Nessus Agent 10.4.1

Security Target

Version 1.1

28 June 2023

Prepared for:

Tenable, Inc.
6100 Merriweather Drive
12th Floor
Columbia, MD 21044

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 7)
- This ST is conformant to the *Protection Profile for Application Software*, Version 1.4, 7 October 2021 ([APP_PP]) and *Functional Package for Transport Layer Security (TLS)*, Version 1.1, 1 March 2019 ([TLS_PKG]) along with all applicable errata and interpretations from the certificate issuing scheme.

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

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

As explained in section 4, Security Objectives, the Security Objectives of [APP_PP] have been included by reference into this ST.

All claimed SFRs are defined in [APP_PP] and [TLS_PKG]. 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 –Nessus Agent 10.4.1 Security Target

ST Version – Version 1.1

ST Date – 28 June 2023

TOE Identification – Nessus Agent 10.4.1, supported on RHEL 8.7 and Windows Server 2019

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 1.4, 7 October 2021 ([APP_PP]) with the following optional and selection-based SFRs:
  - FCS_CKM.1/AK
  - FCS_CKM.2
  - FCS_COP.1/SKC
  - FCS_COP.1/Hash
  - FCS_COP.1/KeyedHash
  - FCS_COP.1/Sig
  - FCS_HTTPS_EXT.1/Client
  - FCS_RBG_EXT.2
  - FIA_X509_EXT.1
  - FIA_X509_EXT.2
  - FPT_TUD_EXT.2
- *Functional Package for Transport Layer Security (TLS)*, Version 1.1, 1 March 2019 ([TLS_PKG]) with the following optional and selection-based SFRs:
  - FCS_TLSC_EXT.1
  - FCS_TLSC_EXT.5
- The following NIAP Technical Decisions affecting [APP_PP] 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 for the reasons stated:

  TD0624: Addition of DataStore for Storing and Setting Configuration Options

  - N/A; the TOE is not evaluated on an Android platform.

  TD0628: Addition of Container Image to Package Format

- The ST accounts for this TD.

TD0650: Conformance claim sections updated to allow for MOD_VPNC_V2.3 and 2.4

- No change to ST; affects only PP conformance claims and the ST does not claim conformance to the relevant PP-Module.

TD0664: Testing activity for FPT_TUD_EXT.2.2

- No change to ST; affects only evaluation activities.

TD0669: FIA_X509_EXT.1 Test 4 Interpretation

- No change to ST; affects only evaluation activities.

TD0717: Format changes for PP_APP_V1.4

- The ST accounts for this TD.

TD0719: ECD for PP APP V1.3 and 1.4

- No change to ST; affects only PP conformance to CC.

TD0736: Number of elements for iterations of FCS_HTTPS_EXT.1

- N/A; the ST does not claim FCS_HTTPS_EXT.1/Server.

TD0743: FTP_DIT_EXT.1.1 Selection exclusivity

- The ST accounts for this TD.

- The following NIAP Technical Decisions affecting [TLS_PKG] 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 for the reasons stated:

TD0442: Updated TLS Ciphersuites for TLS Package

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

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

- N/A; the TOE does not claim TLS server functionality.

TD0499: Testing with pinned certificates

- No change to ST; affects only evaluation activities. Additionally, the TOE does not support pinned certificates.

TD0513: CA Certificate loading

- No change to ST; affects only evaluation activities.

TD0588: Session Resumption Support in TLS package

- N/A; the TOE does not claim TLS server functionality.

TD0726: Corrections to (D)TLSS SFRs in TLS 1.1 FP

  - N/A; the TOE does not claim TLS server functionality.

  TD0739: PKG_TLS_V1.1 has 2 different publication dates

  - N/A; the TOE does not implement RSA-based key establishment methods.

- 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. The [APP_PP] indicates iterated SFRs using a short descriptive label placed at the end of the component. For example, [APP_PP] iterates FCS_COP.1 for different cryptographic operations (FCS_COP.1/SKC for symmetric key cryptography, FCS_COP.1/Hash for cryptographic hashing, etc.).
  - 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., [<u>selection item, [*assignment item inside selection*]</u>]).
  - Selection: allows the specification of one or more elements from a list. Selections are indicated using underlines and are surrounded by brackets (e.g., [<u>selection item</u>]).
  - 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 are used in this ST:

Table 1: Terms and Definitions

Term	Definition
Nessus Agent	The TOE; an application that is installed on an endpoint system to collect details about that system's configuration and behavior.
Nessus	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.
Scan	The process by which the TOE actively collects data from a target system.
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 Abbreviations and Acronyms

Table 2: Abbreviations and Acronyms

Term	Definition
AES	Advanced Encryption Standard
ASLR	Address Space Layout Randomization
CA	Certificate Authority
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
CN	Common Name
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
FQDN	Fully Qualified Domain Name
GB	Gigabyte
GCM	Galois/Counter Mode
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

Term	Definition
NNM	Nessus Network Monitor
OCSP	Online Certificate Status Protocol
OE	Operational Environment
OID	Original Issue Document
OS	Operating System
PII	Personally Identifiable Information
PKI	Public Key Infrastructure
PP	Protection Profile
RAM	Random Access Memory
RHEL	Red Hat Enterprise Linux
RSA	Rivest, Shamir and Adleman (algorithm for public-key cryptography)
SAN	Subject Alternative Name
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
ST	Security Target
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TOE	Target of Evaluation
TSF	TOE Security Function

2 Product and TOE Description

2.1 Introduction

Nessus Agent 10.4.1 (Nessus Agent) is a software product designed to be installed on an endpoint system to facilitate local scanning of that system. Local scanning allows Nessus Agent to collect detailed information about the system's hardware, software, and configuration, which can be used to determine compliance with organizational security policies and whether potential exploitable vulnerabilities are present on that system.

