VMware Horizon Agent 8 2209 (Horizon 8.7)

Security Target

Version 1.0

17 May 2023

Prepared for:

VMware, Inc.
3401 Hillview Avenue
Palo Alto, CA 94304

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)
- TOE Summary Specification (Section 6)
- Protection Profile Claims (Section 7)
- Rationale (Section 8)
- A TOE Usage of Third-Party Components (Appendix 0)

1.1 Security Target, TOE and CC Identification

ST Title – VMware Horizon Agent 8 2209 (Horizon 8.7) Security Target

ST Version – Version 1.0

ST Date – 17 May 2023

TOE Identification – VMware Horizon Agent 8 2209 (Horizon 8.7)

TOE Developer – VMware, Inc.

Evaluation Sponsor – VMware, 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, 07 October 2021* (App PP) with the following optional and selection-based SFRs:
  - FCS_CKM.1/AK
  - FCS_CKM_EXT.1/PBKDF (iterated by ST author)
  - FCS_CKM.1/SK
  - FCS_CKM.2
  - FCS_COP.1/Hash
  - FCS_COP.1/KeyedHash
  - FCS_COP.1/Sig
  - FCS_COP.1/SKC
  - FCS_HTTPS_EXT.1/Server
  - FCS_RBG_EXT.2
  - FPT_TUD_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:

  TD0442: Updated TLS Ciphersuites for TLS Package

  - This TD is applicable to the TOE.

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

  - This TD is applicable to the TOE but applies specifically to test activities so the ST itself is unaffected.

  TD0499: Testing with pinned certificates

  - This TD is applicable to the TOE because it affects an SFR that the TOE claims. However, the TOE does not support certificate pinning so the TD's modification to the testing for this does not affect the claims made for the TSF.

  TD0513: CA Certificate loading

  - This TD is not applicable to the TOE. It applies to FCS_TLSC_EXT.1, which the TOE does not claim.

  TD0588: Session Resumption Support in TLS package

  - This TD is applicable to the TOE.

  TD0624: Addition of DataStore for Storing and Setting Configuration Options

  - This TD is not applicable to the TOE because it does not have an Android platform version.

  TD0628: Addition of Container Image to Package Format

  - This TD is not applicable to the TOE because it is not packaged in a container format.

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

  - This TD is not applicable to the TOE because it does not claim VPN client functionality.

  TD0655: Mutual authentication in FTP_DIT_EXT.1 for SW App

  - This TD is applicable to the TOE.

  TD0664: Testing activity for FPT_TUD_EXT.2.2

  - This TD is applicable to the TOE.

  TD0669: FIA_X509_EXT.1 Test 4 Interpretation

  - This TD is not applicable to the TOE; the TOE does not claim this SFR.

  TD0709: Number of elements for iterations of FCS_HTTPS_EXT.1

  - This TD is applicable to the TOE.

  TD0717: Format changes for PP_APP_V1.4

  - This TD is applicable to the TOE.

TD0719: ECD for PP APP V1.3 and 1.4

- This TD is not applicable to the TOE; this TD updates the App PP to include a formal ECD which is needed for the PP itself to conform to CC Part 3. This does not change the ST or how the evaluation of the TOE is conducted.

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

- This TD is applicable to the TOE.

