Log Correlation Engine 6.0.6

Security Target

Version 1.0

04 December 2020

Prepared for:

Tenable, Inc.
7021 Columbia Gateway Dr.
Columbia, MD 21046

Prepared by:

Accredited Testing and Evaluation Labs
6841 Benjamin Franklin Drive
Columbia, MD 21046

Contents

Tables

1 Security Target Introduction

The Security Target (ST) contains the following additional sections:

- Product and TOE Description (Section 2)
- Security Problem Definition (Section 3)
- Security Objectives (Section 4)
- IT Security Requirements (Section 5)
-
- The TLS Package does contain evaluation activities for how to evaluate its SFR claims as part of the evaluation of ASE_TSS.1, AGD_OPE.1, AGD_PRE.1, and ATE_IND.1. All Security Functional Requirements specified by the TLS Package will be evaluated in the manner specified in that package.

- TOE Summary Specification (Section 0)
-

- Protection Profile Claims (Section ⍰)
- This ST is conformant to the *Protection Profile for Application Software, Version 1.3, 1 March 2019* (App PP) and *Functional Package for Transport Layer Security (TLS), Version 1.1, February 12,* 2019 (TLS Package) along with all applicable errata and interpretations from the certificate issuing scheme.

The TOE consists of a software application that runs on a Linux operating system as its platform.

As explained in section 3, Security Problem Definition, the Security Problem Definition of the App PP has been included by reference into this ST.

As explained in section 4, Security Objectives, the Security Objectives of the App PP has been included by reference into this ST.

All claimed SFRs are defined in the App PP and TLS Package. All mandatory SFRs are claimed. No optional or objective SFRs are claimed. Selection-based SFR claims are consistent with the selections made in the mandatory SFRs that prompt their inclusion.

- Rationale (Section 0)
- TOE Usage of Third-Party Components (Appendix A)

1.1 Security Target, TOE and CC Identification

ST Title –Log Correlation Engine 6.0.6 Security Target

ST Version – Version 1.0

ST Date – 04 December 2020

TOE Identification – Log Correlation Engine (also known as LCE) 6.0.6, supported on RHEL 7

TOE Developer – Tenable, Inc.

Evaluation Sponsor – Tenable, Inc.

CC Identification – Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, April 2017

1.2 Conformance Claims

This ST and the TOE it describes are conformant to the following CC specifications:

- *Protection Profile for Application Software, Version 1.3, 01 March 2019* (App PP) with the following optional and selection-based SFRs:
  - FCS_CKM.1(1)
  - FCS_CKM.1(3)
  - FCS_CKM.2
  - FCS_COP.1(1)
  - FCS_COP.1(2)
  - FCS_COP.1(3)
  - FCS_COP.1(4)
  - FCS_HTTPS_EXT.1/Server (as specified in NIAP TD0473)
  - FCS_RBG_EXT.2
- *Functional Package for Transport Layer Security (TLS), Version 1.1, February 12, 2019* (TLS Package) with the following optional and selection-based SFRs:
  - FCS_TLSS_EXT.1
- The following NIAP Technical Decisions apply to the TOE and have been accounted for in the ST development and the conduct of the evaluation, or were considered to be non-applicable:

  TD0416: Correction to FCS_RBG_EXT.1 Test Activity

  - No change to ST; affects only test evaluation activities.

  TD0427: Reliable Time Source

  - No change to ST; the ST includes the PP's assumptions by reference and therefore any changes to the assumptions are implicitly made.

  TD0434: Windows Desktop Applications Test

- N/A; TD only affects evaluation for Windows applications and the TOE does not have a Windows version.

TD0435: Alternative to SELinux for FPT_AEX_EXT.1.3

- No change to ST; affects only evaluation activities.

TD0437: Supported Configuration Mechanism

- Changes text selection for FMT_MEC_EXT.1.1. This change has been applied to this ST.

TD0442: Updated TLS Ciphersuites for TLS Package

- No change to ST; affects selections in FCS_TLSS_EXT.1 that are not applicable to the TOE.

TD0444: IPsec Selections

- No change to ST; affects selections in FTP_DIT_EXT.1.1 that are not applicable to the TOE.

TD0445: User Modifiable File Definition

- N/A; TD only affects evaluation for Windows applications and the TOE does not have a Windows version.

TD0465: Configuration Storage for .NET Apps

- N/A; the TOE is not a .NET application.

TD0469: Modification of test activity for FCS_TLSS_EXT.1.1 test 4.1

- No change to ST; affects only evaluation activities.

TD0473: Support for Client or Server TOEs in FCS_HTTPS_EXT

- Changes FCS_HTTPS_EXT.1. This change has been applied to the ST.

TD0486: Removal of PP-Module for VPN Clients from allowed-with list

- N/A; the TOE does not have VPN Client functionality so no attempt was made to claim the VPN Client PP-Module. This TD modifies selections in FDP_DAR_EXT.1.1, but the ST does not choose any of the modified selections so there is no change to the SFR.

TD0495: FIA_X509_EXT.1.2 Test Clarification

- N/A; the TOE does not claim FIA_X509_EXT.1.

TD0498: Application Software PP Security Objectives and Requirements Rationale

- No change to ST; the TD modified portions of the App PP that were not reproduced in the ST.

TD0499: Testing with pinned certificates

- N/A; the TOE does not claim FCS_TLSC_EXT.1.

TD0510: Obtaining random bytes for iOS/macOS

- N/A; the TOE does not use iOS or macOS as its platform.

TD0513: CA Certificate loading

- N/A; the TOE does not claim FCS_TLSC_EXT.1.

TD0515: Use Android APK manifest in test

- N/A; the TOE does not include an Android platform version.

TD0519: Linux symbolic links and FMT_CFG_EXT.1

- No change to ST; affects only evaluation activities.

TD0521: Updates to Certificate Revocation (FIA_X509_EXT.1)

- N/A; the TOE does not claim FIA_X509_EXT.1.

TD0540: Expanded AES Modes in FCS_COP

- No change to ST; affects selections in FCS_COP.1(1) that are not applicable to the TOE.

TD0543: FMT_MEC_EXT.1 evaluation activity update

- N/A; the TOE does not include a Windows platform version.

TD0544: Alternative testing methods for FPT_AEX_EXT.1.1

- No change to ST; affects only evaluation activities.

TD0548: Integrity for installation tests in AppSW PP 1.3

- No change to ST; affects only evaluation activities.

TD0554: iOS/iPadOS/Android AppSW Virus Scan

- No change to ST; affects only evaluation activities.

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

1.3 Conventions

The following conventions have been applied in this document:

- Security Functional Requirements – Part 2 of the CC defines the approved set of operations that may be applied to functional requirements: iteration, assignment, selection, and refinement.
  - Iteration: allows a component to be used more than once with varying operations. An iterated SFR is indicated by a number in parentheses placed at the end of the component. For example, FCS_COP.1(1) through FCS_COP.1(4) indicate that the ST includes four iterations of the FCS_COP.1 requirement: (1), (2), (3), and (4).

  - Assignment: allows the specification of an identified parameter. Assignments are indicated using italics and are surrounded by brackets (e.g., [*assignment item*]). Note that an assignment within a selection would be identified in both italics and underline, with the brackets themselves underlined since they are explicitly part of the selection text, unlike the brackets around the selection itself (e.g., [selection item, [*assignment item inside selection*]]).
  - Selection: allows the specification of one or more elements from a list. Selections are indicated using underlines and are surrounded by brackets (e.g., [selection item]).
  - Refinement: allows technical changes to a requirement to make it more restrictive and allows non-technical changes to grammar and formatting. Refinements are indicated using bold, for additions, and strike-through, for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …"). Note that minor grammatical changes that do not involve the addition or removal of entire words (e.g., for consistency of quantity such as changing "meets" to "meet") do not have formatting applied.

- Other sections of the ST – Other sections of the ST use bolding to highlight text of special interest, such as captions.
- The ST does not show operations that have been completed by the PP authors, though it does preserve brackets to show where such operations have been made.

1.3.1 Terminology

The following terms and abbreviations are used in this ST:

Table 1: Terms and Definitions

Term	Definition
Log Correlation Engine	The TOE; an application that is responsible for collecting log data from a variety of sources and aggregating it into a single collection of results.
Nessus Agent	An environmental component that is installed on an endpoint system to collect details about that system's configuration and behavior.
Nessus Network Monitor	An environmental component that collects and analyzes raw network traffic.
Nessus/Nessus Manager	An environmental component that conducts remote scans of systems to collect data about their configuration and behavior and is used to deploy and collect data from the TOE.
Platform	A general-purpose computer on which the TOE is installed.
Tenable.sc (SecurityCenter)	An environmental component that functions as a centralized aggregator for data collected by the TOE and by other environmental components.

