Assurance Activities Report

for

Guardtime Federal Black Lantern® BL300 Series and BL400 with BLKSI.2.2.1-FIPS

Version 1.0

1 September 2022

Prepared by:

Leidos Inc.
https://www.leidos.com/CC-FIPS140
Common Criteria Testing Laboratory
6841 Benjamin Franklin Drive
Columbia, MD 21046

Prepared for:

National Information Assurance Partnership
Common Criteria Evaluation and Validation Scheme

The Developer of the TOE:

Guardtime Federal
1700 Diagonal Road, Suite 320,
Alexandria, VA 22314

The TOE Evaluation was Sponsored by:

Guardtime Federal
1700 Diagonal Road, Suite 320,
Alexandria, VA 22314

Evaluation Personnel:

Dawn Campbell
Pascal Patin
Allen Sant

Common Criteria Version:

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

Common Evaluation Methodology Version:

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

Protection Profile:

- *collaborative Protection Profile for Network Devices*, Version 2.2e, 23 March 2020 [NDcPP]
- *Evaluation Activities for Network Device cPP*, Version 2.2, December 2019 [SD-ND]

Contents

1 Introduction

This document presents results from performing assurance activities associated with the Guardtime Federal Black Lantern® BL300 Series and BL400 with BLKSI.2.2.1-FIPS evaluation consisting of the BL300-B2, BL300-C2, and BL400-A1 appliances with firmware version BLKSI.2.2.1-FIPS. This report contains sections documenting the performance of assurance activities associated with each of the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) as specified in Evaluation Activities for Network Device cPP, Version 2.2, December 2019 and including the following optional and selection-based SFRs: FAU_STG.1; FAU_STG_EXT.2/LocSpace; FAU_STG_EXT.3/LocSpace; FCS_HTTPS_EXT.1; FCS_NTP_EXT.1; FCS_TLSC_EXT.1; FCS_TLSC_EXT.2; FCS_TLSS_EXT.1; FCS_TLSS_EXT.1; FCS_TLSS_EXT.2; FIA_X509_EXT.1/Rev; FIA_X509_EXT.2; FIA_X509_EXT.3; FMT_MOF.1/Functions; and FMT_MTD.1/CryptoKeys.

Note that, in accordance with NIAP Policy Letter #5, all cryptography in the TOE for which NIST provides validation testing of FIPS-approved and NIST-recommended cryptographic algorithms and their individual components must be NIST validated. The CCTL will verify that the claimed NIST validation complies with the NIAP-approved PP requirements the TOE claims to satisfy. The CCTL verification of the NIST validation will constitute performance of the associated assurance activity. As such, Test activities associated with functional requirements within the scope of Policy Letter #5 are performed by verification of the relevant CAVP certification and not through performance of any testing as specified in the PP or its supporting document.

1.1 Applicable Technical Decisions

The NIAP Technical Decisions referenced below apply to [NDcPP]. Rationale is included for those Technical Decisions that do not apply to this evaluation.

TD0527: Updates to Certificate Revocation Testing (FIA_X509_EXT.1)

This TD is applicable to the TOE.

TD0528: NIT Technical Decision for Missing EAs for FCS_NTP_EXT.1.4

This TD is applicable to the TOE.

TD0536: NIT Technical Decision for Update Verification Inconsistency

This TD is applicable to the TOE.

TD0537: NIT Technical Decision for Incorrect Reference to FCS_TLSC_EXT.2.3

This TD is applicable to the TOE. However, it relates to the cPP, so no changes to the evaluation activities are required.

TD0538: NIT Technical Decision for Outdated link to Allowed-with List

This TD is applicable to the TOE. However, it relates to the cPP, so no changes to the evaluation activities are required.

TD0546: NIT Technical Decision for DTLS - clarification of Application Note 63

N/A – The ST does not claim FCS_DTLSC_EXT.1.

TD0547: NIT Technical Decision for Clarification on developer disclosure of AVA_VAN

This TD is applicable to the TOE.

TD0555: NIT Technical Decision for RFC Reference incorrect in TLSS Test

This TD is applicable to the TOE.

TD0556: NIT Technical Decision for RFC 5077 question

This TD is applicable to the TOE.

TD0563: NIT Technical Decision for Clarification of audit date information

This TD is applicable to the TOE.

TD0564: NIT Technical Decision for Vulnerability Analysis Search Criteria

This TD is applicable to the TOE.

TD0569: NIT Technical Decision for Session ID Usage Conflict in FCS_DTLSS_EXT.1.7

This TD is applicable to the TOE.

TD0570: NIT Technical Decision for Clarification about FIA_AFL.1

This TD is applicable to the TOE.

TD0571: NIT Technical Decision for Guidance on how to handle FIA_AFL.1

This TD is applicable to the TOE.

TD0572: NIT Technical Decision for Restricting FTP_ITC.1 to only IP address identifiers

This TD is applicable to the TOE.

TD0580: NIT Technical Decision for clarification about use of DH14 in NDcPPv2.2e

This TD modifies one of the selections available in FCS_CKM.1, but the ST has not included that selection. The TD is therefore not applicable to the TOE.

TD0581: NIT Technical Decision for Elliptic curve-based key establishment and NIST SP 800-56Arev3

This TD is applicable to the TOE.

TD0591: NIT Technical Decision for Virtual TOEs and hypervisors

This TD modifies the wording of an assumption that is applicable to the TOE (A.LIMITED_FUNCTIONALITY) and is therefore applicable to the TOE in general. Note that the TOE does not have a virtual component.

TD0592: NIT Technical Decision for Local Storage of Audit Records

This TD is applicable to the TOE.

TD0631: NIT Technical Decision for Clarification of public key authentication for SSH Server

This TD provides clarification of public key authentication for SSH Server, but the ST does not claim FCS_SSHS_EXT.1.

TD0632: NIT Technical Decision for Consistency with Time Data for vNDs

This TD modifies FPT_STM_EXT.1 to cater for time stamps obtained from a virtualization platform, however the TOE does not include virtualized devices. Therefore the TD is not applicable to the TOE.

TD0633: NIT Technical Decision for IPsec IKE/SA Lifetimes Tolerance

This TD modifies evaluation activities associated with FCS_IPSEC_EXT.1, but the ST does not claim this SFR.

TD0634: NIT Technical Decision for Clarification required for testing IPv6.

This TD is applicable to the TOE.

TD0635: NIT Technical Decision for TLS Server and Key Agreement Parameters.

This TD is applicable to the TOE.

TD0636: NIT Technical Decision for Clarification of Public Key User Authentication for SSH

This TD provides clarification of public key authentication for SSH client, but the ST does not claim FCS_SSHC_EXT.1.

1.2 Evidence

[ST] Guardtime Federal Black Lantern® BL300 Series and BL400 with BLKSI.2.2.1-FIPS Security Target, Version 1.0, 6 July 2022

[Guide] Guardtime Federal Black Lantern® Guidance Documentation (Guide), v1.0, July 6, 2022

1.3 Conformance Claims

Common Criteria Versions

- Common Criteria for Information Technology Security Evaluation Part 1: Introduction, Version 3.1, Revision 5, dated: April 2017.
- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, Revision 5, dated: April 2017.
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Revision 5, dated: April 2017.

Common Evaluation Methodology Versions

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

Protection Profiles

- [NDcPP] collaborative Protection Profile for Network Devices, Version 2.2e, March 23, 2020
- [SD-ND] Evaluation Activities for Network Device cPP, Version 2.2, December 2019

1.4 SAR Evaluation

The following Security Assurance Requirements (SARs) were evaluated during the evaluation of the TOE:

SAR	Verdict
ASE_CCL.1	Pass
ASE_ECD.1	Pass
ASE_INT.1	Pass
ASE_OBJ.1	Pass
ASE_REQ.1	Pass
ASE_TSS.1	Pass
ADV_FSP.1	Pass
AGD_OPE.1	Pass
AGD_PRE.1	Pass
ALC_CMC.1	Pass
ALC_CMS.1	Pass
ATE_IND.1	Pass
AVA_VAN.1	Pass

The evaluation work units are listed in the proprietary ETR. The evaluators note per the PP evaluation activities that many of the SARs were successfully evaluated through completion of the associated evaluation activities present in the claimed PP.

2 Security Functional Requirement Evaluation Activities

This section describes the evaluation activities associated with the SFRs defined in the ST and the results of those activities as performed by the evaluation team. The evaluation activities are derived from [ND-SD] and modified by applicable NIAP Technical Decisions. Evaluation activities for SFRs not claimed by the TOE have been omitted.

2.1 Security Audit (FAU)

2.1.1 Audit Data Generation (FAU_GEN.1)

2.1.1.1 TSS Activities

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

[ST] Section 5.1.1 states that for audit records involving Generating/import of, changing, or deleting of cryptographic keys, the TOE identifies cryptographic keys by identifying the certificate associated with the key, using the certificate subject identifier and certificate issuer identifier.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.1.2 Guidance Activities

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

[Guide] section 6.4 “Audit Events” provides samples of auditable event log entries. The evaluator checked and ensured that there is at least one instance of each auditable event, comprising the mandatory, optional and selection-based SFR sections as applicable and required by FAU_GEN.1.

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

The evaluator examined the supplied guidance documentation, identifying all mechanisms available to the administrator for configuring and managing the capabilities of the TOE. Those mechanisms related to the SFRs specified in the ST were identified and mapped to the applicable SFRs. In addition, the evaluator sought to confirm that all SFRs that would be expected to have a management capability related to them had appropriate management capabilities identified in the guidance documentation.

The relevant administrative actions related to TSF data related to configuration changes comprise:

- Configuring the banner displayed prior to authentication ([Guide] 7.3)
- Specifying the session inactivity time-out period for local administrative sessions ([Guide] 3.2.3.4, 7.2.1)
- Initiating manual updates to the TOE and verifying updates prior to installation ([Guide] 7.7, 8.1, 7 Table 1)
- Configuring parameters associated with the authentication failure mechanism ([Guide] 7.2.1 describes using the `setconfig` command with security.maxloginretriesbeforedisable to configure the number of allowed login failures before user account disabling.)
- Configuration of audit behavior (i.e., configure local log storage size, configure TOE behavior when local audit storage space is full) ([Guide] 6.2) and clear the local audit storage—by purging the entire local audit log, or by removing a subset of the local log data ([Guide] 6.2.2)
- Configure the list of TOE-provided services available before an entity is identified and authenticated, per FIA_UIA_EXT.1 ([Guide] 7.4, 7.7)
- User account and role management ([Guide] 3.2.3.1)
- Management of cryptographic keys—generate CSRs and generate/remove keys ([Guide] 5.2.1.1, pg. 77-78 “rm [OPTION]... PATH”)
- Re-enabling an administrator account ([Guide] 7.2.1 – moduser)
- Setting the date and time ([Guide] 7.5.2)
- Configuring NTP ([Guide] 7.5.1)
- Managing the TOE’s trust store and designating X.509v3 certificates as trust anchors ([Guide] 5.2)
- Importing X.509 v3 certificates to the TOE’s trust store ([Guide] 5.2)
- Setting the length requirement for passwords ([Guide] 3.2.3.2.2, 7.2.1 includes an example).

  [Guide] Table 1 identifies all of the management functions and commands.

  [Guide] Section 7.2.1 describes the parameters that can be configured using the `setconfig` command.

  [Guide] Section 10 provides a list configuration parameters; possible values; and default values (if applicable).

  [Guide] Section 7.2.1 identifies the key security.minpasswordlength: the minimum password length parameter, set by using `setconfig` command and provides an example of its use. It identifies possible values of 8-32 configured with command “`setconfig` security MinPasswordLength”.

  [Guide] Section 8 details the serial console interface commands available to the Security and Network administrators. Section 9 describes how to use the PUT and GET methods for the RESTful APIs.

2.1.1.3 Test Activities

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

Audit logs were generated for each of the events required for FAU_GEN.1. Each audit was ensured to contain the specified information by the SFR table.

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

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.2 User Identity Association (FAU_GEN.2)

2.1.2.1 TSS & Guidance Activities

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

2.1.2.2 Test Activities

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

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.3 Protected Audit Trail Storage (FAU_STG.1)

2.1.3.1 TSS Activities

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

Section 5.1.2 of [ST] ("Audit Storage and Audit Record Export") states the TOE is a single standalone appliance that stores audit records locally. The local audit storage size is configurable from 500MB to 2GB. The logs comprising the audit trail are stored in the TOE's file system and protected from unauthorized modification and deletion by file system permissions.

The Black Lantern Security Administrator can enable and disable generation of audit records and can configure the behavior of the TOE when local audit storage is full. By default, the TOE will overwrite the oldest locally stored audit record with the newly generated audit record. If the Security Administrator disables this option, the TOE drops all new records and keeps a counter of the audit records dropped when the local storage is full. There are no situations in which lost audit data is not counted. The Black Lantern Security Administrator is able to view the count of dropped audit records and clear local storage. There are two methods to clear the local storage—by removing the entire local storage data, or by removing a subset of the local log data.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.3.2 Guidance Activities

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

[Guide] Section 6 "Audit Functionality" states the TOE does not provide any interfaces to modify audit records. As such, no configuration is necessary to protect the locally stored audit data against unauthorized modification. 6.2.2 states that the Black Lantern TOE protects itself against unauthorized modification and deletion of local audit logs by only permitting administrator users with Security Administrator role to manage the logging functionalities, which includes the clearing of local logs.

2.1.3.3 Test Activities

The evaluator shall perform the following tests:

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

The evaluator authenticated to the TOE as a user without administrator permissions. The evaluator attempted to clear the logs. The evaluator observed that the attempt fails.

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.

The evaluator authenticated to the TOE as a user with administrator permissions. The evaluator attempted to clear the logs. The evaluator observed that the attempt succeeded. The evaluator verified that the specified audit files were deleted.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.4 Protected Audit Event Storage (FAU_STG_EXT.1)

2.1.4.1 TSS Activities

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

Section 5.1.2 of [ST] ("Audit Storage and Audit Record Export") states the TOE can be configured to export audit records to an external audit server over a trusted channel protected by TLS. In this circumstance, the TOE acts as a TLS client. The audit records are exported in real time (i.e., as they are generated).

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.

Section 5.1.2 of [ST] states the local audit storage size is configurable from 500MB to 2GB.

Users with the Security Admin role can enable and disable generation of audit records and can configure the behavior of the TOE when local audit storage is full. By default, the TOE will overwrite the oldest locally stored audit record with the newly generated audit record. If the Security Administrator disables this option, the TOE drops all new records and keeps a counter of the audit records dropped when the local storage is full. There are no situations in which lost audit data is not counted. The Black Lantern Security Administrator is able to view the count of dropped audit records and clear local storage. There are two methods to clear the local audit storage—by purging the entire local audit log, or by removing a subset of the local log data. In both cases, clearing local audit storage resets the counter of dropped audit records to 0.

Section 5.1.2 of [ST] states the logs comprising the audit trail are stored in the TOE's file system and protected from unauthorized modification and deletion by file system permissions.

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

Section 5.1.2 of [ST] states the TOE is a standalone TOE that stores audit data locally.

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.

According to section 5.1.2 of [ST], the behavior of the TOE for when local audit storage is full is configurable. By default, the TOE will overwrite the oldest locally stored audit record with the newly generated audit record. If the Security Administrator disables this option, the TOE drops all new records and keeps a counter of the audit records dropped when the local storage is full.

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

Section 5.1.2 of [ST] states the TOE exports audit records in real time (i.e., as they are generated).

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

The TOE is not distributed. Therefore, this activity is not applicable.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.4.2 Guidance Activities

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

[Guide] Section 5.3.1.1 for instructions on configuring the trusted communication channel to a remote logging server and enabling remote audit storage. The guidance identifies the need to use the `setconfig` command to enable remote logging and TLS, and identify the logging server. The description includes a discussion of installing and use of the logging server's certificate and certificate chain in order to validate the X.509 certificate presented by the external audit server when establishing the connection. It also identifies the external audit server must support TLS v1.2 in order to be able to establish the trusted channel between the TOE and 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.

[Guide] Section 6 "Audit Functionality" states local and remote logging are independent capabilities and do not have behavioral impact on one another. When both capabilities are enabled, audit data is logged locally and then remotely in real-time as generated.

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.

[Guide] Section 6.2 describes the options for FAU_STG_EXT.1.3 and the resulting behavior for each. Section 6.2.1 states that once local logging storage is full, old local log data is by default overwritten with the newest data. The `setconfig` command to used configure the management.locallogkeepnewest parameter. The default value of parameter is 1 meaning that the oldest log entries are overwritten when local storage is full. Otherwise, if it is not set (e.g. it is configured to '0' value), the behavior is to drop newest entries.

This corresponds to the description in the TSS Section 5.1.2.

2.1.4.3 Test Activities

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

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

The evaluator configured the TOE to transmit logs to a remote syslog server which is configured to use protected syslog settings. The evaluator ensured that the logs were transmitted in a secure manner and not sent in plaintext, and the logs received by the remote syslog server were intact.

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

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

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

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

The evaluator configured the TOE to drop new audit data when the storage is full. The evaluator caused the storage to be exhausted. The evaluator ensured that once the storage was full the new audits were dropped, current audit data unchanged and a count of the audits dropped were maintained.

The evaluator cleared the logs.

The evaluator configured the TOE to overwrite the oldest audit records when the storage is full. The evaluator caused the storage to be exhausted. The evaluator ensured that once the storage was full the oldest audits were overwritten, and the new audits were captured.

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

The evaluator checked and ensured that the numbers supplied were correct.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.1.5 Counting Lost Audit Data (FAU_STG_EXT.2/LocSpace)

2.1.5.1 TSS Activities

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

Section 5.1.2 of [ST] states that by default, the TOE will overwrite the oldest locally stored audit record with the newly generated audit record and keep a count of the number of records the TOE has overwritten. The TOE displays the overwritten counter value as a warning whenever an administrator invokes the `viewlog` command. If an administrator disables this option, the TOE drops all new records and keeps a counter of the audit records dropped when the local storage is full.

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

The TOE is not distributed and therefore this activity is not applicable.

2.1.5.2 Guidance Activities

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

[Guide] Section 6.2.1 describes the possible configuration options and the meaning of the result returned by the TOE for each possible configuration. It states that once local logging storage is full, old local log data is, by default, overwritten with the newest and a counter of all overwritten log entries will begin incrementing to track these entries. This overwritten counter value will be available as a warning whenever the viewlog command is invoked. In addition, this overwrite log entry behavior can be changed to drop log entry behavior (and maintaining a dropped count) by disabling overwriting. Sample TOE output shows how many log entries were dropped/overwritten. Section 6.2 identifies the management.locallogkeepnewest setting which can be used to change the overwrite default behavior to drop newest behavior. Table 10 identifies the default value is indeed '1' for overwrite and can be disabled using '0' to drop newest. The description is consistent with [ST] Section 5.1.2.