Nessus Agent is deployed and configured by an environmental instance of Nessus, which also collects scan results from Nessus Agent for aggregation and analysis. Nessus in turn will transmit this data to an environmental instance of Tenable.sc (SecurityCenter), where it can be combined with network traffic and system log data to provide a comprehensive window into the security posture of an organization.

The TOE conforms to [APP_PP] and [TLS_PKG]. 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

Nessus Agent is a vulnerability management product that is designed to provide visibility into system assets. The product is used to collect data from endpoint systems in an organization (e.g. Windows and Linux workstations). Nessus Agent complements the functionality of Nessus, which can perform remote scanning of these systems, by collecting additional data that requires local access to obtain. Nessus is responsible for the distribution and configuration of the various instances of Nessus Agent that may be present in an organization's environment. Nessus Agent will transmit its completed scan results to the environmental instance of Nessus that configured it. This data can be used by Nessus and by Tenable.sc to analyze the potential vulnerabilities of specific systems or to diagnose patterns that are applicable to multiple systems.

Nessus Agent 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 Nessus Agent consists of the mandatory functionality prescribed by [APP_PP] and [TLS_PKG], as well as some selection-based functionality where needed.

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

- **Protection of sensitive data at rest:** the TOE uses encryption to protect sensitive data.
- **Protection of data in transit:** the TOE secures data in transit between itself and its operational environment using TLS and HTTPS.
- **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.

- **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 scanning and data collection capabilities are outside the scope of the TOE (aside from the trusted channel used to transmit the collected data), as is any other product behavior that is not described in [APP_PP] or [TLS_PKG]. 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 Nessus Agent TOE consists of the Nessus Agent application, which is a C/C++ application. The TOE has both Linux and Windows platform versions.

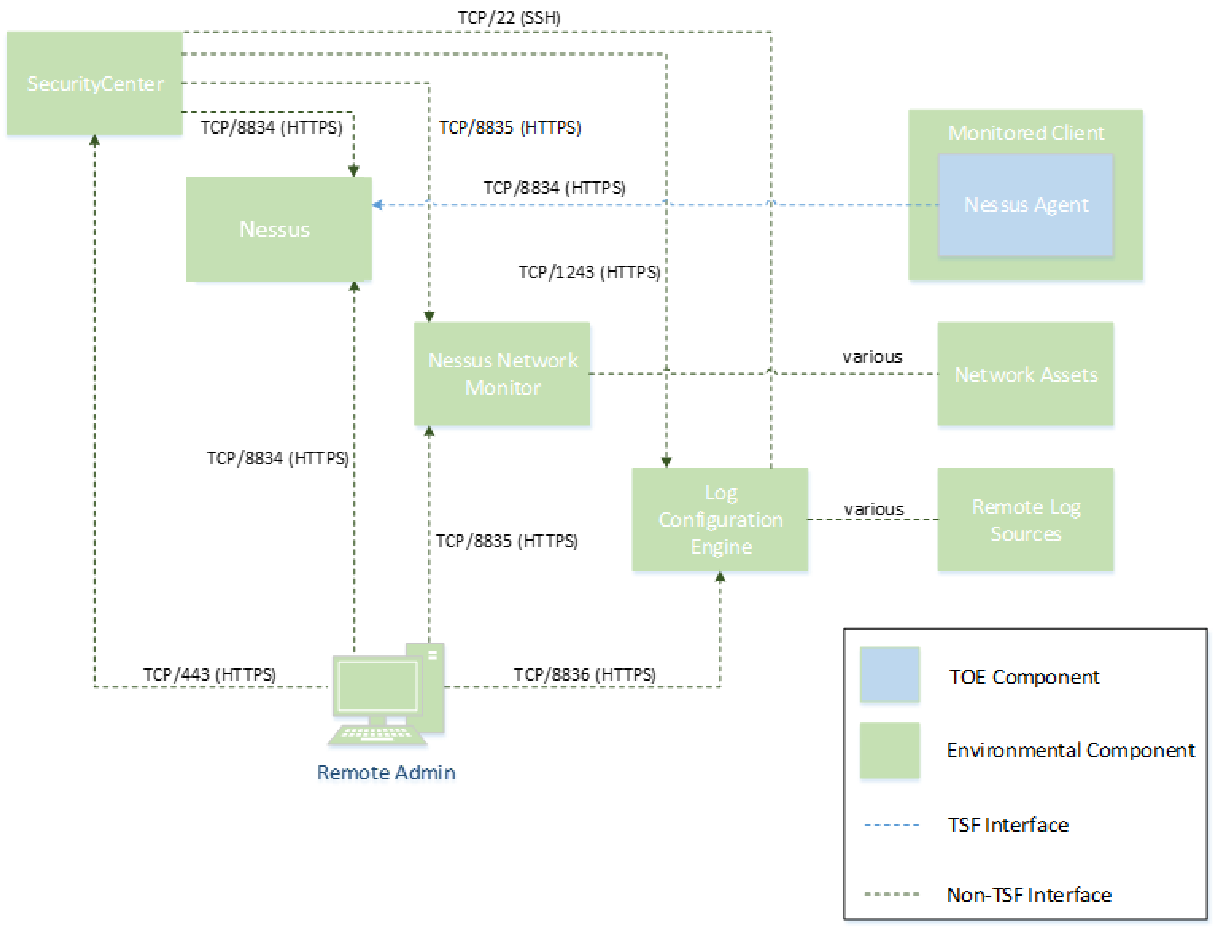
2.4.1 Physical Boundary

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

- Nessus Agent 10.4.1.

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 Nessus Agent. However, it is expected that in a typical deployment, many instances of the Nessus Agent application will be deployed on organizational systems. This does not affect the security claims made by the TOE because there is no direct interface from one instance of Nessus Agent to another.

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

- 1x dual-core CPU (any dual-core CPU's clock speed is sufficient)
- 1 GB RAM
- 2 GB disk storage
- 15-50 IOPS disk speed.