- 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 slash followed by a descriptor for the purpose of the iteration. For example, FCS_HTTPS_EXT.1/Server indicates that the FCS_HTTPS_EXT.1 requirement applies specifically to HTTPS server functionality. The ST author also uses the '/L' and '/W' iteration conventions (which may themselves follow a PP-defined iteration, e.g. FCS_CKM_EXT.1/PBKDF/L) to identify SFRs that only apply to the Linux or Windows platform version of the TOE.
  - 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 selection/assignment 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
Agent	A Horizon component that acts as an endpoint on a protected resource and serves content on that resource (individual applications or an interactive desktop session) to an authorized Horizon Client.
Blast	A communications protocol that is used to transmit interactive desktop and application sessions (user inputs and audio/visual outputs).
Client	A Horizon component that resides on an end user device that the user can run to access enterprise computing resources via the virtual desktop.
Cloud Pod	A self-contained Horizon deployment on a particular network. Multiple cloud pods can be federated, allowing a client on one pod to access resources on another.
Connection Server	A Horizon component that is responsible for determining the authorizations of a Horizon Client user and facilitating the establishment of Agent communications so that authorized resources can be served to that user.
Horizon	A collection of products that are used to allow an organizational user to access shared enterprise resources in a protected network from a single client application.
Unified Access Gateway	A network device that acts as a proxy between a Horizon Client on an unprotected network and other Horizon components on a protected internal network. The Unified Access Gateway is responsible for authenticating Horizon Client users and passing their validated identity to a Connection Server via SAML assertion. It is also responsible for establishing Horizon Agent connectivity on behalf of the client.
Virtual Desktop	The virtual desktop is the set of enterprise computing resources that are served to a user within an interactive Horizon Client session. For the purpose of the TSF, the important consideration is that all virtual desktop content is transmitted over TLS.

1.3.2 Acronyms

Table 2: Acronyms

Term	Definition
ASLR	Address Space Layout Randomization
CDR	Client Drive Redirection
DRBG	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
JMS	Java Message Service
JRE	Java Runtime Environment
PGP	Pretty Good Privacy
PII	Personally Identifiable Information
PP	Protection Profile
RDP	Remote Desktop Protocol
SAML	Security Assertion Markup Language
SAN	Subject Alternative Name
ST	Security Target
SWID	Software Identification (standard)
TOE	Target of Evaluation
TSF	TOE Security Functionality
UAG	Unified Access Gateway
VM	Virtual Machine
VPN	Virtual Private Network
WAN	Wide-Area Network

2 Product and TOE Description

2.1 Introduction

VMware Horizon is a collection of applications that work together to deliver centralized enterprise resources to end users. This is done by providing users with a “virtual desktop” that consolidates their authorized enterprise computing environments and applications into a single view that is presented to them through a client application.

For this Security Target, the Target of Evaluation (TOE) is the VMware Horizon Agent 8 application (“Horizon Agent”), specifically version 2209 or 8.7. This is a server application that is responsible for serving content on the system it runs on to an authorized Horizon Client accessing it through the virtual desktop. Depending on configuration, this may refer to the Horizon Client being able to launch specific applications on the Agent’s host OS platform or to run an interactive remote desktop session on the platform OS itself.

The TOE conforms to the App PP and TLS Package. As such, the evaluation of 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

VMware Horizon is a suite of applications that establish a virtualization environment within an organization. The Horizon applications collectively allow users to access virtualized desktops or enterprise resources from their end user device. These resources are made available with granular security controls that allow users to access only the capabilities for which they are authorized.

VMware Horizon as a suite consists of several components:

- Horizon Clients are applications that are installed on end user devices. A user accesses their virtual desktop through the Horizon Client.
- Horizon Agents are applications that run on virtual servers in the enterprise environment. These agents facilitate remote access to the desktop of a virtual server or to specific applications running on that server that may be served directly to the virtual desktop.
- The Horizon Connection Server is responsible for brokering connections between Horizon Clients and Horizon Agents to authenticate users and serve appropriate resources to a particular user based on enterprise permissions.

A VMware Horizon deployment typically includes one or more instances of the VMware Unified Access Gateway (UAG) as well. The purpose of the UAG is to enforce separation of internal and external networks. This allows the Horizon Client to act as a TLS VPN to access services within the protected network when the end user device is in an external setting such as an untrusted mobile Wi-Fi network.

In cases where a Horizon deployment needs to give users access to resources that span multiple physical data centers or are maintained by multiple organizations, Horizon also supports a cloud pod architecture. This allows for multiple Connection Servers to be federated so that access to Horizon Agent resources on disparate WANs can be served through a single Horizon Client session. Within a single pod, a single Connection Server can also be replicated to ensure availability.