1.3.2 Acronyms

Table 2: Acronyms

Term	Definition
API	Application Programming Interface
AES	Advanced Encryption Standard
ASLR	Address Space Layout Randomization
CAVP	Cryptographic Algorithm Validation Program

CBC	Cipher Block Chaining
CC	Common Criteria for Information Technology Security Evaluation
CCECG	Common Criteria Evaluated Configuration Guidance
CEM	Common Evaluation Methodology for Information Technology Security
CTR	Counter (cryptographic mode)
CVE	Common Vulnerabilities and Exposures
DRBG	Deterministic Random Bit Generator
EAR	Entropy Analysis Report
ECC	Elliptic Curve Cryptography
ECDHE	Elliptic Curve Diffie-Hellman (Ephemeral)
ECDSA	Elliptic Curve Digital Signature Algorithm
FIPS	Federal Information Processing Standard
GB	Gigabyte
GCM	Galois/Counter Mode
GUI	Graphical User Interface
HMAC	Hashed Message Authentication Code
IOPS	Input/Output Operations Per Second
LCE	Log Correlation Engine
NIAP	National Information Assurance Partnership
NIST	National Institute of Standards and Technology
NNM	Nessus Network Monitor
OCSP	Online Certificate Status Protocol
OE	Operational Environment
OS	Operating System
PBKDF	Password-Based Key Derivation Function
PII	Personally Identifiable Information
PKI	Public Key Infrastructure
PP	Protection Profile
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RSA	Rivest, Shamir and Adleman (algorithm for public-key cryptography)
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SSH	Secure Shell
SSL	Secure Sockets Layer
ST	Security Target
TB	Terabyte

TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Function

2 Product and TOE Description

2.1 Introduction

Log Correlation Engine (LCE) is a software product that is designed to collect log data from various environmental sources to detect potential security and compliance violations.

LCE also connects to an environmental instance of Tenable.sc (SecurityCenter) which serves as a single point to aggregate and analyze data collected from various Tenable applications, including LCE.

The TOE conforms to the App PP and TLS Package. As such, the security-relevant functionality of the product is limited to the claimed requirements in those standards. The security-relevant functionality is described in sections 2.3 and 2.4. The product overview in section 2.2 below is intended to provide the reader with an overall summary of the entire product so that its intended usage is clear. The subset of the product functionality that is within the evaluation scope is subsequently described in the sections that follow it.

2.2 Product Overview

LCE is a vulnerability management product that is designed to provide visibility into organizational assets through analysis of the log data these assets produce. The product is used to collect information about its environment that is used to diagnose and enhance the security posture of the environment. It does this by aggregating, normalizing, and analyzing event log data from various sources. This log data can then be used to establish baseline behavior for the target assets such that departures from this baseline may be indicative of vulnerability exploitation or compliance violations.

LCE also supports plugins, which can be downloaded and added to the product to detect specific vulnerabilities.

2.3 TOE Overview

The Target of Evaluation (TOE) for LCE consists of the mandatory functionality prescribed by the App PP and TLS Package, as well as some selection-based functionality where needed.

The logical boundary is summarized in section 2.4.2 below. In general, the following LCE capabilities are considered to be within the scope of the TOE:

- **Protection of sensitive data at rest:** the TOE uses encryption to protect credentials and other sensitive data.
- **Protection of data in transit:** the TOE secures data in transit between itself and its operational environment using TLS and HTTPS. Note that the TOE also has one logical interface that uses SSH but it relies on the underlying OS platform to provide this.
- **Trusted updates:** the TOE provides visibility into its current running version and the vendor distributes updates to it that are digitally signed so that administrators can securely maintain up-to-date software.
- **Remote administration:** the TOE provides a Web GUI to administer its security functions. Note however that the bulk of the product's administration functions are outside the scope of the App PP and TLS Package and are therefore not part of the TOE.

- **Cryptographic services:** the TOE includes an implementation of OpenSSL with NIST-validated algorithm services that it uses to secure data at rest and in transit.
- **Secure interaction with operating system:** the TOE is designed to interact with underlying host operating system platforms in such a way that the TOE cannot be used as an attack vector to compromise an operating system.

The TOE's data collection and analysis activities are outside the scope of the TOE, as is any other product behavior that is not described in the App PP or TLS Package. The content and execution of plugins is similarly excluded from the TOE, although they are discussed in the context of network communications because the TSF must use platform network resources to acquire them.

2.4 TOE Architecture

The LCE TOE consists of the LCE application, which is a C/C++ application with a JavaScript web front-end running on a proprietary web server. The TOE is a Linux application.

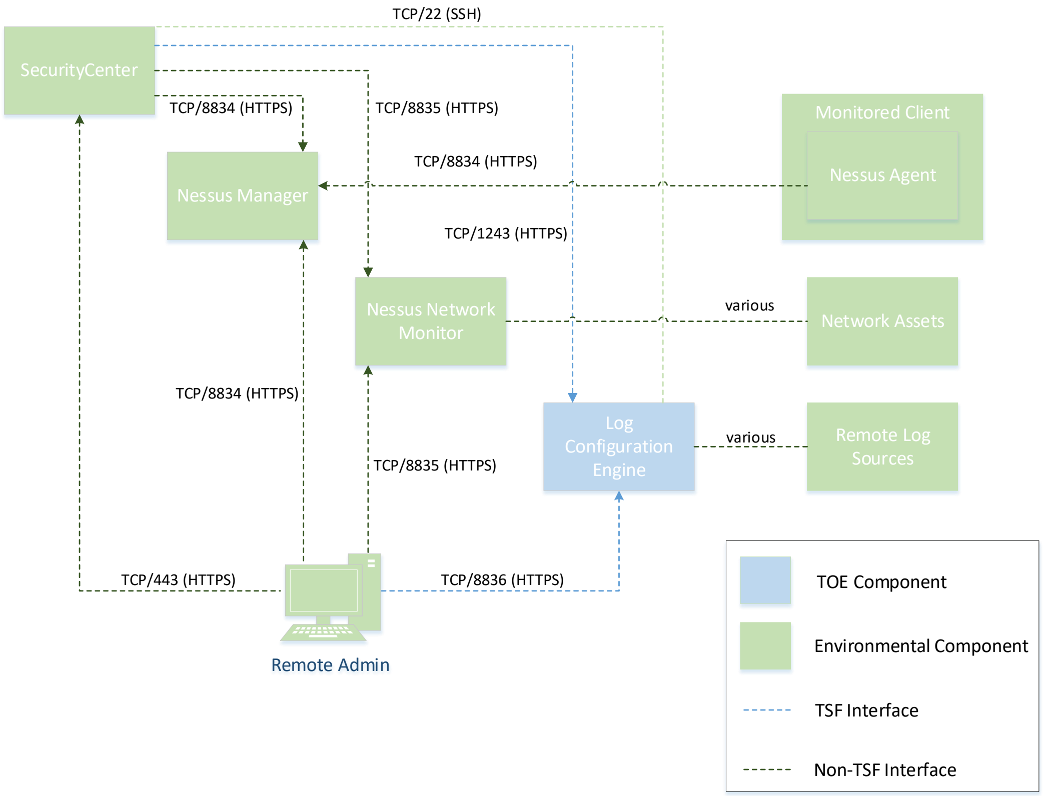
2.4.1 Physical Boundary

The TOE consists of the following component, as shown in Figure 1 below:

- Log Correlation Engine (LCE) 6.0.6

Figure 1 shows the TOE in a sample deployment with other Tenable applications in its operational environment.

Figure 1 - TOE Boundary



TSF-relevant remote interfaces are shown in Figure 1. Note that the TOE consists of exactly one instance of LCE.

The TOE has the following system requirements for its host platform:

- 8 total processor cores
- 12 GB RAM
- 1 TB disk storage (10,000 to 15,000 RPM HDD or SSD or equivalent IOPS capability in RAID 0/10 configuration)

These system requirements reflect the lightest usage scenarios for the TOE. Additional factors such as network size and storage retention requirements will affect the system requirements for a particular deployment. Refer to the relevant TOE documentation (as referenced in section 2.5) for the specific system requirements that apply to a given deployment.