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

[Guide] Section 6.2.2 states that when the local storage is cleared the operation cannot be reversed and all locally stored audit data will be permanently removed from Black Lantern.

2.1.5.3 Test Activities

The evaluator shall verify that the numbers provided by the TOE according to the selection for FAU_STG_EXT.2/LocSpace are correct when performing the tests for FAU_STG_EXT.1.3.

The evaluator checked and ensured that the numbers supplied were correct.

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

N/A, The TOE is not distributed

2.1.6 Action in Case of Possible Audit Data Loss (FAU_STG_EXT.3/LocSpace)

2.1.6.1 TSS Activities

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

Section 5.1.2 of [ST] states that the TOE reports audit log warning messages when the local storage has been reduced to 25%, 15%, 10%, 5%, 4%, 3%, 2%, and 1% of available storage space. The warning messages are written to the audit log.

For distributed TOEs the evaluator shall examine the TSS to ensure it describes to which TOE components this SFR applies and how each TOE component realises this SFR. Since this SFR is optional, it might only apply to some TOE components but not all. This might lead to the situation where all TOE components store their audit information themselves but FAU_STG_EXT.3/LocSpace is supported only by one of the components. In particular, the evaluator has to verify, that the TSS describes for every component supporting this functionality, whether the warning is generated by the component itself or through another component and name the corresponding component in the latter case. The evaluator has to verify that the TSS makes clear any situations in which audit records might be 'invisibly lost'.

The TOE is not distributed and therefore this is not applicable.

2.1.6.2 Guidance Activities

The evaluator shall also ensure that the guidance documentation describes how the Security Administrator 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.

[Guide] Section 6.2.1 states that Black Lantern warns the Security Administrator once there is 25% local storage space remaining by issuing log storage warning audit records. Additional warnings are issued when the remaining capacity reaches 15%, 10%, 5%, 4%, 3%, 2%, and 1%. This corresponds to the description in the TSS.

2.1.6.3 Test Activities

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

The evaluator observed that the TOE generated audit records for each of the specified remaining storage amounts defined, 25%, 15%, 10%, 5%, 4%, 3%, 2%, and 1% of available storage space.

For distributed TOEs the evaluator shall verify the correct implementation of display warning for local storage space for all TOE components that are supporting this feature according to the description in the TSS. The evaluator shall verify that each component that supports this feature according to the description in the TSS is capable of generating a warning itself or through another component.

The TOE is not distributed and therefore this is not applicable.

2.2 Cryptographic Support (FCS)

2.2.1 Cryptographic Key Generation (FCS_CKM.1)

2.2.1.1 TSS Activities

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

Section 5.2.3 of [ST] (“Cryptographic Key Generation and Establishment”) identifies the key sizes supported by the TOE. The TOE supports the following key generation schemes and their usage:

- RSA schemes supporting cryptographic key sizes of 2048 or 4096 bits, for TLS authentication.

- ECC schemes supporting NIST curves P-256, P-384, and P-521 with key sizes of 256, 384, and 521 bits, for TLS authentication and key establishment.

TLS is used for remote management and communication with the external audit server.

2.2.1.2 Guidance Activities

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

[Guide] Section 5.2.1.1 describes how to generate 2048-bit or 4096-bit RSA and 256, 384, 521-bit ECC key pairs using the `genkey` command. Keys generated by the Black Lantern can be used for Black Lantern certificate (localhost) generation. The subsection 'Certificate Signing Request (CSR) generation' describes how Black Lantern can be configured to use one of the previously generated key pairs when generating a CSR.

2.2.1.3 Test Activities

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

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

a) Random Primes:

•Provable primes

• Probable primes

b) Primes with Conditions:

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

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

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

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

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test

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

FIPS 186-4 Public Key Verification (PKV) Test

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

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. 54 The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

• Primes q and p shall both be provable primes

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

and two ways to generate the cryptographic group generator g:

• Generator g constructed through a verifiable process

• Generator g constructed through an unverifiable process.

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

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

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

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

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

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

• g != 0,1

• q divides p-1

• g^q mod p = 1

• g^x mod p = y

for each FFC parameter set and key pair.

Section 5.2.1 of [ST] (“Cryptographic Operations”), Table 7 (“Cryptographic Functions Implemented by CSL Direct v2.0.0”) identifies the CAVP certifications verifying asymmetric key generation, as follows.

Algorithm	Tested Capabilities	Certificates
RSA schemes using cryptographic key sizes of 2048-bit or 4096-bit that meet FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.3	Key Generation Mode: B.3.3 Properties: Modulo: 2048 Primality Tests: C.3 Properties: Modulo: 4096 Primality Tests: C.3 Public Exponent Mode: Fixed Public Key Format: Standard	A1515 RSA KeyGen (FIPS 186-4)
ECC schemes using “NIST curves” P-256, P-384, P-521, that meet FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4	Curve: P-256, P-384, P-521 Secret Generation Mode: Testing Candidates	A1515 ECDSA KeyGen (FIPS186-4)

Modified by TD0580

FFC Schemes using “safe-prime” groups

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

2.2.2 Cryptographic Key Establishment (FCS_CKM.2)

2.2.2.1 TSS Activities

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

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

Scheme	SFR	Service
RSA	FCS_TLSS_EXT.1	Administration
ECDH	FCS_SSHC_EXT.1	Audit Server
ECDH	FCS_IPSEC_EXT.1	Authentication Server

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

Section 5.2.3 of [ST] (“Cryptographic Key Generation and Establishment”) Table 9 identifies the following key establishment methods supported by the TOE:

- Elliptic curve-based key establishment schemes.

This key establishment method corresponds to the *ECC schemes using 'NIST curves'* key generation scheme specified in FCS_CKM.1.

Section 5.2.3 of [ST] identifies the usage for each scheme, as follows:

Scheme	SFR	Service
Elliptic curve-based	FCS_TLSC_EXT.1 FCS_TLSS_EXT.1	Audit Server Administration

2.2.2.2 Guidance Activities

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

[Guide] Section 5.3.3.3 "Key Establishment" provides instructions to ensure the use of Elliptic Curve-based key establishment for TLS communication. The TOE supports only one scheme: Elliptic Curve-based and so there is no configuration to ensure it is the scheme used. The instructions re-iterate the need to load appropriate certificates (as was described earlier).

2.2.2.3 Test Activities

Key Establishment Schemes The evaluator shall verify the implementation of the key establishment schemes of the supported by the TOE using the applicable tests below. ***SP800-56A Key Establishment Schemes*** Performed in accordance with NIAP Policy Letter #5.

Section 5.2.1 of [ST] ("Cryptographic Operation"), Table 7 ("Cryptographic Functions Implemented by CSL Direct v2.0.0") identifies the CAVP certifications verifying SP 800-56A key establishment schemes, as follows.

Algorithm	Tested Capabilities	Certificates
Elliptic curve-based key establishment schemes that meet NIST Special Publication 800-56A Revision 3, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"	Domain Parameter Generation Methods: P-256, P-384, P-521 Scheme: Ephemeral Unified: KAS Role: Initiator, Responder	A1515 KAS-ECC-SSC Sp800-56Ar3

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

The TOE does not implement any RSA-based key establishment schemes.

Removed by TD0580

~~***Diffie-Hellman Group 14***~~

~~The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_TRP.1/Admin, FTP_TRP.1/Join, FTP_ITC.1 and FPT_ITT.1 that uses Diffie-Hellman group 14.~~

FFC Schemes using "safe-prime" groups

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

The TOE does not use FFC Schemes and this activity is not applicable.

2.2.3 Cryptographic Key Destruction (FCS_CKM.4)

2.2.3.1 TSS Activities

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

Section 5.2.4 of [ST] ("Cryptographic Key Destruction"), Table 10 ("Private Keys, Symmetric Keys, and CSPs") lists all relevant keys and includes the following information: origin, storage location of the key or CSP; the purpose of the key or CSP and how it is encrypted.

Section 5.2.4 states the TOE does not store plaintext keys in non-volatile memory—all keys are encrypted at rest using 256 bit AES. For keys stored in or decrypted into volatile memory, once the keys are no longer needed, the TOE deallocates the memory back to the kernel. The memory is zeroized when power is removed from the TOE.

The root or top-level key-encrypting key is also an AES 256 bit key, derived from a special hardware-based secret value called the OTPMK (one time programmable master key). The OTPMK is implemented in specially-designed circuitry by the chip manufacturer. The vendor uses this key value to protect long-term keys stored on the TOE at rest.

The evaluator confirmed the description of keys is consistent with the functions performed by the TOE.

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

Section 5.2.4 of [ST] states that the TOE does not store plaintext keys in non-volatile memory—all keys are encrypted at rest using 256 bit AES.

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

The ST selects "invocation of an interface" in FCS_CKM.4 for keys stored in non-volatile memory. However, the TSS Section 5.2.4 states that the TOE does not store plaintext keys in non-volatile memory—all keys are encrypted at rest using 256 bit AES. As such, the SFR and this activity are vacuously satisfied.

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

Section 5.2.4 of [ST] indicates that all keys stored in a non-plaintext form are encrypted using 256 bit AES, using a root or top-level key-encrypting key (kek). This kek is also an AES 256 bit key, derived from a special hardware-based secret value called the OTPMK (one time programmable master key).

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

Section 5.2.4 of [ST] states there are no configurations or circumstances that do not conform to the key destruction requirement.

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

The ST does not specify the use of "a value that does not contain any CSP" to overwrite keys.

2.2.3.2 Guidance Activities

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

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

[Guide] Section 5.3.3.4 “Key Destruction” states there are no configurations or circumstances that do not conform to the plaintext key zeroized destruction method.

2.2.3.3 Test Activities

None defined.

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

2.2.4.1 TSS Activities

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

Section 5.2.1 of [ST] (“Cryptographic Operations”) states the TOE performs AES encryption and decryption in accordance with ISO 18033-3, with key size of 256 bits, in the GCM mode of operation as specified in ISO 19772.

2.2.4.2 Guidance Activities

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

[Guide] Section 5.3.3.2 “Key Generation” states that for TLS communication, the ciphersuite is restricted to AES-256-GCM and no other configuration is necessary.

2.2.4.3 Test Activities

Performed in accordance with NIAP Policy Letter #5.

Section 5.2.1 of [ST] (“Cryptographic Operation”), Table 7 (“Cryptographic Functions Implemented by CSL Direct v2.0.0”) identifies the CAVP certifications verifying AES encryption and decryption, as follows.

Algorithm	**Tested Capabilities**	**Certificates**
AES as specified in ISO 18033-3, GCM as specified in ISO 19772 (256-bit)	Direction: Decrypt, Encrypt IV Generation: External Key Length: 256	A1515 AES-GCM

2.2.5 Cryptographic Operation (Signature Generation and Verification) (FCS_COP.1/SigGen)

2.2.5.1 TSS Activities

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

Section 5.2.1 of [ST] ("Cryptographic Operation") states the TOE provides cryptographic signature services using RSA with key sizes of 2048/4096 bits, and ECDSA NIST curves P-256, P-384, and P-521 with key sizes of 256, 384, or 521 bits.

2.2.5.2 Guidance Activities

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

[Guide] Section 5.3.3.2 "Key Generation" refers to the list of supported RSA and ECDSA key sizes in Table 6 (Section 5.2.1.1.1) that the TOE supports for cryptographic signature services. The list is the same as that in the TSS. Section 5.2.1.1.1 contains additional information and an example configuration. Section 1.2 indicates that the Black Lantern firmware automatically enables FIPS mode which ensures all other settings outside of those specifically described in the guide necessary for CC conformance are set by default. There is no other specific configuration specified as necessary for the TOE to use the signature services.

2.2.5.3 Test Activities

Performed in accordance with NIAP Policy Letter #5.

Section 5.2.1 of [ST] ("Cryptographic Operation"), Table 7 ("Cryptographic Functions Implemented by CSL Direct v2.0.0") identifies the CAVP certifications verifying digital signature services, as follows.

Algorithm	Tested Capabilities	Certificates
RSA schemes using cryptographic key sizes of 2048-bit or 4096 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5.5	**RSA Signature Generation (FIPS186-4)** Signature Type: PKCS 1.5 Modulo: 2048 Hash Algorithm: SHA2-256 Hash Algorithm: SHA2-384 Hash Algorithm: SHA2-512 Modulo: 4096 Hash Algorithm: SHA2-256 Hash Algorithm: SHA2-384 Hash Algorithm: SHA2-512	A1515 RSA SigGen (FIPS 186-4)

Algorithm	Tested Capabilities	Certificates
	RSA Signature Verification (FIPS186-4) Signature Type: PKCS 1.5 Modulo: 2048 Hash Algorithm: SHA2-256 Hash Algorithm: SHA2-384 Hash Algorithm: SHA2-512 Signature Type: PKCPSS Modulo: 2048 Hash: SHA2-256; Salt Length: 10 Hash: SHA2-384; Salt Length: 10 Hash: SHA2-512; Salt Length: 10	A1515 RSA SigVer (FIPS 186-4)
ECDSA schemes using "NIST curves" P-256, P-384 and P-521 that meet the following: FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5	**ECDSA Signature Generation (FIPS186-4)** Curve: P-256, P-384, P-521 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512 **ECDSA Signature Verification (FIPS186-4** Curve: P-256, P-384, P-521 Hash Algorithm: SHA2-256, SHA2-384, SHA2-512	A1515 ECDSA SigGen (FIPS 186-4) A1515 ECDSA SigVer (FIPS 186-4)

2.2.6 Cryptographic Operation (Hash Algorithm) (FCS_COP.1/Hash)

2.2.6.1 TSS Activities

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

Section 5.2.1 of [ST] ("Cryptographic Operations") states the TOE uses SHA hashing in conjunction with the following cryptographic operations: for integrity as part of HMAC-SHA operations within TLS/ PBKDF2; during digital signature calculation (hashing of the message); and also for authentication of NTP servers. Specifically, The TOE uses the SHA hash algorithms as follows:

- as part of the HMAC algorithm that provides data integrity for TLS (SHA-384)
- as part of the HMAC algorithm that supports Password-Based Key Derivation Function 2 (PBKDF2) used when storing authentication credentials (SHA-256)
- as part of RSA and ECDSA digital signature generation and verification (SHA-256, SHA-384, SHA-512)
- for NTP authentication (SHA-1, SHA-256, SHA-384, SHA-512).

2.2.6.2 Guidance Activities

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

[Guide] Section 5.3.3.5 "Hash Algorithms" states Hash algorithm configuration is available when configuring authenticated NTP and provides configuration instructions in Section 7.5.1. Section 5.3.3.5 states that to establish a TLS communication channel, the hashing function is specified by cryptographic criteria, and that the TOE supports SHA-256, SHA-384, or SHA-512. There is no configuration necessary to restrict this.

Section 7.2.2 describes how the passwords are stored as salted SHA256 hashes using PBKDF2 and states that there is no configuration required. Section 7.5.1 describes how to configure the hash algorithms for NTP which can be SHA-1, SHA-256, SHA-384, or SHA-512.

2.2.6.3 Test Activities

Performed in accordance with NIAP Policy Letter #5.

Section 5.2.1 of [ST] ("Cryptographic Operation"), Table 7 ("Cryptographic Functions Implemented by CSL Direct v2.0.0") identifies the CAVP certifications verifying cryptographic hashing, as follows.

Algorithm	Tested Capabilities	Certificates
SHS as defined in ISO/IEC 10118-3:2004	SHA-1 (digest size 160 bits) SHA-256 (digest size 256 bits) SHA-384 (digest size 384 bits) SHA-512 (digest size 512 bits)	A1515 SHA-1, SHA2-256, SHA2-384, SHA2-512

2.2.7 Cryptographic Operation (Keyed Hash Algorithm) (FCS_COP.1/KeyedHash)

2.2.7.1 TSS Activities

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

Section 5.2.1 of [ST] ("Cryptographic Operations") states the HMAC function implemented by the TOE uses key lengths, hash function, block size, and output MAC length as summarized in the following table:

Hash Function	Key Length	Block Size	Output MAC Length
SHA-256	16-256 bits in 8 bit increments	512 bits	256 bits
SHA-384	16-512 bits in 8 bit increments	1024 bits	384 bits

The TOE uses HMAC-SHA-384 in support of TLS and HMAC-SHA-256 in support of PBKDF.

2.2.7.2 Guidance Activities

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

[Guide] Section 5.3.3.6 “Keyed Hash Algorithm” states that no configuration is required to ensure HMAC functionality. Section 7.2.2 states that no additional configuration is needed for the PBKDF2 function with HMAC-SHA-256 function.

2.2.7.3 Test Activities

Performed in accordance with NIAP Policy Letter #5.

Section 5.2.1 of [ST] (“Cryptographic Operation”), Table 7 (“Cryptographic Functions Implemented by CSL Direct v2.0.0”) identifies the CAVP certifications verifying cryptographic keyed hashing, as follows.

Algorithm	Tested Capabilities	Certificates
HMAC that meets ISO/IEC 9797-2:2011, Section 7 “MAC Algorithm 2”	HMAC-SHA2-256 MAC: 256 Key Length: 16-256 Increment 8, digest size 256 bits HMAC-SHA2-384 MAC: 384 Key Length: 16-512 Increment 8, digest size 384 bits	A1515 HMAC-SHA2-256 HMAC-SHA2-384

2.2.8 HTTPS Protocol (FCS_HTTPS_EXT.1)

2.2.8.1 TSS Activities

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

Section 5.2.5.3 of [ST] (“HTTPS”) states the TOE implements HTTPS according to RFC 2818 by using a TLS v1.2 session to secure the HTTP connection.

2.2.8.2 Guidance Activities

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

[Guide] Section 3.2.2 states that by default the HTTPS <Server> RESTful API remote management communication is protected by using TLS encryption. To enable TLS to function properly, certificates will need to be created and installed in the Black Lantern and the corresponding remote host machines. Instructions for configuring certificates for the management connection is provided in Section 5.2.

2.2.8.3 Test Activities

This test is now performed as part of FIA_X509_EXT.1/Rev testing. Tests are performed in conjunction with the TLS evaluation activities. If the TOE is an HTTPS client or an HTTPS server utilizing X.509 client authentication, then the certificate validity shall be tested in accordance with testing performed for FIA_X509_EXT.1.

2.2.9 NTP Protocol (FCS_NTP_EXT.1)

2.2.9.1 TSS Activities

The evaluator shall examine the TSS to ensure it identifies the version of NTP supported, how it is implemented and what approach the TOE uses to ensure the timestamp it receives from an NTP timeserver (or NTP peer) is from an authenticated source and the integrity of the time has been maintained.