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 port must be open for the TOE to function, but the specific port is configurable if the default port cannot be used:

- TCP/8834 (for communications with Nessus).

The TOE's operational environment includes the following:

- Other Tenable components (i.e., an instance of Nessus; Tenable.sc, Nessus Network Monitor, and Log Correlation Engine are expected to be present in the TOE's operational environment because they all contribute to the same overall capability, but the TOE does not interact with these other applications directly).
- Platform (hardware and software) on which the TOE is hosted.
    - The TOE is capable of running on a general-purpose Windows or Linux operating system on standard consumer-grade hardware on either a physical or virtual machine. For the evaluated configuration, the TOE was tested on virtualized instances of Windows Server 2019 and RHEL 8.7, each 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.

2.4.2 Logical Boundary

This section summarizes the security functions provided by the TOE:

- Timely Security Updates
- Cryptographic Support
- User Data Protection
- Identification and Authentication
- 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 in transit. The TOE does not store credential data on the local system so no separate data at rest protection mechanism is implemented.

For data in transit, the TOE implements TLS/HTTPS as a client to communicate with an instance of Nessus in the operational environment. The TOE's TLS client does not support mutual authentication.

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

2.4.2.3 User Data Protection

The TOE is compatible with the use of platform full disk encryption to protect sensitive data at rest.

The TOE relies on the network connectivity and system log capabilities of its host OS platform. The TOE supports application-initiated uses of the network. The TOE also accesses various system resources as part of conducting system scans. Specifically, the TOE supports local scanning of the system that it is installed on.

2.4.2.4 Identification and Authentication

The TOE supports X.509 certificate validation as part of establishing TLS and HTTPS connections. The TOE supports various certificate validity checking methods and can also check certificate revocation status using OCSP. If the validity status of a certificate cannot be determined, the certificate will be accepted. All other cases where a certificate is found to be invalid will result in rejection without an administrative override.

2.4.2.5 Security Management

Both the TOE binary components themselves and the configuration settings they use are stored in locations recommended by the platform vendors.

The TOE does not include a direct user interface to manage its functionality. Security-relevant configuration of the TOE is initiated from the Nessus application in the TOE's operational environment. This configuration relates to the circumstances under which the TOE will transmit data about the local system's hardware, software, and configuration information (i.e., scan results) back to its operational environment.

2.4.2.6 Privacy

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

2.4.2.7 Protection of the TSF

The TOE enforces various mechanisms to prevent itself from being used as an attack vector to its host OS platform. Each TOE platform version (Windows and Linux) 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.

Each TOE platform version 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 or through its connection with the environmental Nessus application. The format of the software update is dependent on the TOE platform version. All updates are digitally signed to guarantee their authenticity and integrity.

2.4.2.8 Trusted Path/Channels

The TOE encrypts sensitive data in transit between itself and its operational environment using TLS and HTTPS.

2.5 TOE Documentation

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

- Nessus Agent 10.4.x User Guide, Last Updated: May 19, 2023.

3 Security Problem Definition

This ST includes by reference the Security Problem Definition, composed of threats and assumptions, from [APP_PP]. The Common Criteria also provides for organizational security policies to be part of a security problem definition, but no such policies are defined in [APP_PP].

As a functional package, [TLS_PKG] 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 [APP_PP].

In general, the threat model of [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

As with the Security Problem Definition, this ST includes by reference the security objectives defined in [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, [TLS_PKG] does not contain a statement of Security Objectives. The TOE's use of TLS is intended to satisfy the O.PROTECTED_COMMS objective of [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 1.4, 7 October 2021
- *Functional Packages for Transport Layer Security (TLS),* Version 1.1, 1 March 2019

As a result, any selection, assignment, or refinement operations already performed in these documents 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 [APP_PP] and [TLS_PKG]. These documents define the following extended SAR and extended SFRs; since they have not been redefined in this ST, [APP_PP] and [TLS_PKG] 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_HTTPS_EXT.1/Client 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
- FDP_DAR_EXT.1 Encryption of Sensitive Application Data
- FDP_DEC_EXT.1 Access to Platform Resources
- FDP_NET_EXT.1 Network Communications
- FIA_X509_EXT.1 X.509 Certificate Validation
- FIA_X509_EXT.2 X.509 Certificate Authentication
- 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

Defined in [TLS_PKG]:

- FCS_TLS_EXT.1 TLS Protocol
- FCS_TLSC_EXT.1 TLS Client Protocol
- FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension.

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_EXT.1 Cryptographic Key Generation Services
	FCS_CKM.1/AK Cryptographic Asymmetric Key Generation
	FCS_CKM.2 Cryptographic Key Establishment
	FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption
	FCS_COP.1/Hash Cryptographic Operation – Hashing
	FCS_COP.1/KeyedHash Cryptographic Operation – Keyed-Hash Message Authentication
	FCS_COP.1/Sig Cryptographic Operation – Signing
	FCS_HTTPS_EXT.1/Client 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_TLSC_EXT.1 TLS Client Protocol (TLS Package)
	FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension (TLS Package)
FDP: User Data Protection	FDP_DAR_EXT.1 Encryption of Sensitive Application Data
	FDP_DEC_EXT.1 Access to Platform Resources
	FDP_NET_EXT.1 Network Communications
FIA: Identification and authentication	FIA_X509_EXT.1 X.509 Certificate Validation
	FIA_X509_EXT.2 X.509 Certificate Authentication
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 Protection of Data in Transit

5.2.1 Cryptographic Support (FCS)

5.2.1.1 FCS_CKM_EXT.1 Cryptographic Key Generation Services

FCS_CKM_EXT.1.1[1] The application shall [

- Implement asymmetric key generation

].

5.2.1.2 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation

FCS_CKM.1.1/AK[2] 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-384 and [P-256, P-521]] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4]

].

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 meets the following: [NIST Special Publication 800-56A, "Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography"]

].