The following network ports must be open for the TOE to function:

- TCP/22 (for communications between Tenable.sc and LCE)

Additional network ports must be open, but these are configurable if the default ports cannot be used. The connections and their default ports are as follows:

- TCP/1243 (for communications between Tenable.sc and LCE)
- TCP/8836 (for administrator communications)

LCE also requires relevant ports and IP addresses to be accessible for the collection of log data from various systems in the operational environment. This functionality is not within the scope of the TOE because it is not “sensitive data” but it is necessary for the product to function as advertised. In particular, if network configuration between LCE and a target system blocks traffic to that system, it may result in a false positive or false negative because the lack of log data could either hide malicious activity or, if applied after an initial baseline has been established, the lack of log data from a system could be reported as anomalous even if the system itself continues to behave in the intended manner.

The TOE’s operational environment includes the following:

- Other Tenable components (an instance of Tenable.sc—Nessus Manager, Nessus Agent(s), and Nessus Network Monitor are expected to be present in the TOE’s operational environment because they also interface with Tenable.sc but the TOE does not interact with these applications directly).
- Platform (hardware and software) on which the TOE is hosted.
    - The TOE is capable of running on a general-purpose Linux operating system on standard consumer-grade hardware on either a physical or virtual machine. For the evaluated configuration, the TOE was tested on a virtualized instance of RHEL 7 running on VMware ESXi 6.5 on a system using an AMD Ryzen Threadripper 1950X processor with the Zen microarchitecture.
- Full disk encryption is required for the TOE platform to ensure adequate data-at-rest protection.
- The platform on which the TOE is deployed is required to provide SSH server functionality through its host operating system.
- Web browser, used to access the GUI interface.

2.4.2 Logical Boundary

This section summarizes the security functions provided by the TOE:

- Timely Security Updates
- Cryptographic Support
- User Data Protection
- Security Management
- Privacy
- Protection of the TSF
- Trusted Path/Channels

2.4.2.1 Timely Security Updates

The TOE developer has internal mechanisms for receiving reports of security flaws, tracking product vulnerabilities, and distributing software updates to customers in a timely manner.

2.4.2.2 Cryptographic Support

The TOE implements cryptography to protect data at rest and in transit.

For data at rest, the TOE stores credential data to log in to the TOE as well as passphrase data used to protect PKI certificates that the TOE uses to authenticate to environmental components. This stored data is encrypted using AES or a PBKDF, depending on the data that is being stored.

For data in transit, the TOE implements TLS/HTTPS as a server. The TOE implements a TLS server for its administrative interface and to communicate with other Tenable products in its operational environment. The TOE does not support mutual authentication.

The TOE implements all cryptography used for these functions using its own implementations of OpenSSL with NIST-approved algorithms. The TOE's DRBG is seeded using entropy from the underlying OS platform.

Some product functionality requires the use of SSH; the TOE does not claim SSH functionality as it invokes its platform to implement this.

2.4.2.3 User Data Protection

The TOE uses cryptographic mechanisms to protect sensitive data at rest. Credential data is protected through the use of a PBKDF while all other sensitive data is protected by the TOE platform's suse of full disk encryption.

The TOE relies on the network connectivity and system log capabilities of its host OS platform. The TOE supports user-initiated, externally-initiated, and application-initiated uses of the network.

2.4.2.4 Security Management

The TOE itself and the configuration settings it uses are stored in locations recommended by the platform vendor.

The TOE includes a web GUI. The web GUI enforces username/password authentication using locally-stored credentials that are created using the TOE. The TOE does not include a default user account to access its management interface.

The security-relevant management functions supported by the TOE relate to configuration of transmission of system data (through collection of log data from external systems).

2.4.2.5 Privacy

The TOE does not handle personally identifiable information (PII) of any individuals.

2.4.2.6 Protection of the TSF

The TOE enforces various mechanisms to prevent itself from being used as an attack vector to its host OS platform. The TOE implements address space layout randomization (ASLR), does not allocate any memory with both write and execute permissions, does not write user-modifiable files to directories that contain executable files, is compiled using stack overflow protection, and is compatible with the security features of its host OS platform.

The TOE contains libraries and invokes system APIs that are well-known and explicitly identified.

The TOE has a mechanism to determine its current software version. Software updates to the TOE can be acquired by leveraging its OS platform. All updates are digitally signed to guarantee their authenticity and integrity.

2.4.2.7 Trusted Path/Channels

The TOE encrypts sensitive data in transit between itself and its operational environment using TLS and HTTPS. It facilitates the transmission of sensitive data from remote users over TLS and HTTPS.

The TOE may also invoke OS platform functionality to establish SSH communications with an instance of Tenable.sc in its operational environment.

2.5 TOE Documentation

Tenable provides the following product documentation in support of the installation and secure use of the TOE:

- Log Correlation Engine 6.0.x User Guide, Last Updated: November 6, 2020

3 Security Problem Definition

This ST includes by reference the Security Problem Definition, composed of threats and assumptions, from the App PP, including the inclusion of A.PLATFORM as required by TD0427. The Common Criteria also provides for organizational security policies to be part of a security problem definition, but no such policies are defined in the App PP.

As a functional package, the TLS Package does not contain a Security Problem Definition. The TOE's use of TLS is intended to mitigate the T.NETWORK_ATTACK and T.NETWORK_EAVESDROP threats defined by the App PP.

In general, the threat model of the App PP is designed to protect against the following:

- Disclosure of sensitive data at rest or in transit that the user has a reasonable expectation of security for.
- Excessive or poorly-implemented interfaces with the underlying platform that allow an application to be used as an intrusion point to a system.

This threat model is applicable to the TOE because aggregated and analyzed vulnerability scan results could show an attacker what system weaknesses are present in the environment if they were able to obtain this data. It is also applicable because the TOE is a collection of executable binaries that an attacker could attempt to use to compromise the underlying OS platform if it was designed in such a manner that this exploitation was possible.

4 Security Objectives

Like the Security Problem Definition, this ST includes by reference the security objectives defined in the App PP. This includes security objectives for the TOE (used to mitigate threats) and for its operational environment (used to satisfy assumptions).

As a functional package, the TLS Package does not contain a Security Problem Definition. The TOE’s use of TLS is intended to satisfy the O.PROTECTED_COMMS objective of the App PP by implementing a specific method by which network communications are protected.

5 IT Security Requirements

This section defines the Security Functional Requirements (SFRs) and Security Assurance Requirements (SARs) that serve to represent the security functional claims for the Target of Evaluation (TOE) and to scope the evaluation effort.

The SFRs have all been drawn from the following Protection Profiles (PP) and Functional Packages:

- *Protection Profile for Application Software*, Version 1.3, March 1, 2019
- *Functional Packages for Transport Layer Security (TLS),* Version 1.1, February 12, 2019

As a result, any selection, assignment, or refinement operations already performed by that PP on the claimed SFRs are not identified here (i.e., they are not formatted in accordance with the conventions specified in section 1.3 of this ST). Formatting conventions are only applied on SFR text that was chosen at the ST author's discretion.

5.1 Extended Requirements

All of the extended requirements in this ST have been drawn from the App PP and TLS Package. These documents define the following extended SAR and extended SFRs; since they have not been redefined in this ST, the App PP and TLS Package should be consulted for more information regarding these extensions to CC Parts 2 and 3.

Defined in App PP:

- ALC_TSU_EXT.1 Timely Security Updates
- FCS_CKM_EXT.1 Cryptographic Key Generation Services
- FCS_CKM.1(3) Password Conditioning
- FCS_HTTPS_EXT.1/Server HTTPS Protocol (as specified in NIAP TD0473)
- FCS_RBG_EXT.1 Random Bit Generation Services
- FCS_RBG_EXT.2 Random Bit Generation from Application
- FCS_STO_EXT.1 Storage of Credentials
- FDP_DAR_EXT.1 Encryption of Sensitive Application Data
- FDP_DEC_EXT.1 Access to Platform Resources
- FDP_NET_EXT.1 Network Communications
- FMT_CFG_EXT.1 Secure by Default Configuration
- FMT_MEC_EXT.1 Supported Configuration Mechanism
- FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information
- FPT_AEX_EXT.1 Anti-Exploitation Capabilities
- FPT_API_EXT.1 Use of Supported Services and APIs
- FPT_IDV_EXT.1 Software Identification and Versions
- FPT_LIB_EXT.1 Use of Third Party Libraries
- FPT_TUD_EXT.1 Integrity for Installation and Update
- FPT_TUD_EXT.2 Integrity for Installation and Update
- FTP_DIT_EXT.1 Protection of Data in Transit (iterated by ST author)

Defined in TLS Package:

- FCS_TLS_EXT.1 TLS Protocol
- FCS_TLSS_EXT.1 TLS Server Protocol

5.2 TOE Security Functional Requirements

The following table identifies the SFRs that are satisfied by the TOE.