The TOE must support at least one of the methods or may use multiple methods, as specified in the SFR element 1.2. The evaluator shall ensure that each method selected in the ST is described in the TSS, including the version of NTP supported in element 1.1, the message digest algorithms used to verify the authenticity of the timestamp and/or the protocols used to ensure integrity of the timestamp.

Section 5.2.5.4 of [ST] ("NTP Protocol") states the TOE supports NTP v4. It can use SHA-1, SHA-256, SHA-384, or SHA-512 as its means of ensuring the timestamps it receives are from an authenticated source and their integrity has been maintained. This description is consistent with the specification of FCS_NTP_EXT.1 and the selections made in the ST.

2.2.9.2 Guidance Activities

FCS_NTP_EXT.1.1

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

[Guide] Section 7.5.1 "NTP Settings" provides instructions to configure NTP using the `setconfig` command. The description includes configuring up to 10 NTP Servers and configuring the authentication NTP parameters (SHA-1, SHA-256, SHA-384, or SHA-512 as indicated in Table 10). NTP v4 is used by default and no configuration is necessary.

FCS_NTP_EXT.1.2

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

Assurance Activity Note:

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

[Guide] Section 7.5.1 "NTP Settings" provides instructions to the Security Administrator as to how to configure the authentication method consistent with the selections in the ST. SHA-1, SHA-256, SHA-384, or SHA-512 are identified as the available parameters in Table 10.

FCS_NTP_EXT.1.3

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

[Guide] Section 7.5.1 “NTP Settings” states the TOE does not update its real-time clock based on timestamps received from broadcast or multicast addresses and no additional configuration is required for this behavior.

2.2.9.3 Test Activities

FCS_NTP_EXT.1.1

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

The evaluator configured the TOE to use NTP against a series of server that support each of the defined values. The evaluator ensured that the TOE could correctly receive NTP time updates using each of the defined NTP versions.

FCS_NTP_EXT.1.2

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

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

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

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

The evaluator configured a series of NTP servers to use one of the supported authentication algorithms specified by the ST. the evaluator configured the TOE to connect to one of the servers using one of the specified authentication algorithms and a valid key. The evaluator iterated the utilized key if more selections were made than NTP servers available.

FCS_NTP_EXT.1.3

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

The evaluator configured a NTP server to send broadcast and multicast time updates to the network. The evaluator observed that the TOE did not update its system time based on these updates received.

Modified in accordance with TD0528.

FCS_NTP_EXT.1.4

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

The evaluator observed that the TOE could be configured with at least 3 NTP servers.

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

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

The evaluator directed unsolicited NTP server records at the TOE and observed that the TOE did not act upon any of the received NTP records, and the time stayed in synch with the configured servers.

2.2.10 Cryptographic Operation (Random Bit Generation) (FCS_RBG_EXT.1)

2.2.10.1 TSS Activities

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

Section 5.2.2 of [ST] ("Random Bit Generation") states the TOE implements a NIST-approved AES-CTR Deterministic Random Bit Generator (DRBG), as specified in ISO/IEC 18031:2011 Table C.1 "Security Strength Table for Hash Functions". The implementation uses one platform-based entropy source, which accumulates entropy from one hardware-based noise source, which has a minimum 256 bits of entropy.

2.2.10.2 Guidance Activities

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

[Guide] Section 5.3.3.1 “Random Number Generation” states no administrator configuration is required for the RNG functionality.

2.2.10.3 Test Activities

Performed in accordance with NIAP Policy Letter #5.

Section 5.2.1 of [ST] (“Cryptographic Operations”), Table 7 (“Cryptographic Functions Implemented by CSL Direct v2.0.0”) identifies the CAVP certification verifying deterministic random bit generation, as follows.

Algorithm	Tested Capabilities	Certificates
CTR_DRBG in accordance with ISO/IEC 18031:2011	Counter DRBG Mode: AES-256	A1515 Counter DRBG

2.2.11 TLS Client Protocol without Mutual Authentication (FCS_TLSC_EXT.1)

2.2.11.1 TSS Activities

FCS_TLSC_EXT.1.1

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

Section 5.2.5.1 of [ST] (“TLS Client Protocol”) states the TOE’s TLS client implementation supports the following ciphersuites:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289.

This list is identical to the ciphersuites selected in the SFR.

FCS_TLSC_EXT.1.2

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

Section 5.2.5.1 of [ST] states the TOE’s TLS client implementation supports reference identifiers per RFC 6125 section 6, and also supports IPv4 addresses in the CN or SAN field. The TOE supports the use of wildcards in each of these cases.

The Security Administrator configures reference identifiers for the external audit server by configuring the applicable server hostname or IPv4 address parameters. The TOE supports certificate pinning by allowing the Security Administrator to import Intermediate and Root Certificate Authority (CA) certificates and associating them with a predefined host.

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

The TOE is not distributed. Therefore, this activity is not applicable.

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

Section 5.2.5.1 of [ST] states the TOE enforces canonical format for IPv4 addresses in accordance with RFC 3986. When comparing an IPv4 address in the CN field to the reference identifier, the TOE converts the text representation of the IP address to a binary representation in network byte order.

FCS_TLSC_EXT.1.4

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

Section 5.2.5.1 of [ST] states the TOE presents the Supported Elliptic Curves Extension in its Client Hello message. By default, the Supported Elliptic Curves Extension specifies the following NIST curves: secp256r1; secp384r1; and secp521r1.

2.2.11.2 Guidance Activities

FCS_TLSC_EXT.1.1

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

[Guide] Section 5.3.1 "Client Mode Configuration" indicates that the TOE uses only the Transport Security Layer (TLS) protocol, version 1.2, with mutual authentication. Section 5.3.1.1 describes only certificate configuration as necessary for the TOE's TLS Client to communicate with a Remote Logging Server. Section 5.3.3.3.1 states that the TOE is designed to support only two ciphersuites. These ciphersuites are the same two specified in [ST]. Therefore, there is no configuration necessary to ensure that TLS conforms to the description of FCS_TLSC_EXT.1.1 in the TSS.

FCS_TLSC_EXT.1.2

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

[Guide] Section 5.3.1.1 “Trusted Communication Channel with a Remote Logging Server” states the TOE uses reference identifiers per RFC 6125 section 6 and IPv4 addresses in the certificate's Subject Alternative Name (SAN) as the key for certificate lookup when the SAN field is present. If the SAN field is not present, the Black Lantern uses the Common Name (CN) as the key for certificate lookup. The SAN (or CN) is the hostname of the remote machine where it hosts the logging server. The Black Lantern associates each remote machine with a hostname. To associate the CA chain with a remotehost, the Security Administrator must import it into the Black Lantern as per instructions provided in 5.2.1.2. The instructions indicate to specify a <hostname> identifier (certificate CN or SAN) for the CA chain of the remotehost.

The [Guide] provides a set of warnings and/or CA policy recommendations that would result in secure TOE use. The evaluator found that Section 5.3.1.1 states that: for the logging server, the TOE validates the certificate and certificate chain to the certificate of a trusted known Root Certificate Authority (CA). Therefore, the Black Lantern must have the CA chain that the logging server's certificate is linked against. The instructions include how to check for the CA chain and if not present how to import it into the TOE using the import command. The description indicates that the hostname/ipaddress is identified in the certificate.

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

The TOE is not distributed.

FCS_TLSC_EXT.1.4

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

Section 5.2.5.1 of [ST] states the TOE supports the Elliptic Curves Extension (specifying only P-256, P-384, and P-521) in its Client Hello, and these are supported by default (no configuration required).

2.2.11.3 Test Activities

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

FCS_TLSC_EXT.1.1

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

For each of the selected ciphersuites the evaluator configured a TLS server to only accept that specific ciphersuite. The evaluator then configured the TOE to attempt a connection to the TLS server and observed that the connection could be completed successfully and the Application Data record was not sent in plaintext.

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.

The evaluator configured a TLS server to present a certificate containing only the ClientAuthentication EKU and observed that the connection attempt failed when the TOE attempted the connection. The evaluator configured a TLS server to present a certificate containing only the ServerAuthentication EKU and observed that the attempt succeeded when the TOE attempted the connection.

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

The evaluator configured a proprietary tool to present an ECDSA ciphersuite while presenting an RSA certificate. The evaluator caused the TOE to attempt a connection and observed that the attempt failed.

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

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

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

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

The evaluator configured a proprietary tool to forcefully present the ciphersuite TLS_NULL_WITH_NULL_NULL and observed that the TOE rejected the connection.

The evaluator configured a proprietary tool to forcefully present a ciphersuite that was not in the ClientHello and observed that the TOE rejected the connection.

The evaluator configured a proprietary tool to modify the server key exchange to use a non-supported group and observed that the TOE rejected the connection.

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

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

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

The evaluator configured a proprietary TLS server to forcefully present a TLS version that is not supported by the TOE and observed that the TOE rejected the connection attempt.

The evaluator configured a proprietary TLS server to modify a byte in the signature block of the Server Key Exchange record. The evaluator observed that the TOE rejected the connection.

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

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

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

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

The evaluator configured a proprietary TLS server to modify a byte in the ServerFinished record. The evaluator observed that the TOE rejected the ServerFinished record.

The evaluator configured a proprietary TLS server to send a record that was garbled instead of the ServerFinished record and observed that the TOE rejected the ServerFinished record.

The evaluator configured a proprietary TLS server to modify the nonce value locally after sending the record to the TOE. The evaluator observed that the TOE rejected the connection attempt.

FCS_TLSC_EXT.1.2

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

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

or

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

or

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

Note that for some tests additional conditions apply.

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

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

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

Test 1 [conditional]: The evaluator shall present a server certificate that contains a CN that does not match the reference identifier and does not contain the SAN extension. The evaluator shall verify that the connection fails. The evaluator shall repeat this test for each identifier type (e.g. IPv4, IPv6, FQDN) supported in the CN. When testing IPv4 or IPv6 addresses, the evaluator shall modify a single decimal or hexadecimal digit in the CN. Remark: Some systems might require the presence of the SAN extension. In this case the connection would still fail but for the reason of the missing SAN extension instead of the mismatch of CN and reference identifier. Both reasons are acceptable to pass Test 1.

For each supported CN type, the evaluator presented a certificate without the SAN field and a CN that was not valid for the TLS server. The evaluator caused the TOE to attempt a connection and observed that the TOE rejected the connection.

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

For each supported CN type, the evaluator presented a certificate with a SAN that is not valid for the TLS server and a CN that is valid for the TLS server. The evaluator caused the TOE to attempt a connection and observed that the TOE rejected the connection.

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

For each supported CN type, the evaluator presented a certificate without a SAN and a CN that is valid for the TLS server. The evaluator caused the TOE to attempt a connection and observed that the TOE accepted the connection.

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

For each supported CN type, the evaluator presented a certificate with a SAN that is valid for the TLS server and a CN that is not valid for the TLS server. The evaluator caused the TOE to attempt a connection and observed that the TOE accepted the connection.

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

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

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

Each of the following tests was performed for both CN and SAN DNS reference identifiers.

The evaluator configured a TLS server to present a certificate that had a wildcard not in the leftmost position and caused the TOE to attempt a connection with a reference ID that would fit the criteria of the certificate. The evaluator observed that the TOE rejected the connection.

The evaluator configured the reference ID on the TOE with a single leftmost id and caused the TLS server to present a certificate with a wildcard in the leftmost position. The evaluator observed that the connection succeeded.

The evaluator then configured the reference ID on the TOE with two leftmost ID’s and caused the TLS server to present a certificate with a wildcard in the leftmost position. The evaluator observed that the connection failed.

The evaluator then configured the reference ID on the TOE with no leftmost ID’s and caused the TLS server to present a certificate with a wildcard in the leftmost position. The evaluator observed that the connection failed.

Modified by TD0634:

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

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

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

Test 6: [conditional] If IP address identifiers supported in the SAN or CN, the evaluator shall present a server certificate that contains a CN that matches the reference identifier, except one of the groups has been replaced with a wildcard asterisk (*) (e.g. CN=*.168.0.1 when connecting to 192.168.0.1...

This negative test corresponds to the following section of the Application Note 64/105: "The exception being, the use of wildcards is not supported when using IP address as the reference identifier."

The evaluator configured the TOE to have a reference ID that is an IP address. The evaluator configured the TLS server to present a certificate with a wildcard present in the CN field that is otherwise valid for the TLS server. The evaluator caused the TOE to attempt a connection and observed that the TOE rejected the connection.

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

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

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

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

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

N/A, The TOE does not claim FPT_ITT.

FCS_TLSC_EXT.1.3

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

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

The evaluator configured the TOE to have the Root CA installed and then configured the TLS server to present the requisite Intermediate CA certificates along with the End-Entity Certificate. The evaluator observed that the TOE accepted the connection.

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

For each Certificate failure that could be contrived the evaluator presented certificates that would meet the requirements for the failure and observed that the TOE rejected the certificate in each case.

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

N/A, The TOE does not claim any administrative overrides.

FCS_TLSC_EXT.1.4

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

For each supported ECDHE/DHE value, the evaluator configured a TLS server to perform one of the supported values and observed that the TOE was able to complete a connection using the specified Key Exchange method.

2.2.12 TLS Client Support for Mutual Authentication (FCS_TLSC_EXT.2)

2.2.12.1 TSS Activities

FCS_TLSC_EXT.2.1

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

Section 5.2.5.1 of [ST] states that the TOE's TLS client implementation supports TLS communication with mutual authentication using X.509v3 certificates. Section 5.3.4 indicates that all certificates are imported manually into the TOE. The TOE's localhost certificate is used for all PKI-based communication.

2.2.12.2 Guidance Activities

FCS_TLSC_EXT.2.1

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

[Guide] Section 5.2 describes steps required for configuration of the localhost certificate used for identity of the Black Lantern TOE in mutual authentication with the audit server. Note that configuration of the server-side certificates for the audit server are described in Section 5.3.

2.2.12.3 Test Activities

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

FCS_TLSC_EXT.2.1

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

2.2.13 TLS Server Protocol without Mutual Authentication (FCS_TLSS_EXT.1)

2.2.13.1 TSS Activities

FCS_TLSS_EXT.1.1

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

Section 5.2.5.2 of [ST] ("TLS Server Protocol") states the TOE's TLS server implementation supports TLS v1.2 only and the following TLS cipher suites when acting as a TLS server:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289.

This list is identical to the set of ciphersuites specified in FCS_TLSS_EXT.1.1.

FCS_TLSS_EXT.1.2

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

Section 5.2.5.2 of [ST] states the TOE supports TLS 1.2 only and rejects ClientHello messages that do not specify support TLS v1.2.

FCS_TLSS_EXT.1.3

Modified by TD0635

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

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

Section 5.2.5.2 "TLS Server Protocol" of [ST] states that the TOE can perform key establishment using the secp256r1, secp384r1, and secp521r1 ECDHE curves.

FCS_TLSS_EXT.1.4

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

Section 5.2.5.2 of [ST] states the TOE does not support session resumption or session tickets.

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

The TOE's TLS implementation does not support session resumption or session tickets.

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

The TOE's TLS implementation does not support session resumption or session tickets.

Added in accordance with TD0569.

If the TOE claims a TLS server capable of session resumption (as a single context, or across multiple contexts), the evaluator verifies that the TSS describes how session resumption operates (i.e. what would trigger a full handshake, e.g. checking session status, checking Session ID, etc.). If multiple contexts are used the TSS describes how session resumption is coordinated across those contexts. In case session establishment and session resumption are always using a separate context, the TSS shall describe how the contexts interact with respect to session resumption (in particular regarding the session ID). It is acceptable for sessions established in one context to be resumable in another context.

Section 5.2.5.2 of [ST] states the TOE does not support session resumption.

2.2.13.2 Guidance Activities

FCS_TLSS_EXT.1.1

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

[Guide] Section 3.2.2 states that the HTTPS RESTful API management communication is by default protected by using TLS. Section 5.2.1 describes the initial configuration required for the RESTful interface which only includes the establishment of server certificates, and private key/CSR generation. Section 5.3.2 "Server Mode Configuration" states that the TOE only supports TLSv1.2, therefore there is no configuration specified as necessary for the TLS version. There is no configuration required to restrict ciphersuites advertised by the TOE, two are supported by default as stated in Section 5.3.3.3.1.

FCS_TLSS_EXT.1.2

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

[Guide] Section 5.3.2 "Server Mode Configuration" states the TOE supports TLS 1.2 server protocol with mutual authentication and rejects client connection attempts that use SSL 2.0, SSL 3.0, TLS 1.0, and TLS 1.1. There is no configuration necessary to ensure this.

FCS_TLSS_EXT.1.3

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

[Guide] Section 5.3.3.3.1 "Elliptic Curve-based Key Establishment" provides instructions to ensure the use of Elliptic Curve-based key establishment for TLS communication. The instructions re-iterate the need to load appropriate certificates (as was described earlier).

Added in accordance with TD0569.

FCS_TLSS_EXT.1.4

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

[Guide] Section 5.3 "Secure Communication" states that for both TLS client and server modes, session resumption and session tickets are not supported, and no additional configuration is necessary to ensure this.

2.2.13.3 Test Activities

FCS_TLSS_EXT.1.1

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

For each selected ciphersuite, the evaluator attempted to open a connection to the TOE offering only that ciphersuite. The evaluator verified in each case that the TOE accepted the connection.

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.

The evaluator attempted to open a TLS connection to the TOE offering a cipher that is not supported by the TOE and verified that the TOE rejected the connection attempt.

The evaluator attempted to open a TLS connection to the TOE offering only the ciphersuite TLS_NULL_WITH_NULL_NULL and observed that the TOE rejected the connection attempt.

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

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

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

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

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

The evaluator configured a proprietary TLS client to modify the client finished record prior to sending it to the TOE. The evaluator observed that the TOE rejected the connection attempt.

The evaluator observed that the TOE was capable of properly and actually encrypting the TLS Finished and Application Data records.

FCS_TLSS_EXT.1.2

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

The evaluator attempted a connection sequentially only offering one of the non-supported TLS versions. The evaluator verified that in each case the TOE rejected the connection attempt.

FCS_TLSS_EXT.1.3

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

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

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

The evaluator attempted a ECDHE ciphersuite connection and observed that the TOE was able to complete the connection when the presented elliptic curve was one of the values supported by the TOE and rejected the connection when the presented elliptic curve was not one of the values supported by the TOE.

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

N/A, the TOE does not support any DHE ciphersuites.

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

N/A, the TOE does not support any plain RSA ciphersuites.

Modified in accordance with TD0569.