5.2.1.4 FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption

FCS_COP.1.1/SKC[3] 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].

[1] Modified in accordance with TD0717.

[2] Modified in accordance with TD0717.

[3] Modified in accordance with TD0717.

5.2.1.5 FCS_COP.1/Hash Cryptographic Operation – Hashing

FCS_COP.1.1/Hash[4] 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].

5.2.1.6 FCS_COP.1/Sig Cryptographic Operation – Signing

FCS_COP.1.1/Sig[5] 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 5]

].

5.2.1.7 FCS_COP.1/KeyedHash Cryptographic Operation – Keyed-Hash Message Authentication

FCS_COP.1.1/KeyedHash[6] The application shall perform [keyed-hash message authentication] in accordance with a specified cryptographic algorithm [

- HMAC-SHA-256,
- HMAC-SHA-384]

and [

- no other algorithms

] with key sizes [*256 bits, 384 bits*] and message digest sizes [256, 384] and [no other size] bits that meet the following: [FIPS Pub 198-1, '*The Keyed-Hash Message Authentication Code'* and FIPS Pub 180-4, '*Secure Hash Standard'*].

5.2.1.8 FCS_HTTPS_EXT.1/Client HTTPS Protocol

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

FCS_HTTPS_EXT.1.2/Client The application shall implement HTTPS using TLS as defined in the Functional Package for TLS.

FCS_HTTPS_EXT.1.3/Client The application shall [not establish the application-initiated connection] if the peer certificate is deemed invalid.

[4] Modified in accordance with TD0717.

[5] Modified in accordance with TD0717.

[6] Modified in accordance with TD0717.

5.2.1.9 FCS_RBG_EXT.1 Random Bit Generation Services

FCS_RBG_EXT.1.1 The application shall [

- implement DRBG functionality

] for its cryptographic operations.

5.2.1.10 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 [CTR_DRBG (AES)].

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

- no other noise source

] with a minimum of [

- 256 bits

] 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.

5.2.1.11 FCS_STO_EXT.1 Storage of Credentials

FCS_STO_EXT.1.1 The application shall [

- not store any credentials

] to non-volatile memory.

5.2.1.12 FCS_TLS_EXT.1 TLS Protocol (TLS Package)

FCS_TLS_EXT.1.1 The product shall implement [

- TLS as a client

].

5.2.1.13 FCS_TLSC_EXT.1 TLS Client Protocol (TLS Package)

FCS_TLSC_EXT.1.1[7] The product shall implement TLS 1.2 (RFC 5246) and [no earlier TLS versions] as a client 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_TLSC_EXT.1.2 The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

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

FCS_TLSC_EXT.1.3 The product shall not establish a trusted channel if the server certificate is invalid [

- <u>with no exceptions</u>

].

5.2.1.14 FCS_TLSC_EXT.5 TLS Client Support for Supported Groups Extension (TLS Package)

FCS_TLSC_EXT.5.1 The product shall present the Supported Groups Extension in the Client Hello with the supported groups [

- <u>secp256r1,</u>
- <u>secp384r1,</u>
- <u>secp521r1</u>

].

5.2.2 User Data Protection (FDP)

5.2.2.1 FDP_DAR_EXT.1 Encryption of Sensitive Application Data

FDP_DAR_EXT.1.1 The application shall [

- <u>leverage platform-provided functionality to encrypt sensitive data</u>

] in non-volatile memory.

5.2.2.2 FDP_DEC_EXT.1 Access to Platform Resources

FDP_DEC_EXT.1.1 The application shall restrict its access to [

- <u>network connectivity</u>

].

FDP_DEC_EXT.1.2 The application shall restrict its access to [

- <u>[*system configuration*]</u>

].

5.2.2.3 FDP_NET_EXT.1 Network Communications

FDP_NET_EXT.1.1 The application shall restrict network communication to [

- <u>[*application-initiated network communication for*</u>
  - <u>*transmission of scan results to Nessus,*</u>
  - <u>*periodic polling of Nessus to determine if scan needs to be initiated or if binary/plugin updates need to be acquired,*</u>
  - <u>*check for and download of binary and plugin updates from Nessus*</u>

  ]

].

5.2.3 Identification and Authentication (FIA)

5.2.3.1 FIA_X509_EXT.1 X.509 Certificate Validation

FIA_X509_EXT.1.1 The application shall [<u>implement functionality</u>] to validate certificates in accordance with the following rules:

- RFC 5280 certificate validation and certificate path validation.
- The certificate path must terminate with a trusted CA certificate.

- The application shall validate a certificate path by ensuring the presence of the basicConstraints extension and that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- The application shall validate that any CA certificate includes caSigning purpose in the key usage field.
- The application shall validate the revocation status of the certificate using [OCSP as specified in RFC 6960].
- The application shall validate the extendedKeyUsage (EKU) field according to the following rules:
    - Certificates used for trusted updates and executable code integrity verification shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3) in the extendedKeyUsage field.
    - Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the EKU field.
    - Client certificates presented for TLS shall have the Client Authentication purpose (id-kp 2 with OID 1.3.6.1.5.5.7.3.2) in the EKU field.
    - S/MIME certificates presented for email encryption and signature shall have the Email Protection purpose (id-kp 4 with OID 1.3.6.1.5.5.7.3.4) in the EKU field.
    - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.
    - Server certificates presented for EST shall have the CMC Registration Authority (RA) purpose (id-kp-cmcRA with OID 1.3.6.1.5.5.7.3.28) in the EKU field.

FIA_X509_EXT.1.2 The application shall treat a certificate as a CA certificate only if the basicConstraints extension is present and the CA flag is set to TRUE.

5.2.3.2 FIA_X509_EXT.2 X.509 Certificate Authentication

FIA_X509_EXT.2.1 The application shall use X.509v3 certificates as defined by RFC 5280 to support authentication for [HTTPS, TLS].