Table 3: TOE Security Functional Components

Requirement Class	Requirement Component
FCS: Cryptographic Support	FCS_CKM.1(1) Cryptographic Asymmetric Key Generation
	FCS_CKM.1(3) Password Conditioning
	FCS_CKM.2 Cryptographic Key Establishment
	FCS_CKM_EXT.1 Cryptographic Key Generation Services
	FCS_COP.1(1) Cryptographic Operation – Encryption/Decryption
	FCS_COP.1(2) Cryptographic Operation – Hashing
	FCS_COP.1(3) Cryptographic Operation – Signing
	FCS_COP.1(4) Cryptographic Operation – Keyed-Hash Message Authentication
	FCS_HTTPS_EXT.1/Server HTTPS Protocol
	FCS_RBG_EXT.1 Random Bit Generation Services
	FCS_RBG_EXT.2 Random Bit Generation from Application
	FCS_STO_EXT.1 Storage of Credentials
	FCS_TLS_EXT.1 TLS Protocol (TLS Package)
	FCS_TLSS_EXT.1 TLS Server Protocol (TLS Package)
FDP: User Data Protection	FDP_DAR_EXT.1(1) Encryption of Sensitive Application Data (by TOE)
	FDP_DAR_EXT.1(2) Encryption of Sensitive Application Data (by OE)
	FDP_DEC_EXT.1 Access to Platform Resources
	FDP_NET_EXT.1 Network Communications
FMT: Security Management	FMT_CFG_EXT.1 Secure by Default Configuration
	FMT_MEC_EXT.1 Supported Configuration Mechanism
	FMT_SMF.1 Specification of Management Functions
FPR: Privacy	FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information
FPT: Protection of the TSF	FPT_AEX_EXT.1 Anti-Exploitation Capabilities
	FPT_API_EXT.1 Use of Supported Services and APIs
	FPT_IDV_EXT.1 Software Identification and Versions
	FPT_LIB_EXT.1 Use of Third Party Libraries
	FPT_TUD_EXT.1 Integrity for Installation and Update
	FPT_TUD_EXT.2 Integrity for Installation and Update
FTP: Trusted Path/Channels	FTP_DIT_EXT.1(1) Protection of Data in Transit
	FTP_DIT_EXT.1(2) Protection of Data in Transit

5.2.1 Cryptographic Support (FCS)

5.2.1.1 FCS_CKM.1(1) Cryptographic Asymmetric Key Generation

FCS_CKM.1.1(1) The application shall [

- implement functionality

] to generate asymmetric cryptographic keys in accordance with a specified cryptographic key generation algorithm [

- [ECC schemes] using ["NIST curves" P-256, P-384 and [no other curves]] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4]

].

5.2.1.2 FCS_CKM.1(3) Password Conditioning

FCS_CKM.1.1(3) A password/passphrase shall perform [Password-based Key Derivation Functions] in accordance with a specified cryptographic algorithm as specified in FCS_COP.1(4), with [*10000*] iterations, and output cryptographic key sizes [128] that meet the following [NIST SP 800-132].

FCS_CKM.1.2(3) The TSF shall generate salts using a RBG that meets FCS_RBG_EXT.1 and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM.1.1(3).

5.2.1.3 FCS_CKM.2 Cryptographic Key Establishment

FCS_CKM.2.1 The application shall [implement functionality] to perform cryptographic key establishment in accordance with a specified cryptographic key establishment method: [

- [Elliptic curve-based key establishment schemes] that meet the following: [NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"]

].

5.2.1.4 FCS_CKM_EXT.1 Cryptographic Key Generation Services

FCS_CKM_EXT.1.1 The application shall [

- Implement asymmetric key generation

].

5.2.1.5 FCS_COP.1(1) Cryptographic Operation – Encryption/Decryption

FCS_COP.1.1(1)[1] The application shall perform encryption/decryption in accordance with a specified cryptographic algorithm [

- AES-CBC (as defined in NIST SP 800-38A) mode,
- AES-GCM (as defined in NIST SP 800-38D) mode

] and cryptographic key sizes [128-bit, 256-bit].

5.2.1.6 FCS_COP.1(2) Cryptographic Operation – Hashing

FCS_COP.1.1(2) The application shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [

- SHA-256,
- SHA-384

] and message digest sizes [

- 256,
- 384

] bits that meet the following: FIPS Pub 180-4.

5.2.1.7 FCS_COP.1(3) Cryptographic Operation – Signing

FCS_COP.1.1(3) The application shall perform cryptographic signature services (generation and verification) in accordance with a specified cryptographic algorithm [

- RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 4

].

5.2.1.8 FCS_COP.1(4) Cryptographic Operation – Keyed-Hash Message Authentication

FCS_COP.1.1(4) The application shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm

- HMAC-SHA-256

and [

- SHA-384

] with key sizes [*256 bits, 384 bits*] and message digest sizes 256 and [384] bits that meet the following: FIPS Pub 198-1 *The Keyed-Hash Message Authentication Code* and FIPS Pub 180-4 *Secure Hash Standard*.

[1] This SFR is modified by TD0543 but this ST does not claim any of the selections that were added by the TD.

5.2.1.9 FCS_HTTPS_EXT.1/Server HTTPS Protocol[2]

FCS_HTTPS_EXT.1.1/Server The application shall implement the HTTPS protocol that complies with RFC 2818.

FCS_HTTPS_EXT.1.2/Server The application shall implement HTTPS using TLS as defined in the TLS package.

5.2.1.10 FCS_RBG_EXT.1 Random Bit Generation Services

FCS_RBG_EXT.1.1 The application shall [

- <u>implement DRBG functionality</u>

] for its cryptographic operations.

5.2.1.11 FCS_RBG_EXT.2 Random Bit Generation from Application

FCS_RBG_EXT.2.1 The application shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [<u>CTR_DRBG (AES)</u>].

FCS_RBG_EXT.2.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from a platform-based DRBG and [

- <u>no other noise source</u>

] with a minimum of [

- <u>256 bits</u>

] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

5.2.1.12 FCS_STO_EXT.1 Storage of Credentials

FCS_STO_EXT.1.1 The application shall [

- <u>implement functionality to securely store [*Web GUI authentication credentials, PKI certificate passphrases*] according to [FCS_CKM.1(3)]</u>

] to non-volatile memory.

5.2.1.13 FCS_TLS_EXT.1 TLS Protocol (TLS Package)

FCS_TLS_EXT.1.1 The product shall implement [

- <u>TLS as a server</u>

].

[2] As specified in NIAP TD0473.

5.2.1.14 FCS_TLSS_EXT.1 TLS Server Protocol (TLS Package)

FCS_TLSS_EXT.1.1[3] The product shall implement TLS 1.2 (RFC 5246) and [no earlier TLS versions] as a server that supports the cipher suites [

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289]

and also supports functionality for [

- none

].

FCS_TLSS_EXT.1.2 The product shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and [TLS 1.1].

FCS_TLSS_EXT.1.3 The product shall perform key establishment for TLS using [

- ECDHE parameters using elliptic curves [secp256r1, secp384r1] and no other curves

].

5.2.2 User Data Protection (FDP)

5.2.2.1 FDP_DAR_EXT.1(1)Encryption of Sensitive Application Data (by TOE)

FDP_DAR_EXT.1.1(1)[4] The application shall [

- protect sensitive data in accordance with FCS_STO_EXT.1

] in non-volatile memory.

Application Note: *"Sensitive data" includes both the credential data specified in FCS_STO_EXT.1 as well as system scan, network traffic, and log data that is collected from the Operational Environment. This data is not credential data, but it is still protected using the methods specified in FCS_STO_EXT.1. This is because all sensitive data, regardless of whether or not it is credential data, is stored in an encrypted database.*

[3] This SFR is modified by TD0442 but this ST does not claim any of the selections that were added by the TD.

[4] This SFR is modified by TD0486 but this ST does not claim any of the selections that were added by the TD.

5.2.2.2 FDP_DAR_EXT.1(2)Encryption of Sensitive Application Data (by OE)

FDP_DAR_EXT.1.1(2)[5] The application shall [

- leverage platform-provided functionality to encrypt sensitive data

] in non-volatile memory.

Application Note: *The database encryption referenced in FDP_DAR_EXT.1(1) requires a secret key to be stored on the platform. This is considered to be sensitive data and is therefore protected using platform-provided means.*

5.2.2.3 FDP_DEC_EXT.1 Access to Platform Resources

FDP_DEC_EXT.1.1 The application shall restrict its access to [

- network connectivity

].