FCS_TLSS_EXT.1.4

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

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

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

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

c) The client verifies the Server Hello message contains a zero-length session identifier or passes the following steps:

Note: The following steps are only performed if the ServerHello message contains a non-zero length SessionID.

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

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

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

Remark: If multiple contexts are supported for session resumption, the session ID or session ticket may be obtained in one context for resumption in another context. It is possible that one or more contexts may only permit the construction of sessions to be reused in other contexts but not actually permit resumption themselves. For contexts which do not permit resumption, the evaluator is required to verify this behaviour subject to the description provided in the TSS. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

The evaluator attempted a connection to the TOE where the client would support SessionTickets and observed that the TOE did not perform SessionTickets. The evaluator then attempted to resume a session with the SessionID of a valid session and observed that the TOE did not perform resumption based on SessionID and observed that a new SessionID was provided for the second connection attempt.

Modified in accordance with TD0569.

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

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

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

Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ID may be obtained in one context for resumption in another context. There is no requirement that the session ID be obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ID constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

N/A, the TOE does not support session resumption.

Modified in accordance with TD0556 and TD0569.

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

a) The evaluator shall permit a successful TLS handshake to occur in which a session ticket is exchanged with the non-TOE client. The evaluator shall then attempt to correctly reuse the previous session by sending the session ticket in the ClientHello. The evaluator shall confirm that the TOE responds with an abbreviated handshake described in section 3.1 of RFC 5077 and illustrated with an example in figure 2. Of particular note: if the server successfully verifies the client's ticket, then it may renew the ticket by including a NewSessionTicket handshake message after the ServerHello in the abbreviated handshake (which is shown in figure 2). This is not required, however as further clarified in section 3.3 of RFC 5077.

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

Remark: If multiple contexts are supported for session resumption, for each of the above test cases, the session ticket may be obtained in one context for resumption in another context. There is no requirement that the session ticket be obtained and replayed within the same context subject to the description provided in the TSS. All contexts that can reuse a session ticket constructed in another context must be tested. It is not mandated that the session establishment and session resumption share context. For example, it is acceptable for a control channel to establish and application channel to resume the session.

N/A, the TOE does not support session resumption.

2.2.14 TLS Server Support for Mutual Authentication (FCS_TLSS_EXT.2)

2.2.14.1 TSS Activities

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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

Section 5.2.5.2 of [ST] describes the TOE's TLS server implementation support for TLS communication with mutual authentication of TLS clients using X.509v3 certificates. If the client certificate is invalid, The TOE will not establish a TLS connection. The TOE does not support any fallback authentication functions and does not implement any administrative override. The expected identifier of a client is configured into the TOE as a remote client configuration parameter. The TOE supports Fully Qualified Domain name (FQDN) and IPv4 address identifiers. The TOE matches FQDN identifiers according to RFC 6125, whereas for IPv4 address identifiers, the TOE performs a bit-wise comparison. During a TLS connection attempt, the TOE compares the expected identifier with the identifier in the client certificate's Subject Alternative Name (SAN) field, if it is available; otherwise, it is compared to the Common Name (CN) field. If the expected identifier does not match the SAN or CN, then the connection is not established.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

The evaluator shall verify the TSS describes how the TSF uses certificates to authenticate the TLS client. The evaluator shall verify the TSS describes if the TSF supports any fallback authentication functions (e.g. username/password, challenge response) the TSF uses to authenticate TLS clients that do not present a certificate. If fallback authentication functions are supported, the evaluator shall verify the TSS describes whether the fallback authentication functions can be disabled.

Section 5.2.5.2 of [ST] states that the TOE does not support any fallback authentication functions and does not implement any administrative override. The expected identifier of a client is configured into the TOE as a remote client configuration parameter. The TOE supports Fully Qualified Domain name (FQDN) and IPv4 address identifiers. The TOE matches FQDN identifiers according to RFC 6125, whereas for IPv4 address identifiers, the TOE performs a bit-wise comparison. During a TLS connection attempt, the TOE compares the expected identifier with the identifier in the client certificate's Subject Alternative Name (SAN) field, if it is available; otherwise, it is compared to the Common Name (CN) field. If the expected identifier does not match the SAN or CN, then the connection is not established.

FCS_TLSS_EXT.2.3

The evaluator shall verify that the TSS describes which types of identifiers are supported during client authentication (e.g. Fully Qualified Domain Name (FQDN)). If FQDNs are supported, the evaluator shall verify that the TSS describes that corresponding identifiers are matched according to RFC6125. For all other types of identifiers, the evaluator shall verify that the TSS describes how these identifiers are parsed from the certificate, what the expected identifiers are and how the parsed identifiers from the certificate are matched against the expected identifiers.

Section 5.2.5.2 of [ST] states that the TOE supports Fully Qualified Domain name (FQDN) and IPv4 address identifiers The TOE matches FQDN identifiers according to RFC 6125, whereas for IPv4 address identifiers, the TOE performs a bit-wise comparison. During a TLS connection attempt, the TOE compares the expected identifier with the identifier in the client certificate's Subject Alternative Name

(SAN) field, if it is available; otherwise, it is compared to the Common Name (CN) field. If the expected identifier does not match the SAN or CN, then the connection is not established.

2.2.14.2 Guidance Activities

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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

[Guide] Section 5.3.2 states that the Black Lantern must have the client's CA chain installed and Section 5.2.1.2 “Importing Remote Host CA Chain” provides the instructions for importing the client CA chain. Hostname is certificate CN or SAN. The expected identifier of a client is configured into the TOE as a remote client configuration parameter as described in Section 5.3.2.1. The parameter values for the external management client are hostname/FQDN or IP address as specified in Section 5.3.2.1.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

The evaluator shall verify the guidance describes how to configure the TLS client certificate authentication function. If the TSF supports fallback authentication functions, the evaluator shall verify the guidance provides instructions for configuring the fallback authentication functions. If fallback authentication functions can be disabled, the evaluator shall verify the guidance provides instructions for disabling the fallback authentication functions.

[Guide] Section 5.3.2 Server Mode Configuration states that “The Black Lantern is, by default, configured to use TLS 1.2 with mutual authentication to provide confidentiality and data integrity for the trusted path between remote client and itself.” Once the client certificate (5.2.1.2 Import client cert chain) and the other configuration in section 5.3.2 is done, then there is no other configuration required to force the TLS client to authenticate. It is done by default.

Section 5.2.5.2 of [ST] states that the TOE does not support any fallback authentication functions and does not implement any administrative override.

FCS_TLSS_EXT.2.3

The evaluator shall ensure that the AGD guidance describes the configuration of expected identifier(s) for X.509 certificate-based authentication of TLS clients. The evaluator ensures this description includes all types of identifiers described in the TSS and, if claimed, configuration of the TOE to use a directory server.

[Guide] The expected identifier of a client is configured into the TOE as a remote client configuration parameter as described in Section 5.3.2.1. The parameter values for the external management client are hostname/FQDN or address. The TSS does not claim the use of a directory server.

2.2.14.3 Test Activities

For all tests in this chapter the TLS client used for testing of the TOE shall support mutual authentication.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 1a [conditional]: If the TOE requires or can be configured to require a client certificate, the evaluator shall configure the TOE to require a client certificate and send a Certificate Request to the client. The evaluator shall attempt a connection while sending a certificate_list structure with a length of zero in the Client Certificate message. The evaluator shall verify that the handshake is not finished successfully and no application data flows.

The evaluator configured the TOE to require Mutual Authentication and attempted to connect to the TOE and did not present a client certificate. The evaluator observed that the TOE did not accept the connection.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 1b [conditional]: If the TOE supports fallback authentication functions and these functions cannot be disabled. The evaluator shall configure the fallback authentication functions on the TOE and configure the TOE to send a Certificate Request to the client. The evaluator shall attempt a connection while sending a certificate_list structure with a length of zero in the Client Certificate message. The evaluator shall verify the TOE authenticates the connection using the fallback authentication functions as described in the TSS. 329 Note: Testing the validity of the client certificate is performed as part of X.509 testing.

N/A, The TOE does not support fallback authentication functions.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 2 [conditional]: If TLS 1.2 is claimed for the TOE, the evaluator shall configure the server to send a certificate request to the client without the supported_signature_algorithm used by the client's certificate. The evaluator shall attempt a connection using the client certificate and verify that the connection is denied.

The evaluator configured a client to present a certificate for ClientAuthentication that was signed by an algorithm that is not one of the supported signature algorithms. The evaluator observed that the TOE rejected the connection attempt.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 3: The aim of this test is to check the response of the server when it receives a client identity certificate that is signed by an impostor CA (either Root CA or intermediate CA). To carry out this test the evaluator shall configure the client to send a client identity certificate with an issuer field that identifies a CA recognised by the TOE as a trusted CA, but where the key used for the signature on the client certificate does not correspond to the CA certificate trusted by the TOE (meaning that the client certificate is invalid because its certification path does not terminate in the claimed CA certificate). The evaluator shall verify that the attempted connection is denied.

The evaluator configured a client to present a certificate that was signed by an imposter CA certificate and observed that the TOE rejected the connection attempt.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 4: The evaluator shall configure the client to send a certificate with the Client Authentication purpose in the extendedKeyUsage field and verify that the server accepts the attempted connection. The evaluator shall repeat this test without the Client Authentication purpose and shall verify that the server denies the connection. Ideally, the two certificates should be identical except for the Client Authentication purpose

The evaluator configured a client to present a certificate that only possessed the ClientAuthentication EKU and observed that the connection was accepted.

The evaluator configured a client to present a certificate that only possessed the ServerAuthentication EKU and observed that the connection was rejected.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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

a) Configure the server to require mutual authentication and then connect to the server with a client configured to send a client certificate that is signed by a Certificate Authority trusted by the TOE. The evaluator shall verify that the server accepts the connection.

b) Configure the server to require mutual authentication and then modify a byte in the signature block of the client's Certificate Verify handshake message (see RFC5246 Sec 7.4.8). The evaluator shall verify that the server rejects the connection.

Note: Testing the validity of the client certificate is performed as part of X.509 testing

The evaluator configured a client to present a certificate that is valid and chains correctly and observed that the TOE accepted the connection.

The evaluator configured a proprietary TLS client to present a valid certificate and modify a byte in the signature of the client's Certificate Verify record and observed that the TOE rejected the connection attempt.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

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

Test 6: Using the administrative guidance, the evaluator shall load a CA certificate or certificates needed to validate the presented certificate used to authenticate an external entity and demonstrate that the function succeeds, and a trusted channel can be established.

The evaluator configured a client to present a certificate that is valid and chains correctly and observed that the TOE accepted the connection.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 7: The evaluator shall then change the presented certificate(s) so that validation fails and show that the certificate is not automatically accepted. The evaluator shall repeat this test to cover the selected types of failure defined in the SFR (i.e. the selected ones from failed matching of the reference identifier, failed validation of the certificate path, failed validation of the expiration date, failed determination of the revocation status). The evaluator performs the action indicated in the SFR selection observing the TSF resulting in the expected state for the trusted channel (e.g. trusted channel was established) covering the types of failure for which an override mechanism is defined.

The evaluator configured a client to present a certificate containing incorrect CN values and verified that the TOE rejected the connection.

The evaluator configured a client to present a certificate that is valid and did not correctly provide the needed CA chains and observed that the TOE rejected the connection.

The evaluator configured a client to present a certificate that is expired and observed that the TOE rejected the connection.

The evaluator configured a client to present a certificate that possesses revocation checking and rendered the revocation responder unavailable. The evaluator observed that the TOE rejected the connection after attempting and failing to determine the revocation status.

The TOE does not implement any override mechanisms for certificate validation failures.

FCS_TLSS_EXT.2.1 and FCS_TLSS_EXT.2.2

Test 8 [conditional]: The purpose of this test is to verify that only selected certificate validation failures could be administratively overridden. If any override mechanism is defined for failed certificate validation, the evaluator shall configure a new presented certificate that does not contain a valid entry in one of the mandatory fields or parameters (e.g. inappropriate value in extendedKeyUsage field) but is otherwise valid and signed by a trusted CA. The evaluator shall confirm that the certificate validation fails (i.e. certificate is rejected), and there is no administrative override available to accept such certificate.

N/A, The TOE does not support administrative overrides.

FCS_TLSS_EXT.2.3

The evaluator shall send a client certificate with an identifier that does not match an expected identifier and verify that the server denies the connection.

The evaluator configured a client to present a certificate with an invalid identity and observed that the TOE rejected the connection attempt.

2.3 Identification and Authentication (FIA)

2.3.1 Authentication Failure Management (FIA_AFL.1)

2.3.1.1 TSS Activities

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

Section 5.3.1 of [ST] states that the TOE provides TOE a RESTful interface over an HTTPS channel for remote admins and they are authenticated by the TOEs local password-based mechanism. Section 5.3.2 of [ST] ("Authentication Failure Management") states that the TOE validates credentials in the HTTPS header of RESTful requests against a local user account and keeps a count of consecutive failed authentication attempts for each configured user. If the number of consecutive failed authentication attempts reaches the configured value for allowed failed attempts, the TOE disables the user account. The section also notes that although all users are subject to lockout after consecutive failed remote authentication attempts, users with the Security Admin role can never be locked out of the SCI. All other users are subject to lockout at both the local and remote interfaces.

A Security Admin can enable the account to restore administrator ability to log onto the TOE.

Section 5.3.2 says that a user in the Security Admin role can use the `setconfig` SCI command or a RESTful API PUT method on the config endpoint to configure the number of consecutive failed remote authentication attempts. This number can be any integer between 1 and 100 inclusive and has a default value of 5.

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

Section 5.3.2 of [ST] states that although all users are subject to lockout after consecutive failed remote authentication attempts, users with the Security Admin role can never be locked out of the SCI. All other users are subject to lockout at both the local and remote interfaces.

2.3.1.2 Guidance Activities

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

[Guide] Section 7.2.1 "Configuration" provides instructions for configuring the security.maxloginretriesbeforedisable setting using the `setconfig` command or RESTful API PUT method on the config endpoint as described in Section 3.2 Table 1 "Configure the authentication failure parameters" entry. This security configuration parameter specifies the number of allowed authentication or login failures before the TOE disables the user account. Table 10 identifies the number can be any integer between 1 and 100 inclusive and has a default value of 5. Section 7.2.1 also describes the moduser command used to re-enable a previously disabled administrator account and provides an example.

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

[Guide] Section 7.2.1 "Configuration" describes that the function does not apply to administrator accounts with Security role, logging into the local management SCI and so Administrator access is always maintained.

Section 5.1.1 requires a new Security Administrator be created, and afterwards, the pre-configured Initial Security Administrator account be disabled. The newly created Security Admin can never be locked out of the SCI.

2.3.1.3 Test Activities

The evaluator shall perform the following tests for each method by which remote administrators access the TOE (e.g. any passwords entered as part of establishing the connection protocol or the remote administrator application):

Test 1: The evaluator shall use the operational guidance to configure the number of successive unsuccessful authentication attempts allowed by the TOE (and, if the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall also use the operational guidance to configure the time period after which access is re-enabled). The evaluator shall test that once the authentication attempts limit is reached, authentication attempts with valid credentials are no longer successful.

The evaluator configured the maximum number of attempts prior to user lockout and observed that once the user was locked out the user could no longer authenticate to the TOE with valid credentials.

Test 2: After reaching the limit for unsuccessful authentication attempts as in Test 1 above, the evaluator shall proceed as follows.

If the administrator action selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall confirm by testing that following the operational guidance and performing each action specified in the ST to re-enable the remote administrator's access results in successful access (when using valid credentials for that administrator).

If the time period selection in FIA_AFL.1.2 is included in the ST, then the evaluator shall wait for just less than the time period configured in Test 1 and show that an authorisation attempt using valid credentials does not result in successful access. The evaluator shall then wait until just after the time period configured in Test 1 and show that an authorisation attempt using valid credentials results in successful access.

The TOE only claims administrator action and the evaluator observed that the TOE does not allow the locked-out user to log in until the administrator unlocked the account. Once the user was unlocked the evaluator was able to successfully log in with valid credentials.

2.3.2 Password Management (FIA_PMG_EXT.1)

2.3.2.1 TSS Activities

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

Section 5.3.3 of [ST] ("Password Management") states the TOE maintains a local database of user accounts with their corresponding passwords. Passwords can be composed of any combination of:

- Upper and lower case letters
- Numbers
- The following special characters: !@#$%^&*()_|<>?.~,.

The minimum length allowed for passwords is configurable in the range 8 to 32, with a default of 8. The maximum length supported for passwords is 32 characters

2.3.2.2 Guidance Activities

The evaluator shall examine the guidance documentation to determine that it:

a) identifies the characters that may be used in passwords and provides guidance to security administrators on the composition of strong passwords, and

b) provides instructions on setting the minimum password length and describes the valid minimum password lengths supported.

[Guide] Sections 3.2.3.2 “Password Policies” and 7.2.2 “Password Guidance” identifies the characters that may be used in passwords as “_!@#$%^&*()?<>.,~|”. Guidance on the secure composition of passwords is provided in Section 7.2.2.

Instructions on setting the minimum password length using the `setconfig` command is provided in Section 7.2.1 and the valid minimum password lengths supported are described in Table 10 (8 to 32 characters).

2.3.2.3 Test Activities

The evaluator shall perform the following tests.

Test 1: The evaluator shall compose passwords that meet the requirements in some way. For each password, the evaluator shall verify that the TOE supports the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that all characters, and a minimum length listed in the requirement are supported and justify the subset of those characters chosen for testing.

The evaluator verified that the TOE would allow passwords to be set that meet the length requirement and all defined values are permitted.

Test 2: The evaluator shall compose passwords that do not meet the requirements in some way. For each password, the evaluator shall verify that the TOE does not support the password. While the evaluator is not required (nor is it feasible) to test all possible compositions of passwords, the evaluator shall ensure that the TOE enforces the allowed characters and the minimum length listed in the requirement and justify the subset of those characters chosen for testing.

The evaluator verified that the TOE enforced the currently configured password length requirement and password that are not long enough are rejected. The evaluator verified that the TOE rejected passwords with values that are not defined as permitted.

2.3.3 Protected Authentication Feedback (FIA_UAU.7)

2.3.3.1 TSS Activities

None defined.

2.3.3.2 Guidance Activities

The evaluator shall examine the guidance documentation to determine that any necessary preparatory steps to ensure authentication data is not revealed while entering for each local login allowed.

Section 5.1.1 “Configuring Local Login” states that password data is obfuscated while it is being entered and only generic success/failure messages are provided. There are no preparatory steps required to ensure authentication data is not revealed while entering login information.