FIA_X509_EXT.2.2 When the application cannot establish a connection to determine the validity of a certificate, the application shall [accept the certificate].

5.2.4 Security Management (FMT)

5.2.4.1 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.

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

5.2.4.2 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].

5.2.4.3 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

].

5.2.5 Privacy (FPR)

5.2.5.1 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

].

5.2.6 Protection of the TSF (FPT)

5.2.6.1 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 built with stack-based buffer overflow protection enabled.

5.2.6.2 FPT_API_EXT.1 Use of Supported Services and APIs

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

5.2.6.3 FPT_IDV_EXT.1 Software Identification and Versions

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

5.2.6.4 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.*

5.2.6.5 FPT_TUD_EXT.1 Integrity for Installation and Update

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

Application Note: *The TOE may leverage the platform specifically to check for updates when communicating with the environmental Nessus application.*

FPT_TUD_EXT.1.2 The application shall [<u>provide the ability, leverage the platform</u>] 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 Application updates shall be digitally signed such that the application platform can cryptographically verify them prior to installation.

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

5.2.6.6 FPT_TUD_EXT.2 Integrity for Installation and Update

FPT_TUD_EXT.2.1[8] The application shall be distributed using [<u>the format of the platform-supported package manager</u>].

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.

FPT_TUD_EXT.2.3 The application installation package shall be digitally signed such that its platform can cryptographically verify them prior to installation.

5.2.7 Trusted Path/Channels (FTP)

5.2.7.1 FTP_DIT_EXT.1 Protection of Data in Transit

FTP_DIT_EXT.1.1[9] The application shall [

- <u>encrypt all transmitted [sensitive data] with [</u>
  - <u>HTTPS as a client in accordance with FCS_HTTPS_EXT.1/Client for [*transmission of scan results to Nessus*]</u>
  - <u>TLS as a client as defined in the Functional Package for TLS for [*transmission of scan results to Nessus*]]</u>

] between itself and another trusted IT product.

5.3 TOE Security Assurance Requirements

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

[8] Modified in accordance with TD0628.

[9] Modified in accordance with TD0743.

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
- Identification and Authentication
- 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, the issue may be in a library that is part of a larger package that the TOE uses, but the TOE does not include that specific library.

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 6.7 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 3.0.9. 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/AK Cryptographic Asymmetric Key Generation		
ECC key pair generation (NIST curves P-256, P-384, P-521)	FIPS PUB 186-4	A3617: ECDSA KeyGen (FIPS186-4)
FCS_CKM.2 Cryptographic Key Establishment		
ECDSA based key establishment	NIST SP 800-56A	A3617: KAS-ECC-SSC Sp800-56Ar3

Functions	**Standards**	**Certificates**
FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption		
AES-CBC, AES-GCM (128, 256 bits)	CBC as defined in NIST SP 800-38A GCM as defined in NIST SP 800-38D	A3617: AES-CBC A3617: AES-GCM
FCS_COP.1/Hash Cryptographic Operation – Hashing		
SHA-256 and SHA-384 (digest sizes 256 and 384 bits)	FIPS PUB 180-4	A3617: SHA2-256 A3617: SHA2-384
FCS_COP.1/Sig Cryptographic Operation – Signing		
RSA (2048-bit or greater)	FIPS PUB 186-4, Section 4	A3617: RSA SigGen (FIPS 186-4) A3617: RSA SigVer (FIPS 186-4)
FCS_COP.1/KeyedHash Cryptographic Operation – Keyed Hash Message Authentication		
HMAC-SHA-256 and SHA-384	FIPS PUB 198-1 FIPS PUB 180-4	A3617: HMAC-SHA2-256 A3617: HMAC-SHA2-384
FCS_RBG_EXT.2 Random Bit Generation from Application		
CTR_DRBG DRBG (256 bits)	NIST SP 800-90A NIST SP 800-57	A3617: Counter DRBG

The TOE generates asymmetric keys in support of trusted communications. The TSF generates ECC keys using P-256, P-384, and P-521. 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 Windows platform version of the TOE relies on a third-party entropy source provided by the platform vendor. The Linux platform version of the TOE relies on the OS platform entropy source as well. Specifically, random numbers are obtained from the following platform APIs, depending on the platform used:

- Windows: SystemPRNG
- Linux: invocation of `/dev/random` pseudo-device.

In both cases, it is assumed that these platforms provide at least 256 bits of entropy.

The TOE uses TLS 1.2 for client communications. The TLS client implementations support the following TLS cipher suites in the TOE's evaluated configuration:

- TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
- TLS_ECDHE_RSA_WITH_AES_128_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, secp384r1, and secp521r1 as the supported values in the Supported Groups extension and uses the same NIST curves for key establishment.

As part of certificate validation in the establishment of TLS connectivity, the TOE will validate the reference identifier of a presented server certificate. This is done through validation of the Common Name (CN) and Subject Alternative Name (SAN) certificate fields, the latter of which is expected to contain the FQDN of the external system that is presenting the certificate to the TOE. The reference identifier is established by configuration. IP addresses are not supported. Wildcards are only supported for the left-most label immediately preceding the public suffix. Certificate pinning is not supported. All digital signatures used for the establishment of TLS communications use 2048-bit RSA.

The TOE uses TLS client functionality for communications between the TOE and an instance of the Nessus application in the operational environment. This communication does not use mutually-authenticated TLS. The TOE's implementation of HTTPS conforms to RFC 2818. The connection will be rejected if certificate validation fails.

The TOE uses cryptographic hash functions (SHA-256, SHA-384) in association with the following cryptographic functions:

- As the cryptographic hash function in the implementation of the keyed-hash message authentication algorithms (HMAC-SHA-256, HMAC-SHA-384), which in turn support integrity verification in TLS
- RSA digital signature generation and verification.

The TOE does not store any credentials so no specific method of secure credential storage is implemented.