FDP_DEC_EXT.1.2 The application shall restrict its access to [

- system logs

].

5.2.2.4 FDP_NET_EXT.1 Network Communications

FDP_NET_EXT.1.1 The application shall restrict network communication to [

- User-initiated communication for [
  - *access to Web GUI,*
  - *check for and download of plugin updates*]
- Respond to [
  - *receipt of log data from operational environment,*
  - *retrieval of log data from by Tenable.sc*]
- [*application-initiated network communication for*
  - *LCE: check for and download of plugin updates,*

  ]

].

5.2.3 Security Management (FMT)

5.2.3.1 FMT_CFG_EXT.1 Secure by Default Configuration

FMT_CFG_EXT.1.1 The application shall provide only enough functionality to set new credentials when configured with default credentials or no credentials.

[5] This SFR is modified by TD0486 but this ST does not claim any of the selections that were added by the TD.

FMT_CFG_EXT.1.2 The application shall be configured by default with file permissions which protect the application's binaries and data files from modification by normal unprivileged users.

5.2.3.2 FMT_MEC_EXT.1 Supported Configuration Mechanism

FMT_MEC_EXT.1.1 The application shall [invoke the mechanisms recommended by the platform vendor for storing and setting configuration options].[6]

5.2.3.3 FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions [

- enable/disable the transmission of any information describing the system's hardware, software, or configuration

].

5.2.4 Privacy (FPR)

5.2.4.1 FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information

FPR_ANO_EXT.1.1 The application shall [

- not transmit PII over a network

].

5.2.5 Protection of the TSF (FPT)

5.2.5.1 FPT_AEX_EXT.1 Anti-Exploitation Capabilities

FPT_AEX_EXT.1.1 The application shall not request to map memory at an explicit address except for [*no exceptions*].

FPT_AEX_EXT.1.2 The application shall [

- not allocate any memory region with both write and execute permissions

].

FPT_AEX_EXT.1.3 The application shall be compatible with security features provided by the platform vendor.

FPT_AEX_EXT.1.4 The application shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.

FPT_AEX_EXT.1.5 The application shall be compiled with stack-based buffer overflow protection enabled.

[6] Modified from original App PP definition by TD0437

5.2.5.2 FPT_API_EXT.1 Use of Supported Services and APIs

FPT_API_EXT.1.1 The application shall use only documented platform APIs.

5.2.5.3 FPT_IDV_EXT.1 Software Identification and Versions

FPT_IDV_EXT.1.1 The application shall be versioned with [[*semantic versioning (SemVer)*]].

5.2.5.4 FPT_LIB_EXT.1 Use of Third Party Libraries

FPT_LIB_EXT.1.1 The application shall be packaged with only [*third-party libraries listed in Appendix A.2*].

Application Note: *The TOE uses a large number of third-party libraries so this information has been provided in an Appendix for readability purposes.*

5.2.5.5 FPT_TUD_EXT.1 Integrity for Installation and Update

FPT_TUD_EXT.1.1 The application shall [leverage the platform] to check for updates and patches to the application software.

FPT_TUD_EXT.1.2 The application shall [provide the ability, leverage the platform] to query the current version of the application software.

FPT_TUD_EXT.1.3 The application shall not download, modify, replace, or update its own binary code.

FPT_TUD_EXT.1.4 The application installation package and its updates shall be digitally signed such that its platform can cryptographically verify them prior to installation.

FPT_TUD_EXT.1.5 The application is distributed [as an additional software package to the platform OS].

5.2.5.6 FPT_TUD_EXT.2 Integrity for Installation and Update

FPT_TUD_EXT.2.1 The application shall be distributed using the format of the platform-supported package manager.

FPT_TUD_EXT.2.2 The application shall be packaged such that its removal results in the deletion of all traces of the application, with the exception of configuration settings, output files, and audit/log events.

5.2.6 Trusted Path/Channels (FTP)

5.2.6.1 FTP_DIT_EXT.1(1) Protection of Data in Transit

FTP_DIT_EXT.1.1(1) The application shall [

- encrypt all transmitted [sensitive data] with [HTTPS in accordance with FCS_HTTPS_EXT.1/**Server**, TLS as defined in the TLS Package]

] between itself and another trusted IT product.

Application Note: *This SFR has been iterated because some sensitive data is transmitted using TLS or HTTPS implemented by the TOE, while other sensitive data is transmitted using* *the platform implementation of SSH. Refer to section 6.7 for a list of trusted channels and the protocol used for each.*

5.2.6.2 FTP_DIT_EXT.1(2) Protection of Data in Transit

FTP_DIT_EXT.1.1(2) The application shall [

- invoke platform-provided functionality to encrypt all transmitted sensitive data with [SSH]

] between itself and another trusted IT product.

Application Note: *This SFR has been iterated because some sensitive data is transmitted using TLS or HTTPS implemented by the TOE, while other sensitive data is transmitted using the platform implementation of SSH. Refer to section 6.7 for a list of trusted channels and the protocol used for each.*

5.3 TOE Security Assurance Requirements

The security assurance requirements for the TOE are included by reference to the App PP.

Table 4: Assurance Components

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1 Basic Functional Specification
AGD: Guidance Documentation	AGD_OPE.1 Operational User Guidance
	AGD_PRE.1 Preparative Procedures
ALC: Life-cycle Support	ALC_CMC.1 Labeling of the TOE
	ALC_CMS.1 TOE CM coverage
	ALC_TSU_EXT.1 Timely Security Updates
ATE: Tests	ATE_IND.1 Independent Testing – Conformance
AVA: Vulnerability Assessment	AVA_VAN.1 Vulnerability Survey

As a functional package, the TLS Package does not define its own SARs. The expectation is that all SARs required by the App PP will apply to the entire TOE, including the portions addressed by the TLS Package. Consequently, the evaluation activities specified in the App PP apply to the entire TOE evaluation, including any changes made to them by subsequent NIAP Technical Decisions as summarized in section 1.2 above.

The TLS Package does contain evaluation activities for how to evaluate its SFR claims as part of the evaluation of ASE_TSS.1, AGD_OPE.1, AGD_PRE.1, and ATE_IND.1. All Security Functional Requirements specified by the TLS Package will be evaluated in the manner specified in that package.

6 TOE Summary Specification

This chapter describes the security functions of the TOE:

- Timely Security Updates
- Cryptographic Support
- User Data Protection
- Security Management
- Privacy
- Protection of the TSF
- Trusted Path/Channels

6.1 Timely Security Updates

Tenable supports a timely security update process for the TOE In addition to their own internal research, the product vendor supports disclosure of potential issues using community forums, direct engagement, and the Tenable support channel. For issues where there is a potential security concern, the support channel uses HTTPS for secure disclosure.

When an issue is reported, Tenable will determine its applicability to the product. The length of time needed to make this determination depends on the complexity of the issue and the extent to which it can be reproduced; well-documented issues such as exposure to a published CVE can be made quickly. If found to be a security issue, a patch is released within 30 days. Tenable monitors the third-party components used by the TOE for potential security issues as well. However, an issue with a dependent component may not be addressed if found not to be applicable to the TOE. For example, security issues are frequently found within the PHP image library but Tenable does not install this library as part of the LCE distribution.

Security updates to the TOE are delivered as regular update packages in the same manner as a functional update. This process is described in section 6.6 below.

6.2 Cryptographic Support

The TOE uses cryptography to secure data in transit between itself and its operational environment.

All TOE cryptographic services are implemented by the OpenSSL cryptographic library. The TOE uses OpenSSL 1.1.1d. The cryptographic algorithms supplied by the TOE are NIST-validated. The following table identifies the cryptographic algorithms used by the TSF, the associated standards to which they conform, and the NIST certificates that demonstrate that the claimed conformance has been met.