2.3.3.3 Test Activities

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

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

The evaluator observed that the TOE permits local logins with valid credentials and provides obscure feedback during the attempt.

2.3.4 Password-based Authentication Mechanism (FIA_UAU_EXT.2)

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.

2.3.5 User Identification and Authentication (FIA_UIA_EXT.1)

2.3.5.1 TSS Activities

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

Section 5.3.1 of [ST] ("User Identification and Authentication") describes the logon process for the TOEs local password-based mechanism to authenticate administrators attempting to access the TOE using the local serial console. In order to login at the local serial console, the administrator enters a username and password. The TOE checks the credentials against its local user database. Once the administrator is successfully identified and authenticated, the administrator will have access to the TOE Serial Console Interface (SCI), allowing the administrator to manage the TOE.

The local password-based mechanism is also used for remote administrative access to the TOE via a RESTful interface over an HTTPS channel. A user with the Security Admin role configures the TOE to authenticate remote administrators against its local user database. Each RESTful request (protected within the HTTPS channel) includes a username and password. The TOE checks these credentials against its local user database. If the TOE authenticates the user and the user possesses the necessary permissions, the TOE performs the management action included in the RESTful request. If the user is not authenticated (i.e., the provided password does not match the claimed user identity), or if the user does not have the necessary permission, the TOE rejects the RESTful request and generates an audit record. Section 5.1.1 of [ST] states the TOE audits all use of the identification and authentication mechanism, including the origin.

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

Section 5.3.1 of [ST] states that only services the TOE allows before user identification and authentication is display of the advisory notice and consent warning and to respond to ICMP echo request (ping) packets, if enabled to do so.

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

The TOE is not distributed. Therefore, this activity is not applicable.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.3.5.2 Guidance Activities

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

[Guide] Section 5.1.1 “Configuring Local Login” and Section 5.2 “Remote Login” describes the preparative steps for logging in to the TOE locally via SCI and remotely via the RESTful API.

ICMP echo requests (ping) are supported in the Black Lantern by default and is available prior to requiring identification or authentication for any non-TOE entity. If desired, the `setconfig` command can be used to disable it, as described in [Guide] Section 7.4.

2.3.5.3 Test Activities

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

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.

The evaluator verified that the TOE only allowed access to the system with valid credential combinations and if invalid combinations are provided the TOE does not allow access.

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.

The evaluator verified that only the configured services are available to a remote entity.

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.

The evaluator verified that only the specified services are available to a local user prior to authentication.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.3.6 X.509 Certificate Validation (FIA_X509_EXT.1/Rev)

2.3.6.1 TSS Activities

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

Section 5.3.4 of [ST] ("X.509 Certificates") states the TOE performs RFC 5280 certificate validation and certificate path validation on all X.509 certificates presented to it in support of secure communication over TLS with external audit servers and remote management using RESTful interface. The TOE supports a path length of at least three certificates. The TOE supports rules for extendedKeyUsage fields for TLS Server certificates (Sever Authentication purpose), TLS Client certificates (Client Authentication purpose), and OCSP certificates presented for OCSP responses (OCSP Signing purpose). The TOE does not use certificates for trusted updates or executable code integrity verification and therefore does not support rules for extendedKeyUsage fields of certificates used for trusted update or executable code integrity verification (Code Signing purpose).

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

Section 5.3.4 of [ST] states the TOE performs revocation checks on certificates presented to the TOE in support of secure communication over TLS with the external audit server and remote management using RESTful interface, using OCSP as specified in RFC 6960.

2.3.6.2 Guidance Activities

The evaluator shall also ensure that the guidance documentation describes where the check of validity of the certificates takes place, describes any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) and describes how certificate revocation checking is performed and on which certificate.

Section 5.3.1.1 states that the TOE validates the logging server certificate and certificate chain to the certificate of a trusted known Root Certificate Authority (CA) as part of establishing a TLS channel between the Black Lantern and the logging server.

Section 5.3.2 states that all certificates are verified at import time.

Section 5.2.1 and 5.3.2.1 provide details on the remote management host certificates that are validated during mutual authentication of TLS communications.

Section 5.2.1 identifies OCSP is used to support certificate status checking (e.g. revoked certificates) during a TLS connection with the logging server and the remote management host client.

Section 5.2.1 states Certificates are not used for trusted updates or executable code integrity. Therefore, the TOE does not support the rules for validating certificates with the Code Signing purpose in the extendedKeyUsage field, and this part of the requirement is trivially satisfied.

2.3.6.3 Test Activities

The evaluator shall demonstrate that checking the validity of a certificate is performed when a certificate is used in an authentication step or when performing trusted updates (if FPT_TUD_EXT.2 is selected). It is not sufficient to verify the status of a X.509 certificate only when it is loaded onto the TOE. It is not necessary to verify the revocation status of X.509 certificates during power-up self-tests (if the option for using X.509 certificates for self-testing is selected). The evaluator shall perform the following tests for FIA_X509_EXT.1/Rev. These tests must be repeated for each distinct security function that utilizes X.509v3 certificates. For example, if the TOE implements certificate-based authentication with IPSEC and TLS, then it shall be tested with each of these protocols:

Test 1a: The evaluator shall present the TOE with a valid chain of certificates (terminating in a trusted CA certificate) as needed to validate the leaf certificate to be used in the function and shall use this chain to demonstrate that the function succeeds. Test 1a shall be designed in a way that the chain can be 'broken' in Test 1b by either being able to remove the trust anchor from the TOEs trust store, or by setting up the trust store in a way that at least one intermediate CA certificate needs to be provided, together with the leaf certificate from outside the TOE, to complete the chain (e.g. by storing only the root CA certificate in the trust store).

The evaluator configured the TOE to have the Root CA installed and configured the TLS Peer to present the Intermediate Cas and End-Entity Certificate to make a valid chain and observed that the TOE successfully validated the Chain.

Test 1b: The evaluator shall then 'break' the chain used in Test 1a by either removing the trust anchor in the TOE's trust store used to terminate the chain, or by removing one of the intermediate CA certificates (provided together with the leaf certificate in Test 1a) to complete the chain. The evaluator shall show that an attempt to validate this broken chain fails.

The evaluator configured the TOE to have the Root CA installed and configured the TLS Peer to present the End-Entity Certificate and observed that the TOE did not validate the Chain and rejected connections.

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

The evaluator verified that when the TLS peer presented an expired certificate the TOE rejected the connection and the certificate.

Test 3: The evaluator shall test that the TOE can properly handle revoked certificates-–conditional on whether CRL or OCSP is selected; if both are selected, then a test shall be performed for each method. The evaluator shall test revocation of the peer certificate and revocation of the peer intermediate CA certificate i.e. the intermediate CA certificate should be revoked by the root CA. The evaluator shall ensure that a valid certificate is used, and that the validation function succeeds. The evaluator then attempts the test with a certificate that has been revoked (for each method chosen in the selection) to ensure when the certificate is no longer valid that the validation function fails. Revocation checking is only applied to certificates that are not designated as trust anchors. Therefore, the revoked certificate(s) used for testing shall not be a trust anchor.

The evaluator observed that when presented with certificate which possess OCSP revocation information the TOE was capable of querying the certificate status from the OCSP responder. The evaluator observed that the TOE revoked connections when the certificate presented was rejected or the chain possessed a revoked certificate.

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.

The evaluator observed that the TOE checked the response of the OCSP responder and if the OCSP responder did not use a certificate that possessed OCSP sign that the OCSP response was rejected.

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.)

The evaluator configured a proprietary tool to modify the first byte of a certificate to be presented to the TOE and observed that the TOE rejected the certificate when presented.

Test 6: The evaluator shall modify any byte in the certificate signatureValue field (see RFC5280 Sec. 4.1.1.3), which is normally the last field in the certificate, and demonstrate that the certificate fails to validate. (The signature on the certificate will not validate.)

The evaluator configured a proprietary tool to modify the last byte of a certificate to be presented to the TOE and observed that the TOE rejected the certificate when presented.

Test 7: The evaluator shall modify any byte in the public key of the certificate and demonstrate that the certificate fails to validate. (The hash of the certificate will not validate.)

The evaluator configured a proprietary tool to modify the public key of a certificate to be presented to the TOE and observed that the TOE rejected the certificate when presented.

Test added in accordance with TD0601.

Tests added by TD0527.

Test 8a: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall establish a valid, trusted certificate chain consisting of an EC leaf certificate, an EC Intermediate CA certificate not designated as a trust anchor, and an EC certificate designated as a trusted anchor, where the elliptic curve parameters are specified as a named curve. The evaluator shall confirm that the TOE validates the certificate chain.

Test 8b: (Conditional on support for EC certificates as indicated in FCS_COP.1/SigGen). The evaluator shall replace the intermediate certificate in the certificate chain for Test 8a with a modified certificate, where the modified intermediate CA has a public key information field where the EC parameters uses an explicit format version of the Elliptic Curve parameters in the public key information field of the intermediate CA certificate from Test 8a, and the modified Intermediate CA certificate is signed by the trusted EC root CA, but having no other changes. The evaluator shall confirm the TOE treats the certificate as invalid.

Test 8c: The evaluator shall establish a subordinate CA certificate, where the elliptic curve parameters are specified as a named curve, that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is accepted into the TOE's trust store. The evaluator shall then establish a subordinate CA certificate that uses an explicit format version of the elliptic curve parameters, and that is signed by a trusted EC root CA. The evaluator shall attempt to load the certificate into the trust store and observe that it is rejected, and not added to the TOE's trust store.

The evaluator configured the TOE to have a Root ECDSA CA certificate installed. The evaluator observed that the TOE permitted certificate chains where the Intermediate CA utilized named curve values and rejected chains where the Intermediate CA utilized explicit curve parameters. The evaluator attempted to load a CA chain into the TOE for validation where the Intermediate CA present used explicit curves for the public key information. The evaluator observed that the TOE rejected the certificate and did not allow it to be loaded into the TOE.

The evaluator shall perform the following tests for FIA_X509_EXT.1.2/Rev. The tests described must be performed in conjunction with the other certificate services assurance activities, including the functions in FIA_X509_EXT.2.1/Rev. The tests for the extendedKeyUsage rules are performed in conjunction with the uses that require those rules. Where the TSS identifies any of the rules for extendedKeyUsage fields (in FIA_X509_EXT.1.1) that are not supported by the TOE (i.e. where the ST is therefore claiming that they are trivially satisfied) then the associated extendedKeyUsage rule testing may be omitted.

The goal of the following tests is to verify that the TOE accepts a certificate as a CA certificate only if it has been marked as a CA certificate by using basicConstraints with the CA flag set to True (and implicitly tests that the TOE correctly parses the basicConstraints extension as part of X509v3 certificate chain validation).

For each of the following tests the evaluator shall create a chain of at least three certificates: a self-signed root CA certificate, an intermediate CA certificate and a leaf (node) certificate. The properties of the certificates in the chain are adjusted as described in each individual test below (and this modification shall be the only invalid aspect of the relevant certificate chain).

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).

The evaluator attempted to load a Certificate chain into the TOE where one of the Intermediate CAs did not contain the basicConstraints extension. The evaluator observed that the attempt failed, and the certificate was not imported.

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

The evaluator attempted to load a Certificate chain into the TOE where one of the Intermediate CAs did contain the basicConstraints extension, but it was set to false. The evaluator observed that the attempt failed, and the certificate was not imported.

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

The TOE only uses certificates for TLS client and server functionality and the TOE only uses one TLS implementation for both purposes.

2.3.7 X.509 Certificate Authentication (FIA_X509_EXT.2)

2.3.7.1 TSS Activities

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

Section 5.3.4 of [ST] ("X.509 Certificates") states the TOE uses X.509 certificates in support of secure communication over TLS with external audit servers (i.e. TLS) and remote management using RESTful interface (i.e. HTTPS). The TOE relies upon the administrator to manually import all certificates (root CA and any needed intermediate certificates).

[Guide] Section 5.2 and 5.3 describe the configuration and certificate import steps required.

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

Section 5.3.4 of [ST] states during revocation checking, if the OCSP server does not respond or if the certificate is invalid, the TOE does not establish the connection (i.e. the certificate is not accepted). There is no administrator configuration or override.

2.3.7.2 Guidance Activities

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

[Guide] Section 5.2.1.2 “Importing Remote Host CA Chain” describes how to import and use certificates to communicate with remote hosts in order to validate server certificates presented. Section 5.2.1 describes how the certificates must be imported so that the TOE has a trust relationship with the external certificates that are presented to it. Section 5.3.1 “Trusted Communication Channel with a Remote Logging Server” describes additional configuration required on the TOE in order to communicate with the external audit server. Section 5.3.2.1 “Trusted Communication Path with Remote Management Client” describes additional configuration required on the TOE in order to set the remote client and the RESTful API service port configurations.

[Guide] Section 5.2.1.2 “Importing Remote Host CA Chain” describes the configuration required on the TOE in order to validate server certificates presented by the external audit server and the remote management client.

[Guide] Section 5.2.1 “Configuring Remote Login” states that during revocation checking, if the OCSP server does not respond or if the certificate is invalid, the TOE does not establish the connection (i.e. the certificate is not accepted).

2.3.7.3 Test Activities

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

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

The evaluator verified that when using a certificate which possessed revocation status provider information, where the evaluator turned the revocation provider off, the TOE did not accept the certificate and the connection was not established.

2.3.8 Certificate Requests (FIA_X509_EXT.3 X.509)

2.3.8.1 TSS Activities

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

The ST selects “device-specific information” in FIA_X509_EXT.3.1 and [ST] Section 5.3.4 describes this information as DNS name and IP address that is stored in the Subject Alternative Name (SAN) list.

2.3.8.2 Guidance Activities

The evaluator shall check to ensure that the guidance documentation contains instructions on requesting certificates from a CA, including generation of a Certificate Request. If the ST author selects "Common Name", "Organization", "Organizational Unit", or "Country", the evaluator shall ensure that this guidance includes instructions for establishing these fields before creating the Certification Request.

[Guide] Section 5.2.1 “Importing Localhost Certificate and Certificate Authority (CA) Chain (as identity of the Black Lantern)” provides instructions on requesting certificates from a CA, including generation of a certificate request using the `gencsr` command. The guidance includes instructions and an example

request for establishing the Common Name, Organization, Organizational Unit, and Country fields as part of the `gencsr` command.

2.3.8.3 Test Activities

The evaluator shall perform the following tests:

Test 1: The evaluator shall use the guidance documentation to cause the TOE to generate a Certification Request. The evaluator shall capture the generated request and ensure that it conforms to the format specified. The evaluator shall confirm that the Certification Request provides the public key and other required information, including any necessary user-input information.

The evaluator caused the TOE to generate a CSR and observed that the TOE was capable of generating a Certificate request. The evaluator observed that the CSR contained the specified information.

Test 2: The evaluator shall demonstrate that validating a response message to a Certification Request without a valid certification path results in the function failing. The evaluator shall then load a certificate or certificates as trusted CAs needed to validate the certificate response message and demonstrate that the function succeeds.

The evaluator signed the response and provided the signed response to the TOE without provided the requisite CA certificates and observed that the TOE did not allow the certificate to be imported. The evaluator then attempted to install the signed response with the CA chain and observed that the TOE accepted the response, and the certificate was imported to the TOE.

2.4 Security Management (FMT)

General requirements for distributed TOEs.

TSS

For distributed TOEs it is required to verify the TSS to ensure that it describes how every function related to security management is realized for every TOE component and shared between different TOE components. The evaluator shall confirm that all relevant aspects of each TOE component are covered by the FMT SFRs.

Guidance Documentation

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

Tests

Tests defined to verify the correct implementation of security management functions shall be performed for every TOE component. For security management functions that are implemented centrally, sampling should be applied when defining the evaluator's tests (ensuring that all components are covered by the sample).

The TOE is not distributed.

2.4.1 FMT_MOF.1/Functions Management of Security Functions Behavior

2.4.1.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1.

For non-distributed TOEs, the evaluator shall ensure the TSS for each administrative function identified the TSS details how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE).

[ST] Section 5.4.3 details how the administrators manage how the TOE handles audit data (i.e., configure local log storage size) and the behavior of the TOE's audit functionality when local audit storage is full. SCI commands and RESTful APIs are identified.

2.4.1.2 Guidance Activities

For distributed TOEs see chapter 2.4.1.2.

For non-distributed TOEs, the evaluator shall also ensure the Guidance Documentation describes how the Security Administrator determines or modifies the behaviour of (whichever is supported by the TOE) transmitting audit data to an external IT entity, handling of audit data, audit functionality when Local Audit Storage Space is full (whichever is supported by the TOE) are performed to include required configuration settings.

[Guide] Section 6.2 describes configuring the local log storage size and the behavior of the TOE's audit functionality when local audit storage is full (overwrite oldest lo entries or drop newest entries) using the `setconfig` command.

2.4.1.3 Test Activities

Test 1 (if 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection):

The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt to modify the security related parameters can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

The evaluator verified that the behavior of the transmission of audit data to an external IT entity could not be modified by a non-administrator user.

Test 2 (if 'transmission of audit data to external IT entity' is selected from the second selection together with 'modify the behaviour of' in the first selection):

The evaluator shall try to modify all security related parameters for configuration of the transmission protocol for transmission of audit data to an external IT entity with prior authentication as Security Administrator. The effects of the modifications should be confirmed. The evaluator does not have to test all possible values of the security related parameters for configuration of the transmission

protocol for transmission of audit data to an external IT entity but at least one allowed value per parameter.

The evaluator verified that the behavior of the transmission of audit data to an external IT entity could be modified by an administrator user.

Test 1 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection):

The evaluator shall try to modify all security related parameters for configuration of the handling of audit data without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). Attempts to modify parameters without prior authentication should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace.

The evaluator verified that the behavior of the handling of audit data to an external IT entity could not be modified by a non-administrator user.

Test 2 (if 'handling of audit data' is selected from the second selection together with 'modify the behaviour of' in the first selection): The evaluator shall try to modify all security related parameters for configuration of the handling of audit data with prior authentication as Security Administrator. The effects of the modifications should be confirmed. The term 'handling of audit data' refers to the different options for selection and assignments in SFRs FAU_STG_EXT.1.2, FAU_STG_EXT.1.3 and FAU_STG_EXT.2/LocSpace. The evaluator does not necessarily have to test all possible values of the security related parameters for configuration of the handling of audit data but at least one allowed value per parameter.

The evaluator verified that the behavior of the handling of audit data to an external IT entity could be modified by an administrator user.

Test 1 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection):

The evaluator shall try to modify the behaviour when Local Audit Storage Space is full without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This attempt should fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

The evaluator verified that the behavior of the behavior when the local audit storage space is full could not be modified by a non-administrator user.