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

- FCS_CKM.1/AK – The TOE uses a NIST-validated implementation to generate asymmetric keys in support of TLS communications.
- 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/SKC – The TOE uses a NIST-validated implementation to perform AES encryption and decryption in support of TLS communications.
- FCS_COP.1/Hash – The TOE uses a NIST-validated implementation to perform cryptographic hashing in association with keyed-hash message authentication and digital signature generation and verification functions.
- FCS_COP.1/Sig – The TOE uses a NIST-validated implementation to generate and verify RSA digital signatures in support of TLS communications.
- FCS_COP.1/KeyedHash – The TOE uses a NIST-validated implementation to perform HMAC functions in support of TLS communications.
- FCS_HTTPS_EXT.1/Client – The TOE implements HTTPS as a client 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 does not store any credential data that requires cryptographic protection.
- FCS_TLS_EXT.1 – The TOE implements TLS to secure data in transit.
- FCS_TLSC_EXT.1 – The TOE implements TLS as a client.
- FCS_TLSC_EXT.5 – The TOE's TLS client implementation presents supported elliptic curves to the server in the Supported Groups extension when an ECDHE cipher suites is negotiated.

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."

Based on this definition, the only sensitive data stored by the TOE is system scan results. This data is not maintained persistently on the platform because it is discarded after its transmission to the environmental Nessus application. However, it may temporarily reside in local storage in the event of the TOE's inability to communicate with Nessus.

In the evaluated configuration, the TOE will be installed on a platform that has full disk encryption enabled. All data at rest is ultimately secured by the operational environment's platform encryption functionality.

The only platform-provided hardware resource the TOE accesses is network connectivity, which it uses to establish connections to the Nessus application in its operational environment. The only sensitive information repository hosted on its platform the TOE accesses is system configuration data, which it collects from the local system for subsequent analysis.

The TOE uses environmental network capabilities to communicate with the environmental Nessus application in its operational environment. The following table highlights the TOE's network usage.

Table 6: TOE Network Usage

Component	User-Initiated	Externally-Initiated	TOE-Initiated
Nessus Agent	None	None	Transmission of scan results to Nessus
			Periodic polling of Nessus to determine if scan needs to be initiated or if binary/plugin updates need to be acquired
			Check for and download of binary and plugin updates from Nessus

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

- FDP_DAR_EXT.1 – Sensitive data at rest is protected by full disk encryption of the TOE's underlying OS platform.

- 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. The use of network resources is initiated by the TOE.

6.4 Identification and Authentication

The TOE uses X.509 certificates for validation of the Nessus TLS server certificate. It validates X.509 certificates using the path validation algorithm defined in RFC 5280, which can be summarized as follows:

- Check the public key algorithm and parameters of each certificate in the path
- Check the current date/time against the validity period of each certificate in the path
- Check the revocation status of each certificate in the path
- Check the issuer name to ensure it equals the subject name of the previous certificate in the path
- Check name constraints to make sure the subject name is within the permitted subtrees list of all previous CA certificates and not within the excluded subtrees list of any previous CA certificate
- Check the asserted certificate policy OIDs against the permissible OIDs of the previous certificate, including any policy mapping equivalencies asserted by the previous certificate
- Check policy constraints to ensure any explicit policy requirements are not violated
- For all CA certificates, confirm the presence of the basicConstraints extension and that the CA flag is set to TRUE, and ensure the key usage field includes the caSigning purpose
- Check the path length does not exceed any maximum path length asserted in this or a previous certificate
- Check the key usage extension
- Process any other recognized critical extensions.

The TOE validates the extendedKeyUsage field according to the following rules:

- Server certificates presented for TLS shall have the Server Authentication purpose (id-kp 1 with OID 1.3.6.1.5.5.7.3.1) in the extendedKeyUsage field.
- OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the EKU field.

The TOE does not use any of the other values for the extended KeyUsage field listed in the requirement (i.e., Code Signing, Client Authentication, Email Protection, or CMC Registration Authority), so this part of the requirement is trivially satisfied.

The TOE validates the certificate chain to its root to ensure it terminates with a trusted CA certificate. It performs a revocation check on each certificate in the chain (except the root certificate) using Online Certificate Status Protocol (OCSP) in accordance with RFC 6960. In the event the revocation status of a certificate cannot be verified (i.e., the OCSP responder cannot be reached), the TOE accepts the certificate.

Because the TOE's use of the certificate validation function is to validate the authenticity of remote endpoints, the TSF chooses what certificates to use based on what is presented to it as part of establishing the TLS session. The TOE is only assigned one certificate for its own use, so there is only one certificate that it will present in cases where a remote entity may need to validate it.

The Identification and Authentication security function is designed to satisfy the following security functional requirements:

- FIA_X509_EXT.1 – X.509 certificates are validated by the TSF when establishing trusted communications.

- FIA_X509_EXT.2 – X.509 certificates are used for TLS. When revocation status of a certificate cannot be determined, the TSF accepts the certificate by default.

6.5 Security Management

The TOE does not provide a direct user interface for configuration of the TOE. All configuration is initiated from the operational environment, using an instance of Nessus that is connected to the TOE.

The TOE is installed into the following location, depending on the platform version:

- Windows: `C:\Program Files\Tenable\nessus_agent`
- Linux: `/opt/nessus_agent`

All directories containing TOE software and data are configured by default in such a manner that nothing is world-writable on Linux and Administrator privileges are required to access them on Windows. Configuration settings that affect the TOE's interaction with the host OS platform are stored in `/etc` for Linux and the Windows Registry for Windows.

The TOE supports the following security-relevant management functions:

- Configuration of transmission of system's hardware, software, or configuration information
    - Conducts system scans of local systems. The environmental Nessus can be used to manually initiate a scan or to define a schedule that causes the TOE to initiate scanning at the specified times and intervals.