Table 5: Cryptographic Functions

Functions	Standards	Certificates
FCS_CKM.1(1) Cryptographic Asymmetric Key Generation		
ECC key pair generation (NIST curves P-256, P-384)	FIPS PUB 186-4	CAVP cert # C1600
FCS_CKM.2 Cryptographic Key Establishment		
ECDSA based key establishment	NIST SP 800-56A	CAVP cert # C1601
FCS_COP.1(1) Cryptographic Operation – Encryption/Decryption		

Functions	Standards	Certificates
AES-CBC and AES-GCM (128, 256 bits)	CBC as defined in NIST SP 800-38A GCM as defined in NIST SP 800-38D	CAVP cert # C1600
FCS_COP.1(2) Cryptographic Operation – Hashing		
SHA-256 and SHA-384 (digest sizes 256 and 384 bits)	FIPS PUB 180-4	CAVP cert # C1600
FCS_COP.1(3) Cryptographic Operation – Signing		
RSA (2048-bit or greater)	FIPS PUB 186-4, Section 4	CAVP cert # C1600
FCS_COP.1(4) Cryptographic Operation – Keyed Hash Message Authentication		
HMAC-SHA-256 and SHA-384	FIPS PUB 198-1 FIPS PUB 180-4	CAVP cert # C1600
FCS_RBG_EXT.2 Random Bit Generation from Application		
CTR_DRBG DRBG (256 bits)	NIST SP 800-90A NIST SP 800-57	CAVP cert # C1600

The TOE generates asymmetric keys in support of trusted communications. The TSF generates ECC keys using P-256 and P-384. These keys are generated in support of the ECDHE key establishment schemes that are used for TLS/HTTPS communications. To ensure sufficient key strength, the TOE also implements DRBG functionality for key generation, using the AES-CTR_DRBG. The proprietary Entropy Analysis Report (EAR) describes how the TSF extracts random data from software-based sources to ensure that an amount of entropy that is at least equal to the strength of the generated keys is present (i.e., at least 256 bits when the largest supported keys are generated) when seeding the DRBG for key generation purposes. The TOE relies on the Linux OS platform entropy source. Specifically, random numbers are obtained from the /dev/random pseudo-device. The platform is assumed to provide at least 256 bits of entropy.

The TOE uses TLS 1.2 for server communications. All other TLS versions are rejected by the TOE. The TLS server implementation supports the following TLS cipher suites in the TOE's evaluated configuration:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA256
- TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

All supported ciphersuites use elliptic curves as the method of key establishment. The TSF presents secp256r1 and secp384r1 as the supported values in the Supported Groups extension and uses the same NIST curves for key establishment.

The TOE uses TLS server functionality for communications between the TOE and the environmental Tenable.sc application and from remote administrators to the Web GUI interface. The TOE does not support mutual authentication. The TOE's implementation of HTTPS conforms to RFC 2818. The TSF does not require client authentication so no certificate validation occurs as part of establishing an HTTPS connection.

The TOE also uses OpenSSL to secure credential data at rest. Specifically, the TOE stores the following credentials:

- Web GUI authentication credentials: username and hashed password data for locally-defined users.
- Passphrases for certificate encryption: used to encrypt the TOE's TLS server certificate.

Passphrases for certificate encryption are encrypted by the TOE using PBKDF2 and administrative credentials to the TOE are encrypted using PBKDF2. The TOE uses the DRBG specified in FCS_RBG_EXT.2 to generate salts that contain at least as many entropy bits as the output key length. The TOE's PBKDF2 implementation performs 10,000 iterations and outputs a 128-bit strength key. Password-based derived keys are formed using a 128-bit salt that is randomly generated by the TOE's DRBG. This is input to the PBKDF function along with the password and specified hashing algorithm, which is SHA-512.

The TOE does not maintain a key hierarchy; the TOE's usage of PBKDF is to generate a hash.

The Cryptographic Support security function is designed to satisfy the following security functional requirements:

- FCS_CKM.1(1) – The TOE uses a NIST-validated implementation to generate asymmetric keys in support of TLS communications.
- FCS_CKM.1(3) – The TOE performs password-based key derivation in support of secure storage of credentials.
- FCS_CKM.2 – The TOE performs NIST-validated key establishment in support of TLS communications.
- FCS_CKM_EXT.1 – The TOE implements its own cryptographic functionality.
- FCS_COP.1(1) – The TOE uses a NIST-validated implementation to perform AES encryption and decryption in support of both TLS communications and secure storage of credentials.
- FCS_COP.1(2) – The TOE uses a NIST-validated implementation to perform cryptographic hashing in support of TLS communications.
- FCS_COP.1(3) – The TOE uses a NIST-validated implementation to generate and verify RSA digital signatures in support of TLS communications.
- FCS_COP.1(4) – The TOE uses a NIST-validated implementation to perform HMAC functions in support of TLS communications and the pseudo-random function used for password-based key derivation.
- FCS_HTTPS_EXT.1/Server – The TOE implements HTTPS as a server to secure data in transit.
- FCS_RBG_EXT.1 – The TOE implements its own random bit generation services.
- FCS_RBG_EXT.2 – The TOE uses a NIST-validated implementation to generate pseudo-random bits and this implementation is seeded with sufficiently strong entropy collected from the operational environment.
- FCS_STO_EXT.1 – The TOE uses its own cryptographic functions to secure credential data at rest.
- FCS_TLS_EXT.1 – The TOE implements TLS to secure data in transit.
- FCS_TLSS_EXT.1 – The TOE implements TLS as a server.

6.3 User Data Protection

The App PP defines ‘sensitive data’ as follows: “Sensitive data may include all user or enterprise data or may be specific application data such as emails, messaging, documents, calendar items, and contacts. Sensitive data must minimally include PII, credentials, and keys. Sensitive data shall be identified in the application’s TSS by the ST author.”

The table below lists the data that is considered to be ‘sensitive data’ for this TOE along with where that data resides.

Table 6: Sensitive Data

Sensitive Data	Exchange	Protection at Rest	Protection in Transit
GUI credentials	Admin’s browser to Web Server over browser connection	FCS_STO_EXT.1 (PBKDF)	HTTPS
Passphrase for PKI certificate encryption	None	FCS_STO_EXT.1 (PBKDF)	N/A
Collected log data	LCE to Tenable.sc	FCS_STO_EXT.1 (AES)	TLS (TSF)/SSH (environment)

All sensitive data that is not credential data is protected by the platform full disk encryption method speicified in FDP_DAR_EXT.1(2).

The underlying platform functionality that the TOE interacts with include network connectivity and system logs. The TOE uses network connectivity for remote management, connections to environmental components, and collection of log data from remote systems. The TOE accesses system log data from target systems for data collection.

The TOE uses environmental network capabilities in various ways. All communications between the TOE and the environmental Tenable.sc component are encrypted, as is remote administrative access. The following table highlights the TOE’s network usage.

Table 7: TOE Network Usage

Component	User-Initiated	Externally-Initiated	TOE-Initiated
LCE	Access to Web GUI	Receipt of log data from operational environment	Check for and download of plugin updates (from Tenable server)
	Check for and download of plugin updates	Retrieval of log data by Tenable.sc	

The User Data Protection security function is designed to satisfy the following security functional requirements:

- FDP_DAR_EXT.1(1) – Sensitive credential data at rest is protected by the TOE’s implementation of PBKDF.
- FDP_DAR_EXT.1(2) – Sensitive data at rest is protected in turn by the platform’s use of full disk encryption.
- FDP_DEC_EXT.1 – The TOE’s use of platform services is well understood by users prior to authorizing the TOE activity.

- FDP_NET_EXT.1 – The TOE communicates over the network for well-defined purposes. Depending on the function, the use of network resources is user-initiated directly through the TSF, remotely initiated by a user performing an action in the operational environment, or initiated by the TOE itself.

6.4 Security Management

The TOE provides a web-based graphical user interface (GUI) that requires user authentication to access. The TOE has a default administrator credential of admin/admin that must be changed on first use. Administrator credentials are stored locally and protected by the TSF as per FCS_STO_EXT.1. Following the initial installation, additional accounts can be created.

During general operations, an administrator will typically interact with the TOE only through the environmental instance of Tenable.sc. However, Tenable.sc does not include the ability to directly modify the initial configuration settings of the TOE.

The TOE is installed into /opt/lce.

All directories containing TOE software and data are configured by default in such a manner that nothing is world-writable. Configuration settings that affect the TOE's interaction with the host OS platform are stored in /etc.

The TOE supports the following security-relevant management functions:

- Configuration of transmission of system's hardware, software, or configuration information
  - Used to configure collection of system and application log data

The Security Management security function is designed to satisfy the following security functional requirements:

- FMT_CFG_EXT.1 – The TOE requires credentials to be defined before administrative use. The TOE is protected from direct modification by untrusted users via its host OS platform.
- FMT_MEC_EXT.1 – Configuration settings for the TOE are stored in an appropriate location in its host OS platform.
- FMT_SMF.1 – Administrators can use the TSF to configure the collection of system and network data from the TOE's operational environment.

