org and cve.mitre.org were searched with the following terms:

- Pulse Policy Secure
- Pulse Policy Secure 5.3R4.10
- pulse secure crypto library
- Pulse Secure Cryptographic Module 2.0
- IVE OS 2.0

Based on the results, no vulnerabilities exist in the TOE that are exploitable. No third party libraries were identified.

4 Testing Environment

The evaluation team performed the independent testing activities to confirm the TOE operates to the TOE security functional requirements as specified in the [ST] for a product claiming conformance to the collaborative Protection Profile for Network Devices Version 1.0, 27 February 2015. The evaluation team devised a Test Plan based on the Testing Assurance Activities specified in [NDcPP]. The Test Plan described how each test activity was to be performed. The evaluation team executed the tests specified in the Test Plan and documented the results in the Test Report.

Independent testing was performed at the UL facility in San Luis Obispo, CA. The evaluators received two platforms identified in the TOE. The hardware/software was provided in the same form that normal customers would receive it. The evaluator installed and configured the TOE in accordance with the vendor provided guidance documentation, and performed the testing procedures as described in the Test Documentation.

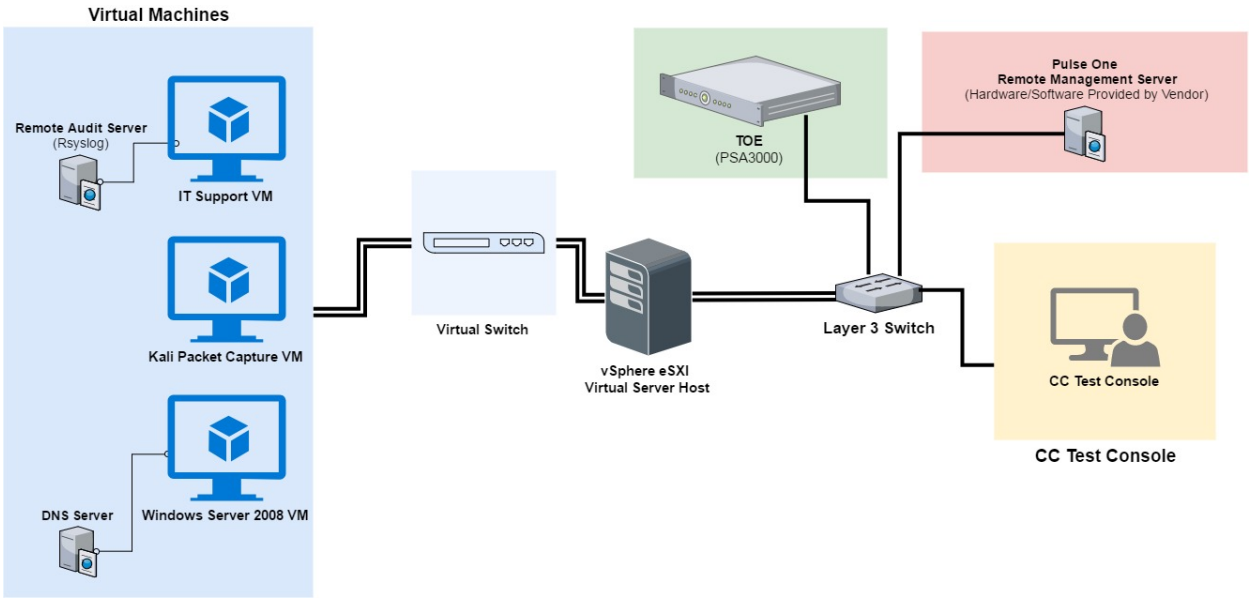

Figure 1 – Functional Testing Components Diagram

5 References

Abbr.	Name	Version	Date
[PP]	collaborative Protection Profile for Network Devices	1.0	February 27, 2015
[ST]	Pulse Policy Secure Security Target	1.0	September 5, 2017
[ENT]	Entropy Assessment - In PCS/PPS Series Appliances	4.10	June 2017
[AGD]	Pulse Policy Secure Operational User Guidance and Preparative Procedures	0.5	June 5, 2017
[ASU]	AssuranceDocument v1.4	1.4	August 5, 2016
[ADM]	Pulse Policy Secure Complete Software Guide	3.0	June 21, 2016
[HW1]	PSA300 Hardware Guide	1.0	April 2016
[HW2]	PSA3000 Hardware Guide	1.0	April 2016
[HW3]	PSA5000 Hardware Guide	1.0	April 2016
[HW4]	PSA7000 Hardware Guide	1.0	April 2016
[HW5]	MAG Series Pulse Secure Gateways Hardware Guide	1.0	September 2015