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

- FMT_CFG_EXT.1 – The TOE does not include functionality that uses administrative credentials. The TOE is protected from direct modification by untrusted users via their host OS platforms.
- FMT_MEC_EXT.1 – Configuration settings for the TOE are stored in appropriate locations for each supported host OS platform.
- FMT_SMF.1 – Administrators can use an environmental application to configure the collection of system data from the TOE's operational environment.

6.6 Privacy

The TOE's primary function is to examine an organizational asset for configuration or operational states that may indicate the presence of a vulnerability or misuse of organizational resources. To this end, the TOE transmits data about system configuration to its operational environment for aggregation, analysis, and reporting. The TOE is not responsible for the collection or transmission of 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.

6.7 Protection of the TSF

The TOE implements several mechanisms to protect against exploitation. The TOE supports address space layout randomization (ASLR) through the use of the `/DYNAMICBASE` (Windows) and `–fPIC` (Linux) compiler flags and relies fully on its underlying host platforms to perform memory mapping. The TOE does not map memory to an explicit address. The TOE also does not use both `PROT_WRITE` and `PROT_EXEC` on the same memory regions. The TOE is written in C. The TOE is compiled with stack overflow protection whether it is intended for use on Linux or Windows. The Linux platform version is compiled with `–fstack-protector-strong` and the Windows platform version is compiled with `/GS`.

Both platform versions of the TOE are designed to run on host OS platforms where platform security features have been enabled (e.g., Windows Defender Export Guard, SELinux 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 Linux platform version of 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 regardless of which platform version it is. An administrator can also check the version of the TOE using the environmental instance of Nessus that is connected to the TOE to check the TOE’s version.

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. However, since a typical deployment may include a large number of Nessus Agent instances, it is also possible to leverage the Nessus platform to obtain a software update for the TOE. The TOE can check for an available update when communicating with Nessus. The candidate update is initially obtained from the operational environment via an administrator downloading it from Tenable’s site, but the administrator is then able to place it on the environmental Nessus so that the TOE obtain it from there. The TOE will not download, modify, replace, or update its own binary code. The TOE is packaged as an .rpm file for Linux and an .exe file for Windows. Each are digitally signed by Tenable using 2048-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.

6.8 Trusted Path/Channels

The TOE encrypts sensitive data in transit between itself and its operational environment using TLS and HTTPS . Listed below is the external interface to the TOE that relies on trusted communications.

Between TOE and environmental Tenable components:

- Between Nessus and TOE:
  - Communications use TLS/HTTPS (TOE is client and Nessus is server)
  - Configurable TCP port, 8834 is default
  - Used by Nessus to collect local scan results from the TOE.

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

- FTP_DIT_EXT.1 – The TOE relies on its own mechanisms to secure 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 1.4, 7 October 2021 ([APP_PP]) and *Functional Package for Transport Layer Security (TLS)*, Version 1.1, 1 March 2019 ([TLS_PKG]) along with all applicable errata and interpretations from the certificate issuing scheme.

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

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

As explained in section 4, Security Objectives, the Security Objectives of [APP_PP] have been included by reference into this ST.

All claimed SFRs are defined in [APP_PP] and [TLS_PKG]. 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 Security Problem Definition, Security Objectives, and Security Assurance Requirements from [APP_PP]. 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 [APP_PP] and [TLS_PKG]. 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 7 demonstrates the relationship between security requirements and functions.

Table 7: Security Functions vs. Requirements Mapping

	Cryptographic Support	**User Data Protection**	**Identification and Authentication**	**Security Management**	**Privacy**	**Protection of the TSF**	**Trusted Path/Channels**
FCS_CKM_EXT.1	X						
FCS_CKM.1/AK	X						
FCS_CKM.2	X						
FCS_COP.1/SKC	X						
FCS_COP.1/Hash	X						
FCS_COP.1/Sig	X						
FCS_COP.1/KeyedHash	X						
FCS_HTTPS_EXT.1/Client	X						
FCS_RBG_EXT.1	X						
FCS_RBG_EXT.2	X						
FCS_STO_EXT.1	X						
FCS_TLS_EXT.1	X						
FCS_TLSC_EXT.1	X						
FCS_TLSC_EXT.5	X						
FDP_DAR_EXT.1		X					
FDP_DEC_EXT.1		X					
FDP_NET_EXT.1		X					
FIA_X509_EXT.1			X				
FIA_X509_EXT.2			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	

	Cryptographic Support	User Data Protection	Identification and Authentication	Security Management	Privacy	Protection of the TSF	Trusted Path/Channels
FPT_IDV_EXT.1						X	
FPT_LIB_EXT.1						X	
FPT_TUD_EXT.1						X	
FPT_TUD_EXT.2						X	
FTP_DIT_EXT.1							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 outside the logical boundary of the product could still present an exploitable vulnerability.

Windows:

CrtSetDbgFlag,_heapmin,_set_invalid_parameter_handler,_strtoi64,_unix2windows_64,_wtoi,_wtoi64, accept, AllocateAndInitializeSid,bind, CheckTokenMembership, CloseHandle, CloseServiceHandle,closesocket, CoCreateInstance, CoInitializeEx, CoInitializeSecurity, CompareStringW,connect, ControlService, CoSetProxyBlanket, CoUninitialize, CreateEvent, CreateFile, CreateFileA, CreatePipe, CreateProcess, CreateService, CreateThread, CreateToolhelp32Snapshot,ctime_s, DeleteFileA, DeleteService, EnterCriticalSection, EnumServicesStatus, FileTimeToSystemTime, FindClose, FindFirstFileA, FindNextFile, FormatMessage, FreeLibrary, FreeSid, GetComputerNameA, GetConsoleMode, GetCurrentProcess, GetCurrentProcessId, GetCurrentThreadId,getenv, GetExitCodeProcess, GetFileAttributes, GetFileSize, GetFileVersionInfoExW, GetFileVersionInfoSizeExW, GetLastError, GetModuleHandle, GetProcAddress, GetProcessHeap, GetProcessId, GetSecurityDescriptorControl, GetSecurityInfo, GetServiceDisplayNameA,getsockname, GetStdHandle, GetSystemTimeAsFileTime, GetTcpTable, GetTimeZoneInformation, GetUserName, GetVersionEx, GetWindowsDirectoryA, GlobalMemoryStatusEx, HeapAlloc, HeapFree, HeapLock, HeapReAlloc, HeapSetInformation, HeapUnlock, HeapWalk,htonl,htons, IEnumWbemClassObject_Next, IEnumWbemClassObject_Release, IsValidSecurityDescriptor, IsWindowsServer, IWbemCallResult_GetCallStatus, IWbemCallResult_GetResultObject, IWbemCallResult_Release, IWbemClassObject_BeginEnumeration, IWbemClassObject_EndEnumeration, IWbemClassObject_Get, IWbemClassObject_GetMethod, IWbemClassObject_Next, IWbemClassObject_Put, IWbemClassObject_QueryInterface, IWbemClassObject_Release, IWbemClassObject_SpawnInstance, IWbemLocator_ConnectServer, IWbemLocator_Release, IWbemServices_AddRef, IWbemServices_ExecMethod, IWbemServices_ExecQuery, IWbemServices_GetObject, IWbemServices_Release, LeaveCriticalSection,listen, LoadLibrary, LocalFree,localtime_s, LsaClose, LsaEnumerateAccountsWithUserRight, LsaFreeMemory, LsaLookupNames, LsaLookupSids, LsaOpenPolicy, LsaQueryDomainInformationPolicy, LsaQueryInformationPolicy,lstrlenW,memset, Module32First, Module32Next, MultiByteToWideChar, NetApiBufferFree, NetGroupGetUsers, NetLocalGroupGetMembers, NetServerEnum, NetServerGetInfo, NetSessionEnum, NetShareEnum, NetUserGetGroups, NetUserGetInfo, NetUserGetLocalGroups, NetUserModalsGet, NetWkstaGetInfo, NetWkstaUserEnum, OpenProcess, OpenSCManagerA, OpenService, PdhAddCounterA, PdhCollectQueryData, PdhOpenQueryA, poll, QueryServiceObjectSecurity, QueryServiceStatus, RaiseException, ReadConsole, ReadFile, recv, RegCloseKey, RegCreateKeyExA, RegEnumKeyA, RegEnumKeyExA, RegEnumValueA, RegGetKeySecurity, RegOpenKeyA, RegOpenKeyExA, RegQueryInfoKeyA, RegQueryValueExA, RegSetValueExA, SafeArrayAccessData, SafeArrayCreateVector, SafeArrayGetDim, SafeArrayGetLBound, SafeArrayGetUBound, SafeArrayUnaccessData, SamCloseHandle, SamConnect, SamFreeMemory, SamOpenDomain, SamQueryInformationDomain, saturate, send, SetConsoleCtrlHandler, SetConsoleMode, SetFilePointer, SetHandleInformation, SetLastError, setsockopt, SetStdHandle, SetUnhandledExceptionFilter, SHGetFolderPath, Sleep, socket, StackWalk64, StartService, SysAllocStringByteLen, SysAllocStringLen, SysFreeString, SysStringLen, SystemTimeToFileTime, SystemTimeToTzSpecificLocalTime, TerminateProcess, time, TzSpecificLocalTimeToSystemTime, VariantClear, VerQueryValueA, vsnprintf_s, w32_change_privilege, WaitForSingleObject, WideCharToMultiByte, Win32, WriteFile, WSACleanup, WSAGetLastError, WSAStartup, ZeroMemory, GetAdaptersAddresses, GetAdaptersInfo, SHGetS(pecialFolderPath, lrand48, GetTickCount64, QueryPerformanceFrequency, QueryPerformanceCounter, GetTickCount, GetUserNameW, DnsQuery_UTF8, DnsRecordListFree, SetPriorityClass, BCryptGenRandom

Linux:

GNU C Library, GNU Standard C++ Library

Used by plugins:

cat, grep, echo, wc, find, ps, type, uname, ls, awk, sysinfo, sysctl, which, strings, head, echo, test, read, readlink, rm, chmod, od, tr, sh, cut, sort, tee, chown, do, while, if, hexdump, strcat, tail, sed, xargs, dd, netstat, ifconfig, ip, hostname

Used by plugins, to test specific platforms/applications:

init, unzip, nwmgr, lanscan, docker, cli, dhcpd, unzip, equery, xl, sqlite3, db2ls, db2level, db2, md5sum, sha256sum, sha1sum, yara, lsof, cmdagent, nails, sc, sc4, vmware-view, loginsight, bash, dmidecode, wget, curl, rpm, cfservd, prelink, netstat, rmsock, inpcb, proctree, service, dpkg, iptables, lsmod, openssl, splunk, namei, java, dd

Used by Nessus bug report generator:

uname, dmesg, tail, killall, sh, uptime, ls, ps, grep, xargs, netstat, arp, df, cat, tail, rpm, free, ifconfig, du, tar

A.2 Third-Party Libraries

Listed below are the third-party libraries, with their version numbers, used by the TOE. Note that these libraries 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 outside the logical boundary of the product could still present an exploitable vulnerability.

Library	Version
apr-iconv	1.2.2
ced	Commit 1193457d
expat	2.5.0
jemalloc	5.2.1
libbzip2	1.0.8
libpcre	8.42

Library	**Version**
libjpeg	9d
libxml2	2.10.3
libxslt	1.1.34
libxmlsec	1.2.25
zlib	1.2.13
openssl	3.0.9
sqlite	3.34.1
jsonsl	Commit 684b60f
Snappy	1.1.7
RapidJSON	1.1.0
MS VC Redist	14.22