6.5 Privacy

The TOE's primary function is to examine organizational assets for configuration or operational states that may indicate the presence of a vulnerability or misuse of organizational resources. To this end, the TOE collects system and application log data and tranmits it to the environmental Tenable.sc application for aggregation, analysis, and reporting. The TOE is not responsible for the collection or transmission of PII. The TOE accepts administrative credentials as part of the GUI login process but user account information is not considered to be PII.

The Privacy security function is designed to satisfy the following security functional requirements:

- FPR_ANO_EXT.1 – The TOE prevents the unnoticed/unauthorized transmission of PII across a network by not having functionality that is intended for such transmissions.

6.6 Protection of the TSF

The TOE implements several mechanisms to protect against exploitation. The TOE implements address space layout randomization (ASLR) through the use of the –fPIC GCC compiler flag and relies fully on its underlying host platforms to perform memory mapping. The TOE also does not use both PROT_WRITE and PROT_EXEC on the same memory regions. There is no situation where the TSF maps memory to an explicit address. The TOE is written in C/C++. It is compiled with stack overflow protection through the use of the –fstack-protector-strong GCC compiler flag. The TOE has a web-based front-end, based on JavaScript. This is interpreted code to which compilation instructions do not apply.

The TOE is designed to run on a host OS platform where SElinux is enabled and enforcing. The TOE uses only documented platform APIs. Appendix A.1 lists the APIs used by the TOE. The TOE also makes use of third-party libraries. Appendix A.2 lists the libraries used by the TOE. The TOE is versioned using semver (Semantic Versioning) in the format x.y(.z) where x is the major version, y is the minor version, and the optional z is the patch version; SWID is not used. The TOE is a standalone application that is not natively bundled as part of a host OS.

The TOE can identify its current running versions through both platform and TSF-mediated methods. The TOE is installed as an RPM and will identify its version in RPM itself. The TOE will also return its version information if its binary is invoked with the –v flag on the OS platform. Administrators can also check the version of the TOE via the Health and Status section on the web GUI.

The TOE can leverage its OS platform to check for software updates and acquire them if they are available. In this case, candidate updates are obtained by the administrator downloading them directly from Tenable’s website or through a package manager such as yum. The TOE will not download, modify, replace, or update its own binary code. The TOE is packaged as an .rpm file. This is digitally signed by Tenable using 4096-bit RSA. Removing (uninstalling) the product will remove all executable code from the host system.

The Protection of the TSF security function is designed to satisfy the following security functional requirements:

- FPT_AEX_EXT.1 – The TOE interacts with its host OS platform in a manner that does not expose the system to memory-related exploitation.
- FPT_API_EXT.1 – The TOE uses documented platform APIs.
- FPT_IDV_EXT. 1 – The TOE is versioned using semver.
- FPT_LIB_EXT.1 – The set of third-party libraries used by the TOE is well-defined.
- FPT_TUD_EXT.1 – There is a well-defined method for checking what version of the TOE is currently installed and whether updates to it are available. Updates are signed by the vendor and validated by the host OS platform prior to installation.
- FPT_TUD_EXT.2 – The TOE can be updated through installation packages.

6.7 Trusted Path/Channels

In the evaluated configuration, the TOE uses both its own cryptographic implementation and its host OS platform to encrypt sensitive data in transit. Listed below are the various external interfaces to the TOE that rely on trusted communications.

Between TOE and operational environment:

- Between administrator and TOE Web GUI (FTP_DIT_EXT.1(1))
    - Communications use TLS/HTTPS (TOE is server)
    - Configurable TCP port, 8836 is default
    - Used to secure administrator interactions with the TOE

Between TOE and environmental Tenable components:

- Between Tenable.sc and TOE (FTP_DIT_EXT.1(1))
    - Communications use TLS (Tenable.sc is client and TOE is server)
    - TCP port 1243
    - Used by Tenable.sc to aggregate unaltered bulk log data collected by the TOE
- Between Tenable.sc and TOE (FTP_DIT_EXT.1(2))
    - Communications use SSH (implemented by the operational environment)
    - TCP port 22
    - Used by Tenable.sc to collect log data that has already been parsed by the TOE as potential vulnerabilities

Note that remote acquisition of log data from the operational environment is not necessarily captured in an encrypted format because the TOE captures this data in the native formats used by the target systems and networks.

All use of SSH is accomplished through TSF invocation of the RHEL `ssh` utility.

The Trusted Path/Channels security function is designed to satisfy the following security functional requirements:

- FTP_DIT_EXT.1(1) – The TOE relies on its own mechanisms to secure some data in transit between itself and its operational environment.
- FTP_DIT_EXT.1(2) – The TOE relies on environmental protection to secure some data in transit between itself and its operational environment.

7 Protection Profile Claims

This ST is conformant to the *Protection Profile for Application Software, Version 1.3, 1 March 2019* (App PP) and *Functional Package for Transport Layer Security (TLS), Version 1.1, February 12,* 2019 (TLS Package) along with all applicable errata and interpretations from the certificate issuing scheme.

The TOE consists of a software application that runs on a Linux operating system as its platform.

As explained in section 3, Security Problem Definition, the Security Problem Definition of the App PP has been included by reference into this ST.

As explained in section 4, Security Objectives, the Security Objectives of the App PP has been included by reference into this ST.

All claimed SFRs are defined in the App PP and TLS Package. All mandatory SFRs are claimed. No optional or objective SFRs are claimed. Selection-based SFR claims are consistent with the selections made in the mandatory SFRs that prompt their inclusion.

8 Rationale

This Security Target includes by reference the App PP Security Problem Definition, Security Objectives, and Security Assurance Requirements. The Security Target does not add, remove, or modify any of these items. Security Functional Requirements have been reproduced with the Protection Profile operations completed. All selections, assignments, and refinements made on the claimed Security Functional Requirements have been performed in a manner that is consistent with what is permitted by the App PP and TLS Package. The proper set of selection-based requirements have been claimed based on the selections made in the mandatory requirements. Consequently, the claims made by this Security Target are sufficient to address the TOE's security problem. Rationale for the sufficiency of the TOE Summary Specification is provided below.

8.1 TOE Summary Specification Rationale

This section in conjunction with Section 0, the

The TLS Package does contain evaluation activities for how to evaluate its SFR claims as part of the evaluation of ASE_TSS.1, AGD_OPE.1, AGD_PRE.1, and ATE_IND.1. All Security Functional Requirements specified by the TLS Package will be evaluated in the manner specified in that package.

TOE Summary Specification, provides evidence that the security functions meet the TOE security requirements. Each description includes rationale indicating which requirements the corresponding security functions satisfy. The combined security functions work together to satisfy all of the security requirements. The security functions described in Section 6 are necessary for the TSF to enforce the required security functionality. Table 8 demonstrates the relationship between security requirements and functions.

Table 8: Security Functions vs. Requirements Mapping

	Cryptographic Support	User Data Protection	Security Management	Privacy	Protection of the TSF	Trusted Path/Channels
FCS_CKM.1(1)	X					
FCS_CKM.1(3)	X					
FCS_CKM.2	X					
FCS_CKM_EXT.1	X					
FCS_COP.1(1)	X					
FCS_COP.1(2)	X					
FCS_COP.1(3)	X					
FCS_COP.1(4)	X					
FCS_HTTPS_EXT.1/Server	X					
FCS_RBG_EXT.1	X					
FCS_RBG_EXT.2	X					
FCS_STO_EXT.1	X					
FCS_TLS_EXT.1	X					
FCS_TLSS_EXT.1	X					
FDP_DAR_EXT.1(1)		X				
FDP_DAR_EXT.1(2)		X				
FDP_DEC_EXT.1		X				
FDP_NET_EXT.1		X				
FMT_CFG_EXT.1			X			
FMT_MEC_EXT.1			X			
FMT_SMF.1			X			
FPR_ANO_EXT.1				X		
FPT_AEX_EXT.1					X	
FPT_API_EXT.1					X	
FPT_IDV_EXT.1					X	

	Cryptographic Support	User Data Protection	Security Management	Privacy	Protection of the TSF	Trusted Path/Channels
FPT_LIB_EXT.1					X	
FPT_TUD_EXT.1					X	
FPT_TUD_EXT.2					X	
FTP_DIT_EXT.1(1)						X
FTP_DIT_EXT.1(2)						X

Appendix A TOE Usage of Third-Party Components

This Appendix lists the platform APIs and third-party libraries that are used by the TOE.

A.1 Platform APIs

Listed below are the platform APIs used by the TOE. Note that these APIs do not necessarily relate to the TOE functionality claimed in the Security Target; however, since they are bundled with the product itself they are disclosed since a vulnerability in outside the logical boundary of the product could still present an exploitable vulnerability.