Test 2 (if 'audit functionality when Local Audit Storage Space is full' is selected from the second selection together with 'modify the behaviour of' in the first selection):

The evaluator shall try to modify the behaviour when Local Audit Storage Space is full with prior authentication as Security Administrator. This attempt should be successful. The effect of the change shall be verified. The evaluator does not necessarily have to test all possible values for the behaviour when Local Audit Storage Space is full but at least one change between allowed values for the behaviour.

The evaluator verified that the behavior of the behavior when the local audit storage space is full could be modified by an administrator user.

Test 3 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection):

The evaluator shall try to determine the behaviour of all options chosen from the second selection without prior authentication as Security Administrator (by authentication as a user with no administrator privileges or without user authentication at all). This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions without administrator authentication shall fail. According to the implementation no other users than the Security Administrator might be defined and without any user authentication the user might not be able to get to the point where the attempt can be executed. In that case it shall be demonstrated that access control mechanisms prevent execution up to the step that can be reached without authentication as Security Administrator.

The evaluator verified that the non-administrative user could not determine the audit behavior.

Test 4 (if in the first selection 'determine the behaviour of' has been chosen together with for any of the options in the second selection):

The evaluator shall try to determine the behaviour of all options chosen from the second selection with prior authentication as Security Administrator. This can be done in one test or in separate tests. The attempt(s) to determine the behaviour of the selected functions with Security Administrator authentication shall be successful.

The evaluator verified that the administrative user could determine the audit behavior.

2.4.2 Management of Security Functions Behavior (FMT_MOF.1/ManualUpdate)

2.4.2.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1 [of [SD-ND]]. There are no specific requirements for non-distributed TOEs.

The TOE is not distributed. Therefore, this activity is not applicable.

2.4.2.2 Guidance Activities

The evaluator shall examine the guidance documentation to determine that any necessary steps to perform manual update are described. The guidance documentation shall also provide warnings regarding functions that may cease to operate during the update (if applicable).

[Guide] Section 7.7 “Software Updates” describes the steps necessary to perform a manual update of the TOE firmware using the `updatebl` command. It also warns the administrator that during the firmware update and reboot, the Black Lantern will temporarily cease to operate until the update is completed.

For distributed TOEs the guidance documentation shall describe all steps how to update all TOE components. This shall contain description of the order in which components need to be updated if the order is relevant to the update process. The guidance documentation shall also provide warnings regarding functions of TOE components and the overall TOE that may cease to operate during the update (if applicable).

The TOE is not distributed. Therefore, this activity is not applicable.

2.4.2.3 Test Activities

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

The evaluator verified that a non-administrative user could not perform updates to the TOE.

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

2.4.3 Management of TSF Data (FMT_MTD.1/CoreData)

2.4.3.1 TSS Activities

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

Section 5.4.1 of [ST] (“Security Roles”) states the TOE implements two default administrator roles that together provide the capabilities of the Security Administrator role. Therefore, access to the management functions is enforced using a role-based access control (RBAC) method.

[ST] Section 5.3.1 states, “The TOE offers no services to external entities prior to identification and authentication, other than to display the advisory notice and consent warning message prior to completing the establishment of an interactive user session, and to respond to ICMP echo request (ping) packets, if enabled to do so. The TOE requires each administrative user to be successfully identified and authenticated before allowing any other TSF-mediated actions on behalf of that administrative user.”

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

Section 5.4.5 of [ST] states the TOE restricts the ability to manage TSF data to the Black Lantern Security Administrator and Network Administrator roles (i.e. using role-based access control methods) and states that this includes the ability to manage the TOE's trust store by uploading X.509 v3 certificates and CA certificates.

2.4.3.2 Guidance Activities

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

[Guide] Section 7.1.1 Table 9: Functions for Managing TSF Data identifies the TSF-data-manipulating functions implemented in response to the requirements of [NDcPP]. The TOE restricts access to these functions to users assigned to the 'Security Admin" and to the "Network Admin" roles defined by the TOE that corresponds to the Security Administrator role defined in [NDcPP].

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

[Guide] Section 7.1.1 states that the ability to manage the TOE's trust store by uploading X.509 v3 certificates and CA certificates is restricted to the Security Admin role. The available gencsr, genkey, import commands related to certificates are identified in Table 9. Section 5.2 provides information to the administrator to configure and maintain the TOE's trust store in a secure way, including commands to securely load CA chain and CA certificates using the import command and to designate a CA certificate as a trust anchor. When importing certificates, the Security Administrator must enter them in reverse order of signing, with the Root CA certificate inputted last.

2.4.3.3 Test Activities

No separate testing for FMT_MTD.1/CoreData is required unless one of the management functions has not already been exercised under any other SFR.

2.4.4 Management of TSF Data (FMT_MTD.1/CryptoKeys)

2.4.4.1 TSS Activities

For distributed TOEs see chapter 2.4.1.1 [of [SD-ND]].

The TOE is not distributed. Therefore, this activity is not applicable.

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

Section 5.4.1 of [ST] ("Management of TSF Data") states management of the cryptographic keys is restricted to the Black Lantern Security Administrator. Section 5.3.4 describes the Black Lantern Security Administrator is able to use the `gencsr` console command to generate a Certificate Signing Request

(CSR) that contain key pairs; import certificates along with the CA chain, covering certificates for the TOE generated by an external CA and certificates used to validate a presented TLS client or server certificate.

Section 5.4.4 of [ST] ("Management of TSF Data-Cryptographic Keys") refers to Table 10 in Section 5.2.4 which lists the keys and CSPs used by the TOE. The TOE manages all of these keys and CSPs automatically without administrator involvement, with the exception of the RSA private kay and the ECDSA private key. A user with the Security Admin role is able to perform the following operations on these keys:

- Generate an RSA or ECDSA key pair using the genkey SCI command
- Associate the generated RSA or ECDSA key pair with an X.509v3 certificate using the gencsr SCI command to generate a CSR (see Section 5.3.4 above for further details)
- Import the signed X.509v3 certificate associated with the generated RSA or ECDSA key pair, along with the certificate chain up to and including the signing root CA, using the import SCI command (see Section 5.3.4 above for further details)
- Delete the generated RSA or ECDSA keys using the rm SCI command to remove the files containing the keys from the file system.

2.4.4.2 Guidance Activities

For distributed TOEs see chapter 2.4.1.2 [of [SD-ND]].

The TOE is not distributed. Therefore, this activity is not applicable.

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

Guide Section 5.2.1 "" describes generating RSA and ECC key pairs and Certificate Signing Requests (CSRs); and importing certificates and certificate chains into the TOE's trust store. The "rm" command used to delete RSA or ECDSA keys by deleting the file or directory they reside within is described in Section 8.1 and an example is provided (search on "Remove contents in keys directory").

2.4.4.3 Test Activities

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

The evaluator verified that the crypto keys could not be controlled a non-administrative user and the user could not modify, delete, generate, or import keys to the TOE.

The evaluator shall try to perform at least one of the related actions with prior authentication as Security Administrator. This attempt should be successful.

The evaluator verified that the crypto keys could not controlled an administrative user and the user could modify, delete, generate, or import keys to the TOE.

2.4.5 Specification of Management Functions (FMT_SMF.1)

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

2.4.5.1 TSS Activities (containing also requirements on Guidance Documentation and Tests)

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

Section 5.4.2 of [ST] ("Specification of Management Functions") lists the management functions provided by the TOE, as specified in FMT_SMF.1. It states the TOE is administered via the SCI (local) and the RESTful (remote) interface, and identifies through which of these interfaces each management function can be accessed.

Section 3.2 "Management Guidance" of [Guide] lists the management functions provided by the TOE. The evaluator compared this list with the list of functions in FMT_SMF.1 and with the details provided in the TSS. The evaluator confirmed the guidance covers all the management functions specified in FMT_SMF.1 and described in the TSS.

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

Section 5.4.2 of [ST] states the TOE provides local administrative access through its SCI. Section 5.3.1 describes the SCI as providing a local command line interface (CLI). 5.4.2 states that once the TOE identifies and authenticates the administrator, the administrator will have access to the SCI's command line interface (CLI) that allows the administrator to manage the TOE.

[Guide] Section 1.5 describes the Local Management interface as performed via the RS-232 serial interface, which provides access to its Serial Console Interface (SCI). Section 4.3 "Connectivity" further describes local administrative access via an RJ45 serial port as a console management interface for local administration of the unit. This interface can be accessed using a third-party terminal emulation application from a directly connected terminal (e.g. PC). The administrator ensures the interface is local by confirming the management device is directly connected to a network port.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.4.5.2 Guidance Activities

See section 2.4.4.1 (of [SD-ND]).

2.4.5.3 Test Activities

The evaluator tests management functions as part of testing the SFRs identified in section 2.4.5. No separate testing for FMT_SMF.1 is required unless one of the management functions in FMT_SMF.1.1 has not already been exercised under any other SFR.

2.4.6 Restrictions on Security Roles (FMT_SMR.2)

2.4.6.1 TSS Activities

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

Section 5.4.1 of [ST] ("Security Roles") states the TOE implements two default administrator roles that together provide the capabilities of the Security Administrator role defined in FMT_SMR.2:

- Black Lantern Security Administrator—responsible for managing all security-related services and configuration of the TOE. The Black Lantern Security Administrator role is able to: add administrative users and roles; modify the password policy; modify behavior of the audit function; manage cryptographic keys, CSRs, and certificates; perform firmware updates; and configure the TOE access banner
- Network Administrator—responsible for managing all network-related services and configurations. The Network Administrator is able to: view and modify network configuration; set the system time; and configure route tables.

It also describes two additional roles that do not contribute to any capabilities required of the Security Administrator role: Application role and Recovery-Agent:

- Application role—responsible for managing all KSI®-related services and configuration
- Recovery-Agent—specific role associated with backing up and recovering the TOE root key.

2.4.6.2 Guidance Activities

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

[Guide] Section 4.3 "Connectivity" describes how the administrator can connect to the local management interface and the RESTful API service port for remote administration. Section 5 describes the necessary steps the Security Administrator must perform to configure both login methods. Section 5.1 covers configuring local login (administering the TOE locally) and Section 5.2 covers configuration for remote login (administering the TOE remotely) and includes importing certificates and CA certificate chains. Section 5.3.2 describes configuration settings for the remote management admininistrator trusted communication path connection and includes setting the remote client and the RESTful API service port configuration.

[Guide] Section 9 “Appendix B” describes how the client can communicate with the Black Lantern using the RESTful Interface for remote administration.

2.4.6.3 Test Activities

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

The evaluator used each of the interfaces of the TOE while performing testing.

2.5 Protection of the TSF (FPT)

2.5.1 Protection of Administrator Passwords (FPT_APW_EXT.1)

2.5.1.1 TSS Activities

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

Section 5.5.1 of [ST] (“Protection of Administrator Passwords”) states the TOE protects administrator passwords using Password Based Key Derivation Function 2 (PBKDF2), as specified in NIST SP 800-132. The TOE’s implementation of PBKDF2 uses HMAC-SHA-256 and a random salt generated by the TOE’s Counter DRBG implementation to derive a pseudorandom value from the password that the TOE then stores, rather than storing the password itself or encrypting the password prior to storage. The TOE does not offer any functions that will disclose to any users a plaintext administrative password.

2.5.1.2 Guidance Activities

None defined.

2.5.1.3 Test Activities

None defined.

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

2.5.2.1 TSS Activities

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

Section 5.2.4 of [ST] (“Cryptographic Key Destruction”), Table 10 (“Private Keys, Symmetric Keys, and CSPs”) lists the keys used by the TOE and how they are stored. Section 5.2.4 states the TOE relies on a top-level key-encrypting key, termed the root key, which is derived from a hardware-based secret called the

one-time programmable master key (OTPMK). The OTPMK is implemented in specially-designed circuity by the chip manufacturer. The TOE uses the root key to protect long-term keys. All pre-shared, symmetric, and private keys that are stored in the TOE are encrypted using 256 bit AES in GCM mode, with the root key as the encryption key. The TOE protects such keys from unauthorized modification or substitution. When keys need to be used by the TOE, it decrypts the keys into volatile memory and does not provide an interface to view those keys. When the TOE is done using a plaintext key, the TOE deallocates the memory location back to the kernel.

2.5.2.2 Guidance Activities

None defined.

2.5.2.3 Test Activities

None defined.

2.5.3 Reliable Time Stamps (FPT_STM_EXT.1)

2.5.3.1 TSS Activities

Modified by TD0632:

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

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

Section 5.5.5 of [ST] ("Reliable Time Stamps") states the TOE includes a real-time clock (RTC) within the TOE hardware. The TOE depends on the RTC to provide accurate date and time for the generated audit records and track inactivity of administrative local sessions. However, there exists inherent drift within the hardware, and therefore the TOE supports synchronization with external NTP servers. The TOE supports NTP v4 as specified in RFC 5905. The TOE authenticates the timestamps it receives from NTP servers using SHA-1, SHA-256, SHA-384, or SHA-512. The Security Administrator can configure up to 10 external NTP servers. The TOE does not update its real-time clock based on timestamps received from broadcast or multicast addresses.

In addition, the Network Administrator can set the system time directly using the `settime` console command.

2.5.3.2 Guidance Activities

Modified by TD0632:

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

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

[Guide] Section 7.5 “Time Synchronization” instructs the administrator how to use the `setconfig` command to establish the communication path between the TOE and up to 10 NTP servers and to configure the NTP client on the TOE. The settime command can be used with either a specific timestamp or NTP server as input to sync with.

2.5.3.3 Test Activities

The evaluator shall perform the following tests:

Test 1: If the TOE supports direct setting of the time by the Security Administrator then the evaluator uses the guidance documentation to set the time. The evaluator shall then use an available interface to observe that the time was set correctly.

The evaluator verified that the administrator was able to set the time of the system and the TOE’s time was set to the specified value.

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.

The evaluator verified that the TOE could synchronize its time with a NTP server and that the time was synchronized correctly.

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.

N/A, the TOE is not composed of multiple parts.

Modified (added) by TD0632:

Test 3: [conditional] If the TOE obtains time from the underlying VS, the evaluator shall record the time on the TOE, modify the time on the underlying VS, and verify the modified time is reflected by the TOE. If there is a delay between the setting the time on the VS and when the time is reflected on the TOE, the evaluator shall ensure this delay is consistent with the TSS and Guidance..

N/A, the TOE is not virtualized and does not obtain time from underlying VS.

2.5.4 TSF Testing (FPT_TST_EXT.1)

2.5.4.1 TSS Activities

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

Section 5.5.3 of [ST] (“TSF Testing”) details the self-tests that are run by the TOE.

During initial start-up of the TOE, each firmware image is verified prior to executing its code, to ensure it has not been modified. If either of the images does not verify correctly, then the TOE reports this as a fault. If neither image verifies correctly, the TOE does not boot up.

In addition to performing verification and decryption of the firmware files, the extended boot also performs a series of Power On Self Tests (POSTs). The tests include memory, register, branch, integer, floating point unit, Timebase and Decrementer tests, Data Cache, sign and verification, and cryptographic algorithms tests (known answer tests). A description is provided for each test that outlines what the test is doing.

If all these tests pass, the overall status on the POST will indicate “OK”. Otherwise, any failed POST will trigger the TOE to display “Failure” on its “Power On Self Test” status. In addition, any errors encountered are printed out at bootup.

The combination of firmware verifications and POSTs performed during bootup, along with the approach taken, enables the TOE to convey to the Administrator that the TOE is operating properly. Furthermore, in the event of catastrophic failures, errors output during bootup will provide the Administrator adequate information to diagnose the issue.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.5.4.2 Guidance Activities

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

Section 7.6 “Self Test” describes the “Failure” on its “Power On Self Test” status display triggered by any failed Power On Self Tests (POSTs). In addition, any errors encountered are printed out at bootup. In addition, the TOE performs verification and decryption of the primary and secondary firmware load files. If either of the images does not verify correctly, then the TOE reports this as a fault (on Boot status). If neither image verifies correctly, the TOE does not boot up. The description in the guidance corresponds to the description of possible errors in the TSS. The actions the administrator should take in response to the error messages are described.

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

The TOE is not distributed. Therefore, this activity is not applicable.

2.5.4.3 Test Activities

It is expected that at least the following tests are performed:

a) Verification of the integrity of the firmware and executable software of the TOE

b) Verification of the correct operation of the cryptographic functions necessary to fulfil any of the SFRs.

Although formal compliance is not mandated, the self-tests performed should aim for a level of confidence comparable to:

a) [FIPS 140-2], chap. 4.9.1, Software/firmware integrity test for the verification of the integrity of the firmware and executable software. Note that the testing is not restricted to the cryptographic functions of the TOE.

b) [FIPS 140-2], chap. 4.9.1, Cryptographic algorithm test for the verification of the correct operation of cryptographic functions. Alternatively, national requirements of any CCRA member state for the security evaluation of cryptographic functions should be considered as appropriate.

The evaluator shall either verify that the self-tests described above are carried out during initial start-up or that the developer has justified any deviation from this.

The TOE performs the self-tests during POST and reports the status in the main terminal with security administrator permissions under the 'getstatus -t post' command.

For distributed TOEs the evaluator shall perform testing of self-tests on all TOE components according to the description in the TSS about which self-test are performed by which component.

The TOE is not distributed. Therefore, this activity is not applicable.

2.5.5 Trusted Update (FPT_TUD_EXT.1)

2.5.5.1 TSS Activities

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

Section 5.5.4 of [ST] ("Trusted Update") states the TOE does not provide automatic updates to the firmware version running on the TOE. Only the Security Administrator is able to query the current executing (i.e., primary) and most recently installed (i.e., secondary) versions of the TOE firmware; both should match since the TOE reboots following installation of a firmware update. The TOE does not install firmware updates with a delayed activation—once the Security Admin initiates the firmware update, the process runs to completion.

Section 5.5.4 indicates the Security Admin and Network Admin can query the currently active firmware version and the most recently loaded firmware version using the `getconfig` SCI command.

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

Section 5.5.4 of [ST] describes the mechanism for updating the system firmware using the `updatebl` SCI command to perform firmware update.

The evaluator verified that Section 5.5.4 of [ST] ("Trusted Update") that the description includes a digital signature verification of the software before installation and that installation fails if the verification fails. The vendor encrypts firmware update packages using 256 bit AES and digitally signs them using ECDSA with NIST curve P-521. The TOE verifies the digital signature and decrypts the image using keys pre-installed during manufacturing of the TOE. The TOE copies the firmware update package from the location specified in the `updatebl` command and proceeds to install the firmware in the primary location. The TOE then reboots. Coming up from reboot, the TOE verifies the firmware installed in the primary location. If verification succeeds, the TOE copies the primary location firmware into the secondary location, and reboots. If the primary location firmware verification fails, the TOE attempts to boot from the secondary location (which still retains the current version of firmware prior to the update attempt).

If any failures occur, including digital signature verification or decryption, the TOE aborts the firmware update process, outputs error messages to the SCI, and records relevant audit logs. In the event of a failed firmware update attempt, the Security Admin can use the information displayed by the SCI and the logs to troubleshoot the incident, and retry.

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

FPT_TUD_EXT.1.2 does not include 'support automatic checking for updates' or 'support automatic updates'. Therefore, this assurance activity is not applicable.

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

The TOE is not distributed. Therefore, this assurance activity is not applicable.

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

The TOE uses a digital signature mechanism to protect the TOE update, not a published hash. Therefore, this assurance activity is not applicable.

2.5.5.2 Guidance Activities

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

[Guide] Section 7.7 “Software Updates” states both the current executing version and the most recently installed version of the Black Lantern firmware can be viewed using the `getconfig` command. The TOE does not install firmware updates with a delayed activation.

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.

[Guide] Section 7.7 describes the verification of the update using digital signature verification. Guardtime Federal is responsible for generating the update image and providing it to the customer. The update image is encrypted and signed by Guardtime Federal. The Black Lantern uses pre-installed keys to verify signature authenticity and decrypt the update image. If the verification or the decryption process of the update image fails, the target Black Lantern will reject the firmware update. If the firmware update fails (due to unsuccessful verification), the Black Lantern aborts the firmware update process and outputs an error message to the serial console. The description corresponds 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.

The TOE uses a digital signature mechanism to protect the TOE update, not a published hash. Therefore, this assurance activity is not applicable.

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

The TOE is not distributed. Therefore, this assurance activity is not applicable.

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

The TOE is not distributed. Therefore, this assurance activity is not applicable.

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

The TOE does not use a certificate-based mechanism for software update digital signature verification. Therefore, this assurance activity is not applicable.

2.5.5.3 Test Activities

The evaluator shall perform the following tests:

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

The evaluator observed that a security administrator was able to update the TOE and as part of the update the TOE automatically rebooted. The evaluator verified that the TOE reported the new software version after the update was applied.

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

1) A modified version (e.g. using a hex editor) of a legitimately signed update
2) An image that has not been signed
3) An image signed with an invalid signature (e.g. by using a different key as expected for creating the signature or by manual modification of a legitimate signature)

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

The evaluator presented update files that were incorrect in each of the specified manners; a legitimate update that had been modified, an unsigned image provided by the vendor, and the unsigned image signed with a key other than valid for verification. The evaluator observed that in each of these cases the TOE rejected the update, did not install the new version, and did not change the currently installed and reported version.

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

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

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

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

N/A, the TOE does not claim published hash.

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

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

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

The TOE is not distributed. Therefore, this test activity is not applicable.

2.6 TOE Access (FTA)

2.6.1 TSF-initiated Termination (FTA_SSL.3)

2.6.1.1 TSS Activities

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

Section 5.6.2 of [ST] (“Session Termination”) states that for remote management of the TOE, the RESTful interface is used. The TOE does not maintain an interactive session over the RESTful API as each request is a self-contained, identified, and authenticated request. As such, the TOE does not establish an authenticated state that is preserved across multiple commands.

2.6.1.2 Guidance Activities

The evaluator shall confirm that the guidance documentation includes instructions for configuring the inactivity time period for remote administrative session termination.

[Guide] Section 7.2.1 include instructions for configuring the inactivity period. This section states that remote management of the Black Lantern is done via the RESTful Interface. The TOE does not maintain an interactive session over the RESTful API as each request is a self-contained, identified, and authenticated request. As such, the TOE does not establish an authenticated state that is preserved across multiple commands. Since there is no interactive remote management interface, this activity is not applicable and vacuously satisfied.

2.6.1.3 Test Activities

For each method of remote administration, the evaluator shall perform the following test:

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

N/A, The TOE does not possess remote interactive sessions for remote management.

2.6.2 User-initiated Termination (FTA_SSL.4)

2.6.2.1 TSS Activities

The evaluator shall examine the TSS to determine that it details how the local and remote administrative sessions are terminated.

Section 5.6.2 of [ST] ("Session Termination") states that the TOE allows administrators to terminate their own local sessions using the exit command. For remote management of the TOE, the RESTful interface is used. The TOE does not maintain an interactive session over the RESTful API as each request is a self-contained, identified, and authenticated request. As such, the TOE does not establish an authenticated state that is preserved across multiple commands.

2.6.2.2 Guidance Activities

The evaluator shall confirm that the guidance documentation states how to terminate a local or remote interactive session.

[Guide] Section 5.1.1 number 8 says that the exit command in the SCI can be used to log out of the session.

Section 8.1 and 8.2 also identifies the exit command admins can use to log out of their local sessions. The description indicates that <quit> and <logout> also work.

[Guide] Section 7.2.1 states that the TOE does not maintain an interactive session over the RESTful API as each request is a self-contained, identified, and authenticated request. As such, the TOE does not establish an authenticated state that is preserved across multiple commands.

2.6.2.3 Test Activities

For each method of remote administration, the evaluator shall perform the following tests:

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.

The evaluator verified that users connected to the TOE locally could end their own sessions.

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.

The TOE does not possess remote interactive sessions and each RESTful command is individually authenticated. Thus, there is no individual logout that can occur other than the RESTful command execution finishing, and the response being provided.

2.6.3 TSF-initiated Session Locking (FTA_SSL_EXT.1)

2.6.3.1 TSS Activities

The evaluator shall examine the TSS to determine that it details whether local administrative session locking or termination is supported and the related inactivity time period settings.

Section 5.6.2 of [ST] ("Session Termination") states the TOE terminates local interactive sessions after a configurable period of inactivity. The Security Administrator uses the `setconfig` console command to set the serialsessiontimeout configuration parameter, which specifies the inactivity timeout value in minutes. At the end of this period of inactivity on the local interactive session, the TOE terminates the session. The serialsessiontimeout parameter can take any integer value between 0 and 2,147,483,647, and has a default value of 300. If serialsessiontimeout is set to 0, session termination is disabled and the local interactive session will never timeout. This is not allowed in the evaluated configuration.

2.6.3.2 Guidance Activities

The evaluator shall confirm that the guidance documentation states whether local administrative session locking or termination is supported and instructions for configuring the inactivity time period.

[Guide] Section 7.2.1 includes instructions for configuring the inactivity period for a local session using the `setconfig` command and Security.SerialSessionTimeout parameter. It indicates that inactive sessions are terminated. Table 10 identifies the range of values as 0 – 2147483647; the default value as 300; and 0 means the serial session will never timeout. Section 3.2.3.4 states that NDcPP compliance requires this inactivity period to be defined greater than 0.

2.6.3.3 Test Activities

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

For multiple inactivity time values, the evaluator connected to the TOE locally to the TOE and verified that after the inactivity time expired the user was automatically logged off of the TOE and had to re-authenticate to perform any actions.

2.6.4 Default TOE Access Banners (FTA_TAB.1)

2.6.4.1 TSS Evaluation Activity

The evaluator shall check the TSS to ensure that it details each administrative method of access (local and remote) available to the Security Administrator (e.g., serial port, SSH, HTTPS). The evaluator shall check the TSS to ensure that all administrative methods of access available to the Security Administrator are listed and that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each administrative method of access. The advisory notice and the consent warning message might be different for different administrative methods of access and might be configured during initial configuration (e.g. via configuration file).

Section 5.3.1 details local administrative access to the TOE Serial Console Interface and remote administrative access to the RESTful interface over an HTTPS channel.

Section 5.6.1 of [ST] ("Access Banner") states the TOE displays a custom login banner prior to an administrative session via the SCI. The Security Admin is the only user role permitted to configure the banner in the TOE. This section also notes that although the RESTful API is used for remote management of the TOE, the TOE does not maintain an interactive session over the RESTful API, so it does not display the login banner over this interface.

The evaluator ensured that the TSS states that the TOE is displaying an advisory notice and a consent warning message for each interactive administrative method of access (which is solely the SCI).

2.6.4.2 Guidance Activities

The evaluator shall check the guidance documentation to ensure that it describes how to configure the banner message.

[Guide] Section 7.3 "Login Banner" states that for the SCI, the banner command can be used to configure a login banner display that can be customized from the default legal login text. Section 8.1 states that the length of the banner cannot contain more than 16,383 characters. If no characters are given while setting the login banner, the login banner will be set to the default text. Table 10 identifies the default text for the configurable parameter: security.legaltextbanner. Due to the nature of the RESTful interface, the login banner is not displayed for this interface.

2.6.4.3 Test Activities

The evaluator shall also perform the following test:

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

The evaluator configured the access banner and observed that the configured banner was presented to the local user prior to authentication being attempted.

2.7 Trusted Path/Channels (FTP)

2.7.1 Inter-TSF Trusted Channel (FTP_ITC.1)

2.7.1.1 TSS Activities

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

Section 5.7 of [ST] ("Trusted Path/Channels") states the TOE can be configured to export audit records to an external syslog server over TLS. In this case, the TOE acts as a TLS client and initiates the connection to

the syslog server. The TOE identifies and authenticates the syslog server by validating the syslog server's X.509 certificate that is presented during the TLS negotiation.

Section 5.3.4 of [ST] states the administrator can import the entire CA chain of (peer) remote hosts that the TOE is expected to communicate with over TLS channels for the purpose of authenticating the peer during the establishment of a TLS channel (dual authentication).

2.7.1.2 Guidance Activities

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

[Guide] Section 5.3 and Section 5.3.1.1 describe how to configure TLS for communication with a remote audit server and includes recovery instructions.

2.7.1.3 Test Activities

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

The evaluator shall perform the following tests:

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.

The evaluator verified that the TOE could establish a session with an authorized IT entity.

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.

The evaluator verified that the TOE could initiate the connection to the authorized IT entity without any action from the administrator other than configuration of the channel.

Test 3: The evaluator shall ensure, for each communication channel with an authorized IT entity, the channel data is not sent in plaintext.

The evaluator verified that the communication was sent in a secure manner and was not sent in plaintext.

Test 4: Objective: The objective of this test is to ensure that the TOE reacts appropriately to any connection outage or interruption of the route to the external IT entities.

The evaluator shall, for each instance where the TOE acts as a client utilizing a secure communication mechanism with a distinct IT entity, physically interrupt the connection of that IT entity for the following durations: i) a duration that exceeds the TOE's application layer timeout setting, ii) a duration shorter than the application layer timeout but of sufficient length to interrupt the network link layer.

The evaluator shall ensure that, when the physical connectivity is restored, communications are appropriately protected and no TSF data is sent in plaintext.

In the case where the TOE is able to detect when the cable is removed from the device, another physical network device (e.g. a core switch) shall be used to interrupt the connection between the TOE and the distinct IT entity. The interruption shall not be performed at the virtual node (e.g. virtual switch) and must be physical in nature.

The evaluator observed that the TOE could recover from an interruption in the core network that broke the route to the IT entity and no traffic was sent in an insecure manner.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of external secure channels to TOE components in the Security Target.

The TOE is not distributed. Therefore, this activity is not applicable.

The developer shall provide to the evaluator application layer configuration settings for all secure communication mechanisms specified by the FTP_ITC.1 requirement. This information should be sufficiently detailed to allow the evaluator to determine the application layer timeout settings for each cryptographic protocol. There is no expectation that this information must be recorded in any public-facing document or report.

2.7.2 Trusted Path (FTP_TRP.1/Admin)

2.7.2.1 TSS Activities

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

Section 5.7 of [ST] states that all remote administrative communication is performed via the RESTful interface, which uses the HTTPS protocol with TLS and mutual authentication. The Security Administrator uses a RESTful client to remotely manage the TOE by sending RESTful requests. Once the administrator is authenticated, the TOE processes the request and the HTTPS connection is terminated.

This is consistent with the selection HTTPS made in FTP_TRP.1/Admin. Furthermore, the corresponding protocol requirements (FCS_HTTPS_EXT.1, FCS_TLSS_EXT.1, FCS_TLSS_EXT.2) are included in the ST.

2.7.2.2 Guidance Activities

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

[Guide] Section 5.2 “Remote Login” describes the TOE’s RESTful Interface provided for the purposes of remote management. The nature of this type of interface does not allow the administrator to initiate remote login sessions to the Black Lantern. Instead, when the administrator needs to remotely manage the Black Lantern, the administrator sends a RESTful request to the Black Lantern. This request is accompanied with authentication credentials, associated with local accounts on the Black Lantern. Once the authentication credentials are authenticated, the Black Lantern processes the RESTful request, and

terminates any communication with the administrator. 5.3.2 indicates that there are no restrictions on the remote client that can be used to communicate with the Black Lantern over the RESTful Interface. The administrator can send a properly formatted RESTful API request using any client (e.g. web browser) to the appropriate URI. Section 9 describes the format of valid PUT and GET requests and provides examples.

2.7.2.3 Test Activities

The evaluated shall perform the following tests. **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.

The evaluator observed that the TOE could be accessed remotely over a secure channel.

Test 2: The evaluator shall ensure, for each communication channel, the channel data is not sent in plaintext.

The evaluator observed that the TOE sent data in an encrypted manner when being accessed remotely.

Further assurance activities are associated with the specific protocols.

For distributed TOEs the evaluator shall perform tests on all TOE components according to the mapping of trusted paths to TOE components in the Security Target.

The TOE is not distributed. Therefore, this activity is not applicable.

3 Security Assurance Requirements

3.1 Class ASE: Security Targeted Evaluation

General ASE

When evaluating a Security Target, the evaluator performs the work units as presented in the CEM. In addition, the evaluator ensures the content of the TSS in the ST satisfies the EAs specified in Section 2 (Evaluation Activities for SFRs).

3.1.1 ASE_TSS.1 TOE Summary Specification for Distributed TOEs

For distributed TOEs only the SFRs classified as 'all' have to be fulfilled by all TOE parts. The SFRs classified as 'One' or 'Feature Dependent' only have to be fulfilled by either one or some TOE parts, respectively. To make sure that the distributed TOE as a whole fulfills all the SFRs the following actions for ASE_TSS.1 have to be performed as part of ASE_TSS.1.1E.

Note that additional Evaluation Activities for the TSS in the case of a distributed TOE are defined in section A.9.1.1 in the SD.

The TOE is not a distributed TOE. Therefore, this activity is not applicable.

3.2 Class ADV: Development

3.2.1 ADV_FSP.1 Basic Functional Specification

The EAs for this assurance component focus on understanding the interfaces (e.g., application programming interfaces, command line interfaces, graphical user interfaces, network interfaces) described in the AGD documentation, and possibly identified in the TOE Summary Specification (TSS) in response to the SFRs. Specific evaluator actions to be performed against this documentation are identified (where relevant) for each SFR in Section 2, and in EAs for AGD, ATE and AVA SARs in other parts of Section 3.

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

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

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

3.2.1.1 ADV_FSP.1 Evaluation Activity

The evaluator shall examine the interface documentation to ensure it describes the purpose and method of use for each TSFI that is identified as being security relevant.

In this context, TSFI are deemed security relevant if they are used by the administrator to configure the TOE, or to perform other administrative functions (e.g. audit review or performing updates). Additionally, those interfaces that are identified in the ST, or guidance documentation, as adhering to the security policies (as presented in the SFRs), are also considered security relevant. The intent is that these interfaces will be adequately tested, and having an understanding of how these interfaces are used in the TOE is necessary to ensure proper test coverage is applied.

The set of TSFI that are provided as evaluation evidence are contained in the Administrative Guidance and User Guidance.

Through review of [ST] and [Guide], the evaluation team identified that the following external interfaces are security relevant:

- RESTful API
- SCI (CLI)
- TLS logical interface
- Syslog interface.

The evaluation team determined the interface documentation described the purpose and method of use for each TSFI identified as being security relevant, sufficient to enable each of the evaluation activities to be completed satisfactorily. The evaluation team's results from performing the evaluation activities are documented in Section 2 of this AAR.

3.2.1.2 ADV_FSP.1 Evaluation Activity

The evaluator shall check the interface documentation to ensure it identifies and describes the parameters for each TSFI that is identified as being security relevant.

The evaluation team determined the interface documentation identified and described the parameters for each TSFI identified as being security relevant, sufficient to enable each of the evaluation activities to be completed satisfactorily. The evaluation team's results from performing the evaluation activities are documented in Section 2 of this AAR.

3.2.1.3 ADV_FSP.1 Evaluation Activity

The evaluator shall examine the interface documentation to develop a mapping of the interfaces to SFRs.

The evaluator uses the provided documentation and first identifies, and then examines a representative set of interfaces to perform the EAs presented in Section 2, including the EAs associated with testing of the interfaces.

It should be noted that there may be some SFRs that do not have an interface that is explicitly "mapped" to invoke the desired functionality. For example, generating a random bit string, destroying a cryptographic key that is no longer needed, or the TSF failing to a secure state, are capabilities that may be specified in SFRs, but are not invoked by an interface.

However, if the evaluator is unable to perform some other required EA because there is insufficient design and interface information, then the evaluator is entitled to conclude that an adequate functional specification has not been provided, and hence that the verdict for the ADV_FSP.1 assurance component is a 'fail'.

The evaluation team examined the interface documentation and was able to map interfaces to SFRs, sufficient to enable each of the evaluation activities to be completed satisfactorily. The evaluation team's results from performing the evaluation activities are documented in Section 2 of this AAR.

3.3 Class AGD: Guidance Documents

It is not necessary for a TOE to provide separate documentation to meet the individual requirements of AGD_OPE and AGD_PRE. Although the EAs in this section are described under the traditionally separate AGD families, the mapping between the documentation provided by the developer and AGD_OPE and AGD_PRE requirements may be many-to-many, as long as all requirements are met in documentation that is delivered to Security Administrators and users (as appropriate) as part of the TOE.

Note that additional Evaluation Activities for the guidance documentation in the case of a distributed TOE are defined in section A.9.1.1 in the SD.

3.3.1 AGD_OPE.1 Operational User Guidance

The evaluator performs the CEM work units associated with the AGD_OPE.1 SAR. Specific requirements and EAs on the guidance documentation are identified (where relevant) in the individual EAs for each SFR.

In addition, the evaluator performs the EAs specified below.

3.3.1.1 AGD_OPE.1 Evaluation Activity

The evaluator shall ensure the Operational guidance documentation is distributed to Security Administrators and users (as appropriate) as part of the TOE, so that there is a reasonable guarantee that Security Administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

The [Guide] is published with the Security Target at the https://www.niap-ccevs.org/ website. The distribution of the documentation provides a reasonable guarantee that administrators and users are aware of the existence and role of the documentation in establishing and maintaining the evaluated configuration.