A.1.1 Shell Builtins and OS Utilities

., alias, break, caller, cd, compgen, continue, declare, echo, enable, eval, exec, exit, export, false, kill, let, local, printf, pwd, read, readonly, return, set, shift, shopt, [], trap, true, type, typeset, ulimit, umask, unalias, unset, wait, at, awk, basename, bc, cat, chkconfig, chmod, chown, cp, cut, date, dd, df, diff, dirname, du, file, find, getconf, head, hostname, grep, gunzip, gzip, join, last, ldd, ln, ls, md5sum, mkdir, mktemp, mv, netstat, nl, od, pgrep, pkill, ps, readlink, rm, rpm, sed, sleep, sort, stat, strings, su, tac, tail, tar, tee, touch, tput, tr, tty, ulimit, uniq, vmstat, which, java, crontab, groupadd, groupdel, makewhatis, runuser, service, sync, sysctl, systemctl, tar, useradd, userdel

A.1.2 OS Syscalls and Library Functions

IO_getc, __assert_fail, __ctype_b_loc, __ctype_get_mb_cur_max, __ctype_tolower_loc, __ctype_toupper_loc, __cxa_atexit, __duplocale, __environ, __errno_location, __fpclassify, __fprintf_chk, __freelocale, __fxstat, __fxstat64, __isinf, __isinff, __isnanf, __isoc99_sscanf, __isoc99_vsscanf, __iswctype_l, __libc_current_sigrtmax, __libc_current_sigrtmin, __libc_start_main, __lxstat, __lxstat64, __memcpy_chk, __memmove_chk, __memset_chk, __newlocale, __nl_langinfo_l, __open_2, __printf_chk, __pthread_key_create, __rawmemchr, __snprintf_chk, __sprintf_chk, __stpcpy_chk, __strcat_chk, __strcoll_l, __strcpy_chk, __strdup, __strftime_l, __strncpy_chk, __strtod_l, __strtof_l, __strtok_r, __strxfrm_l, __tls_get_addr, __towlower_l, __towupper_l, __uselocale, __vfprintf_chk, __vsnprintf_chk, __wcscoll_l, __wcsftime_l, __wcsxfrm_l, __wctype_l, __xpg_basename, __xpg_strerror_r, __xstat, __xstat64, _exit, _setjmp, abort, accept, access, atof, atoi, backtrace, backtrace_symbols, bcopy, bind, bindtextdomain, bsearch, btowc, bzero, calloc, ceil, ceilf, chdir, chmod, chown, clearerr, clock_gettime, close, closedir, closelog, connect, creat64, ctime_r, dl_iterate_phdr, dladdr, dlclose, dlerror, dlopen, dlsym, dup, dup2, environ, ether_hostton, execl, execve, exit, fchmod, fchown, fclose, fcntl, fdatasync, fdopen, feof, feof_unlocked, ferror, ferror_unlocked, fflush, fflush_unlocked, fgetc, fgets, fgets_unlocked, fileno, fileno_unlocked, flock, floor, fmemopen, fmod, fopen, fopen64, fork, fprintf, fputc, fputc_unlocked, fputs, fputs_unlocked, fread, fread_unlocked, free, freeaddrinfo, freeifaddrs, freopen64, fseek, fseeko64, fsync, ftell, ftello64, ftime, ftruncate64, fwrite, fwrite_unlocked, gai_strerror, getaddrinfo, getc, getcontext, getcwd, getdtablesize, getegid, getenv, geteuid, getgid, gethostbyname, gethostname, getifaddrs, getnameinfo, getnetbyname, getpagesize, getpeername, getpid, getprotobyname, getpwnam_r, getpwuid, getpwuid_r, getresgid, getresuid, getrlimit, getrlimit64, getrusage, getservbyname, getsockname, getsockopt, gettext, gettimeofday, getuid, getwc, gmtime_r, htonl, htons, iconv, iconv_close, iconv_open, if_nametoindex, inet_addr, inet_aton, inet_ntoa, inet_ntop, inet_pton, ioctl, isalnum, isalpha, isatty, isblank, iscntrl, isgraph, islower, isprint, isspace, isupper, kill, ldiv, listen, localeconv, localtime, localtime_r, log10, longjmp, lrand48, lseek, lseek64, madvise, malloc, malloc_trim, mbrtowc, mbsnrtowcs, mbsrtowcs, memchr, memcmp, memcpy, memmem, memmove, memset, mkdir, mkostemp64, mkstemp64, mktime, mmap, mmap64, modf, mprotect, mremap, munmap, nanosleep, nftw64, nice, nl_langinfo, ntohl, ntohs, open, open64, opendir, openlog, pathconf, pclose, perror, pipe, pipe2, poll, popen, posix_fadvise64, posix_spawn, posix_spawn_file_actions_adddup2, posix_spawn_file_actions_destroy, posix_spawn_file_actions_init, posix_spawnattr_destroy, posix_spawnattr_init, posix_spawnattr_setflags, pow, prctl, printf, pthread_attr_destroy, pthread_attr_getschedparam, pthread_attr_getstack, pthread_attr_init, pthread_attr_setdetachstate, pthread_attr_setschedparam, pthread_attr_setstack, pthread_attr_setstacksize, pthread_barrier_wait, pthread_cancel, pthread_cond_broadcast, pthread_cond_destroy, pthread_cond_init, pthread_cond_signal, pthread_cond_timedwait, pthread_cond_wait, pthread_create, pthread_detach, pthread_equal, pthread_exit, pthread_getattr_np, pthread_getspecific, pthread_join, pthread_key_create, pthread_key_delete, pthread_kill, pthread_mutex_destroy, pthread_mutex_init, pthread_mutex_lock, pthread_mutex_timedlock, pthread_mutex_trylock, pthread_mutex_unlock, pthread_mutexattr_destroy, pthread_mutexattr_init, pthread_mutexattr_setpshared, pthread_mutexattr_settype, pthread_once, pthread_rwlock_destroy, pthread_rwlock_init, pthread_rwlock_rdlock, pthread_rwlock_unlock, pthread_rwlock_wrlock, pthread_self, pthread_setcancelstate, pthread_setcanceltype, pthread_setspecific, pthread_sigmask, pthread_testcancel, pthread_tryjoin_np, putc, putchar, puts, putwc, pwrite64, qsort, rand, read, readahead, readdir, readdir64, readdir64_r, readdir_r, readlink, realloc, realpath, recv, recvfrom, recvmmsg, recvmsg, rename, rewind, rewinddir, rmdir, round, select, send, sendto, setegid, seteuid, setgid, setitimer, setlocale, setregid, setreuid, setrlimit, setrlimit64, setsockopt, setuid, setvbuf, shutdown, sigaction, sigaddset, sigdelset, sigemptyset, sigfillset, sigismember, signal, sigpending, sigwait, sleep, snprintf, socket, socketpair, splice, sprintf, srand, srand48, sscanf, statfs64, statvfs64, stderr, stdin, stdout, stpcpy, strcasecmp, strcat, strchr, strcmp, strcpy, strcspn, strdup, strerror, strerror_r, strftime, strlen, strncasecmp, strncat, strncmp, strncpy, strnlen, strptime, strrchr, strsignal, strspn, strstr, strtod, strtof, strtok, strtok_r, strtol, strtold_l, strtoll, strtoul, sync_file_range, syscall, sysconf, syslog, system, tcgetattr, tcsetattr, textdomain, time, tmpfile64, tolower, toupper, trunc, uname, ungetc, ungetwc, unlink, usleep, utime, utimes, versionsort64, vfprintf, vsnprintf, waitpid, wcrtomb, wcscmp, wcslen, wcsnrtombs, wctob, wmemchr, wmemcmp, wmemcpy, wmemmove, wmemset, write, writev

A.2 Third-Party Libraries

Listed below are the third-party libraries used by the TOE.

BigInteger.js, c-ares, chosen, farmhash, Flatiron Director, Fontawesome, handlebars, jQuery, jQuery Cookie, jQuery DataTables, jQuery FileUpload, jQuery HotKeys, jQuery Migrate, jQuery scroll.To, jQuery-Storage-API, jQuery TableSorter, jquery.fn sortElements, jQuery tipsy, jQuery UI, jQuery Validation, JSONSL, Keepalived, libcurl, libpcap, libxml2, locache, moment, OpenBSM, OpenSSL, PCRE / libpcre, requirejs, session.js, sugar, underscore, WMI Sample Client (wmic.c), xml2json, zlib