3.3.1.2 AGD_OPE.1 Evaluation Activity

The evaluator shall ensure that the Operational guidance is provided for every Operational Environment that the product supports as claimed in the Security Target and shall adequately address all platforms claimed for the TOE in the Security Target.

[Guide] Section 1 “Common Criteria Introduction” identifies all of the platforms claimed for the TOE in section 1.1 of [ST] (“Security Target, Target of Evaluation, and Common Criteria Identification”).

The [Guide] provides the following information regarding the Operational Environment for all platforms claimed in the Security Target in the identified document sections: overview of interfaces, protocols, and ports (“Hardware Overview”); initial configuration (“Initial Configuration”); and operational environment that the system must be in to ensure a secure deployment (“Environmental Requirements”, “Operational Environment”). All platforms claimed for the TOE are covered in the guidance.

3.3.1.3 AGD_OPE.1 Evaluation Activity

The evaluator shall ensure that the Operational guidance contains instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

[Guide] Section 1.4 “Cryptographic Support” identifies the Guardtime Federal’s Cryptographic Support Library (CSL) Direct module provided by the TOE. This is the only cryptographic engine included in the TOE and no configuration is necessary to enable the CSL Direct module and/or FIPS mode.

3.3.1.4 AGD_OPE.1 Evaluation Activity

The evaluator shall ensure the Operational guidance makes it clear to an administrator which security functionality and interfaces have been assessed and tested by the EAs.

[Guide] Section 1.1 “Target of Evaluation” lists the functionality that was excluded from evaluation and makes it explicitly clear that only the functionality claimed in [ST] was evaluated.

3.3.1.5 AGD_OPE.1 Evaluation Activity

Modified in accordance with TD0536.

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

a) The guidance documentation shall contain instructions for configuring any cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

b) The documentation must describe the process for verifying updates to the TOE for each method selected for FPT_TUD_EXT.1.3 in the Security Target. The evaluator shall verify that this process includes the following steps:

   1) Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

   2) Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes instructions that describe at least one method of validating the hash/digital signature.

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

Part a) is addressed by section 3.3.1.3 above.

Part b) is addressed in section 2.5.5.2 above.

Part c) is addressed by section 3.3.1.4 above.

3.3.2 AGD_PRE.1 Preparative Procedures

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

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

In addition, the evaluator performs the EAs specified below.

3.3.2.1 AGD_PRE.1 Evaluation Activity

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

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

[Guide] Section 1.5 "Operational Environment" describes the intended operational environment and includes the requirements of the Security Objectives for the Operational Environment specified in the Security Target, sufficient for an administrator to verify the operational environment can fulfil its role to support the evaluated security functionality.

3.3.2.2 AGD_PRE.1 Evaluation Activity

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

[Guide] Section 1.1 "Target of Evaluation (TOE)" identifies all of the platforms claimed for the TOE in section 1.1 of [ST] ("Security Target, Target of Evaluation, and Common Criteria Identification"). The instructions and guidance contained in the [Guide] are applicable to all TOE platforms.

3.3.2.3 AGD_PRE.1 Evaluation Activity

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

[Guide] Section 1.5 "Operational Environment" provides guidance allowing an administrator to deploy the product in an environment consistent with the evaluated configuration. Section 4 provides rack installation, connection and network configuration instructions. Section 2 "Hardware Overview" also identifies the ports and connectivity, whereas the remaining parts of the [Guide] provide additional specific configuration instructions for the interfaces.

3.3.2.4 AGD_PRE.1 Evaluation Activity

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

[Guide] provides instructions for managing the security of the TOE both as a product and as a component of the larger operational environment.

3.3.2.5 AGD_PRE.1 Evaluation Activity

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

The preparative procedures must

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

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

[Guide] Section 3 “Security Overview” and Section 5 “Initial Configuration” include instructions to provide a protected administrative capability. Section “Initial Configuration” also identifies the Initial Configuration File, which contains a Security Administrator account. Guardtime Federal delivers the Configuration file; an encrypted Initial Configuration File; and the credentials for the Initial Security Administrator account. The credentials and the configuration file are delivered using different delivery channels agreed upon by Guardtime Federal and the customer at the time of contract signing. This section instructs the admin to create a new Security Administrator account, and how to delete the initial account.

3.4 Class ALC: Life-Cycle Support

3.4.1 ALC_CMC.1 Labelling of the TOE

When evaluating that the TOE has been provided and is labelled with a unique reference, the evaluator performs the work units as presented in the CEM.

3.4.2 ALC_CMS.1 TOE CM Coverage

When evaluating the developer’s coverage of the TOE in their CM system, the evaluator performs the work units as presented in the CEM.

3.5 Class ATE: Tests

3.5.1 ATE_IND.1 Independent Testing – Conformance

3.5.1.1 ATE_IND.1 Assurance Activity

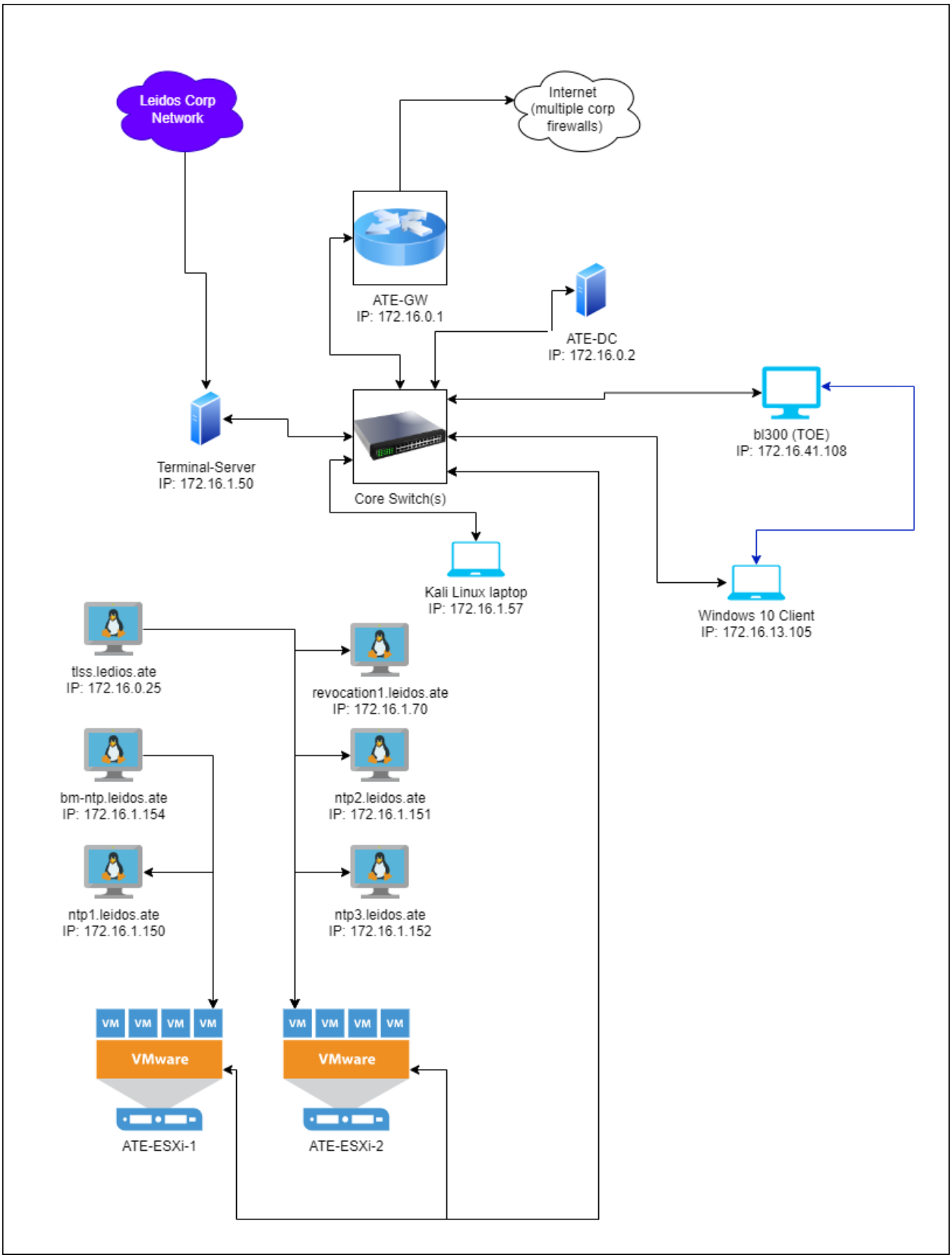
The focus of the testing is to confirm that the requirements specified in the SFRs are being met. Additionally, testing is performed to confirm the functionality described in the TSS, as well as the dependencies on the Operational guidance documentation is accurate.

The evaluator performs the CEM work units associated with the ATE_IND.1 SAR. Specific testing requirements and EAs are captured for each SFR in Sections 2.

The evaluator should consult Appendix A [in the SD] when determining the appropriate strategy for testing multiple variations or models of the TOE that may be under evaluation.

Note that additional Evaluation Activities relating to evaluator testing in the case of a distributed TOE are defined in section A.9.3.1 in the SD.

Testing of the TOE was performed at the Leidos Accredited Testing and Evaluation Lab located in Columbia, Maryland from March 29, 2022 to July 7, 2022. For the purposes of that testing, the configuration depicted in Figure 1 was used as a basis for testing. The test configuration is shown below:



Note: The blue connection between the Windows 10 client and the BL300 (TOE) represents the physical serial connection that is used for local connection/configuration.

The following components were used to create the test configuration:

TOE Device:

BL300:

Model: BL300-B2

Purpose: TOE

IP/MASK/MAC: 172.16.41.108 / 16 / 00:0c:bd:0b:14:b6

Firmware Version: 2.2.1

Protocols used: NTP (client), DNS (client), TLS

Additional Environment Devices:

ATE-GW (Physical)

Purpose: Main router/gateway

IP/MASK/MAC: 172.16.0.1 / 16 / ac:1f:6b:95:0c:1d

OS: PfSense 2.4.4-RELEASE-p2

ATE-DC (Physical)

Purpose: Main Domain Controller (DC) for Test environment/DNS server

IP/MASK/MAC: 172.16.0.2 / 16 / 00:22:19:58:EB:8D

OS: Windows Server 2016 version 1607

Protocols used: RDP, NTP, LDAP, DNS

ATE-ESXi-1 (Physical)

Purpose: Virtualization server

IP/MASK/MAC: 172.16.1.62 / 16 / 10:7b:44:92:77:bf

OS: VMware ESXi, 6.5.0, 5969303

ATE-ESXi-2 (Physical)

Purpose: Virtualization server

IP/MASK/MAC: 172.16.1.63 / 16 / ac:1f:6b:c6:50:96

OS: VMware ESXi, 6.7.0, 13006603

Terminal Server (Physical)

Purpose: Provide tester access to the Test Environment from corporate network.

IP/MASK/MAC: 172.16.1.50 / 16 / D4:BE:D9:B4:FE:66

OS: Windows server 2016 version 1607

Protocols used: RDP, NTP, LDAP, DNS, SSH

TLSS.leidos.ate (VM)

Purpose: Hosts TLS Test Tools

IP/MASK/MAC: 172.16.0.25 / 16 / 00:50:56:b1:66:0b

Host: ATE-ESXi-2

OS: Ubuntu 18.04.5

Protocols Used: SSH, TLS, NTP, DNS

Relevant Software:

Proprietary Python TLS test tools

OpenSSL 1.1.1

Wireshark 2.6.10

Windows 10 Client (Physical)

Purpose: Console Access to TOE

IP/MASK/MAC: 172.16.13.105 / 16 / F0:92:1C:58:E3:C1

OS: Windows 10 version 21H1

Protocols Used: RDP, TLS, NTP, SSH, DNS

Relevant Software:

MobaXterm version 21.3

Putty 0.73

Kali Linux Laptop (Physical)

Purpose: Connection interruption capture NTP Python tools.

IP/MASK/MAC: 172.16.1.57 / MASK / 00:0e:c6:bb:c8:62

OS: Kali Linux 5.2.0-Kali3-amd64

Protocols Used: SSH, TLS, NTP, DNS

Relevant Software:

Wireshark 3.0.5

Python

Revocation1.leidos.ate (VM)

Purpose: Hosts TLS test tools

IP/MASK/MAC: 172.16.1.70 / 16 / 00:50:56:b1:a0:fd

Host: ATE-ESXi-2

OS: Ubuntu 18.04.4

Protocols Used: SSH, TLS, NTP, DNS, OCSP

Relevant Software:

OpenSSL 1.1.1

NTP1.leidos.ate (VM)

Purpose: NTP server in cluster

IP/MASK/MAC: 172.16.1.150 / 16 / 00:50:56:b1:9f:2e

Host: ATE-ESXi-1

OS: Ubuntu 18.04.4 LTS

Protocols Used: SSH, NTP, DNS

Relevant Software:

Ntpd 4.2.8p10

NTP2.leidos.ate (VM)

Purpose: NTP server in cluster

IP/MASK/MAC: 172.16.1.151 / 16 / 00:50:56:b1:ae:31

OS: Ubuntu 18.04.4 LTS

Protocols Used: SSH, NTP, DNS

Relevant Software:

Ntpd 4.2.8p10

NTP3.leidos.ate (VM)

Purpose: NTP server in cluster

IP/MASK/MAC: 172.16.1.152 / 16 / 00:50:56:b1:57:09

Host: ATE-ESXi-2

OS: Ubuntu 18.04.4 LTS

Protocols Used: SSH, NTP, DNS

Relevant Software:

Ntpd 4.2.8p10

Bm-NTP.leidos.ate (VM)

Purpose: NTP server for Broadcast and Multicast transmission

IP/MASK/MAC: 172.16.1.154 / 16 / 00:50:56:b1:87:af

Host: ATE-ESXi-1

OS: Ubuntu 18.04.6 LTS

Protocols Used: SSH, NTP, DNS

Relevant Software:

Ntpd 4.2.8p10

The evaluation team followed the installation and configuration procedures documented in the product guidance to install the TOE in the test environment.

Subsequently, the evaluators exercised all the test cases. The tests were selected in order to ensure that each of the test assertions specified in *Evaluation Activities for Network Device cPP* were covered. All tests passed.

3.6 Class AVA: Vulnerability Assessment

3.6.1 AVA_VAN.1 Vulnerability Survey

While vulnerability analysis is inherently a subjective activity, a minimum level of analysis can be defined and some measure of objectivity and repeatability (or at least comparability) can be imposed on the vulnerability analysis process. In order to achieve such objectivity and repeatability it is important that the evaluator follows a set of well-defined activities and documents their findings so others can follow their arguments and come to the same conclusions as the evaluator. While this does not guarantee that different evaluation facilities will identify exactly the same type of vulnerabilities or come to exactly the same conclusions, the approach defines the minimum level of analysis and the scope of that analysis and provides Certification Bodies a measure of assurance that the minimum level of analysis is being performed by the evaluation facilities.

In order to meet these goals some refinement of the AVA_VAN.1 CEM work units is needed. The following table indicates, for each work unit in AVA_VAN.1, whether the CEM work unit is to be performed as written, or if it has been clarified by an Evaluation Activity. If clarification has been provided, a reference to this clarification is provided in the table.

Because of the level of detail required for the evaluation activities, the bulk of the instructions are contained in Appendix A in the SD, while an “outline” of the assurance activity is provided below.

3.6.1.1 AVA_VAN.1 Evaluation Activity (Documentation)

Modified in accordance with TD0547.

In addition to the activities specified by the CEM in accordance with Table 2, the evaluator shall perform the following activities.

The evaluator shall examine the documentation outlined below provided by the developer to confirm that it contains all required information. This documentation is in addition to the documentation already required to be supplied in response to the EAs listed previously.

The developer shall provide documentation identifying the list of software and hardware components[1] that compose the TOE. Hardware components should identify at a minimum the processors used by the TOE. Software components include applications, the operating system and other major components that are independently identifiable and reusable (outside the TOE), for example a web server, protocol or cryptographic libraries (independently identifiable and reusable components are not limited to the list provided in the example). This additional documentation is merely a list of the name and version number of the components, and will be used by the evaluators in formulating hypotheses during their analysis.

Section 1.8 of [ST] ("Physical Boundaries") identifies the components that compose the TOE. The information includes the TOE firmware version, the device models, the processor, and the operating system. Section 1.9.2 ("Cryptographic Support") states that the TOE also includes Guardtime Federal's Cryptographic Support Library (CSL) Direct v2.0.0 cryptographic module, which provides the CAVP-certified cryptographic services.

If the TOE is a distributed TOE then the developer shall provide:

a) documentation describing the allocation of requirements between distributed TOE components as in [NDcPP, 3.4]

b) a mapping of the auditable events recorded by each distributed TOE component as in [NDcPP, 6.3.3]

c) additional information in the Preparative Procedures as identified in the refinement of AGD_PRE.1 in additional information in the Preparative Procedures as identified in 3.4.1.2 and 3.5.1.2.

The TOE is not distributed.

3.6.1.2 AVA_VAN.1 Evaluation Activity

The evaluator formulates hypotheses in accordance with process defined in Appendix A in the SD. The evaluator documents the flaw hypotheses generated for the TOE in the report in accordance with the guidelines in Appendix A.3 in the SD. The evaluator shall perform vulnerability analysis in accordance with Appendix A.2 in the SD. The results of the analysis shall be documented in the report according to Appendix A.3 in the SD.

The evaluation team performed a search of the National Vulnerability Database (https://nvd.nist.gov/) on 10 May 2022, with the most recent search on 1 September 2022, using the following search terms:

- BL300-B2
- BL300-C2
- BL400-A1
- NXP T4240r2 QorIQ
- Power Architecture

[1] In this sub-section the term "components" refers to parts that make up the TOE. It is therefore distinguished from the term "distributed TOE components", which refers to the parts of a TOE that are present in one physical part of a distributed TOE. Each distributed TOE component will therefore generally include a number of the hardware and software components that are referred to in this sub-section: for example, each distributed TOE component will generally include hardware components such as processors and software components such as an operating system and libraries.

- BLKSI.2.2.1-FIPS
- Cryptographic Support Library (CSL) Direct
- Green Hills

No vulnerabilities were identified for the TOE.

The evaluation team generated type 4 flaw hypotheses and performed testing. The evaluation team did not identify or formulate any flaws arising from review of evaluation evidence or through testing of the TOE (Type 3 flaw hypothesis).

The evaluation team determined that no residual vulnerabilities exist that are exploitable by attackers with Basic Attack Potential.