2.3 TOE Overview

The Target of Evaluation (TOE) is VMware Horizon Agent 8 application. The specific evaluated version of the application is version 2209 or 8.7; these are synonymous. All references to "Horizon Agent" or simply "Agent" throughout the ST refer to this specific version. The application has both Windows and Linux platform versions.

With respect to the security functionality of the TOE, the TSF is limited to the relevant functionality that is defined in the claimed PP and package. The logical boundary of the TOE is summarized in section 2.4.2. However, the following general capabilities are considered to be within the scope of the TOE:

- **Protection of sensitive data at rest:** the TOE implements and potentially leverages secure platform storage and encryption mechanisms to protect credentials and other sensitive data at rest, depending on platform version.
- **Protection of data in transit:** the TOE secures sensitive data in transit between itself and its operational environment using TLS/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 and Bouncy Castle BC-FJA with CAVP 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 its underlying host operating system platform in such a way that the TOE cannot be used as an attack vector to compromise an operating system. The specific mechanisms to achieve this differ between platform versions but the same underlying security threats are mitigated in both cases.

Notably, there are no standardized security requirements in the claimed PP (or any other published PP at the time of this ST's publication) for implementation of desktop media streaming to a remote client. Therefore, the security of the TOE's external interfaces is only assessed with respect to the ability of the TOE to protect these communications from unauthorized modification or disclosure.

The TOE has a FIPS-compliant mode of operation. The evaluated configuration of the TOE requires this to be enabled so that the cryptographic functionality uses the algorithms claimed in the TSS.

2.4 TOE Architecture

The Horizon Agent TOE consists of the Horizon Agent application. The TOE has both Windows and Linux platform versions. The Windows application consists of C, C++, and Java code, and the Linux application consist of C, C++, Java, Python, JavaScript, and shell code. Third-party components used by the TOE (see Appendix A.2) are linked into the TOE binaries or run as a system service, depending on the component.

2.4.1 Physical Boundary

Figure 1 shows the TOE in a sample deployment with other VMware Horizon applications in its operational environment. Note the following:

- This figure also includes a UAG and therefore assumes that the TOE platform is located inside of the protected network in which all components reside except for the Horizon Client and its underlying system. The UAG acts as a reverse proxy to handle all inbound TLS/HTTPS connections from the Horizon Client.

- Firewalls are not shown between internal and external networks but it is assumed that the UAG is deployed in a DMZ between them.
- Multiple UAGs may be deployed in a load balancer configuration to ensure resource availability. As the claimed PP and package do not have availability requirements, only one UAG is deployed in the tested configuration.
- The second Horizon Connection Server that is depicted on the diagram has its own associated Horizon Agents and other external interfaces. These are omitted for simplicity.
- Horizon Agents are deployed on multiple virtual systems. Horizon Agents support multiple hypervisors but the TOE's evaluated configuration assumes the use of VMware ESXi for this. For simplicity's sake, the Horizon Agent is represented by a single logical component on the figure. The associated physical device and hypervisor are not shown on the figure as the TSF only interfaces logically with the specific application component of those systems. The UAG and Connection Server are similarly deployed in virtualized environments and the figure does not show their dependent components. Of note to the TOE is that there is no direct interface between multiple Horizon Agents so the presence or absence of other Agents does not affect the TOE's behavior.
- The Connection Server network may include multiple virtual systems on the same physical host that are networked virtually. Specifically, the same physical host may include separate VMs running Horizon Connection Server, Horizon Agent, and vCenter components all as part of the same managed VM infrastructure.
- The 'Database' component refers to the optional Event Database (SQL Server or Postgres) that is used by the Connection Server if configured.
- The Certification Authority depicted in the diagram is not used by the TOE because its only use of certificates is as a TLS server with no mutual authentication; it is included in the environment to support other environmental components.
- The CRL Distribution Points are used by environmental components to check the validity of presented X.509 certificates. The Horizon Agent does not have a certificate validation function in the TOE boundary so this functionality is non-interfering with respect to the TSF.

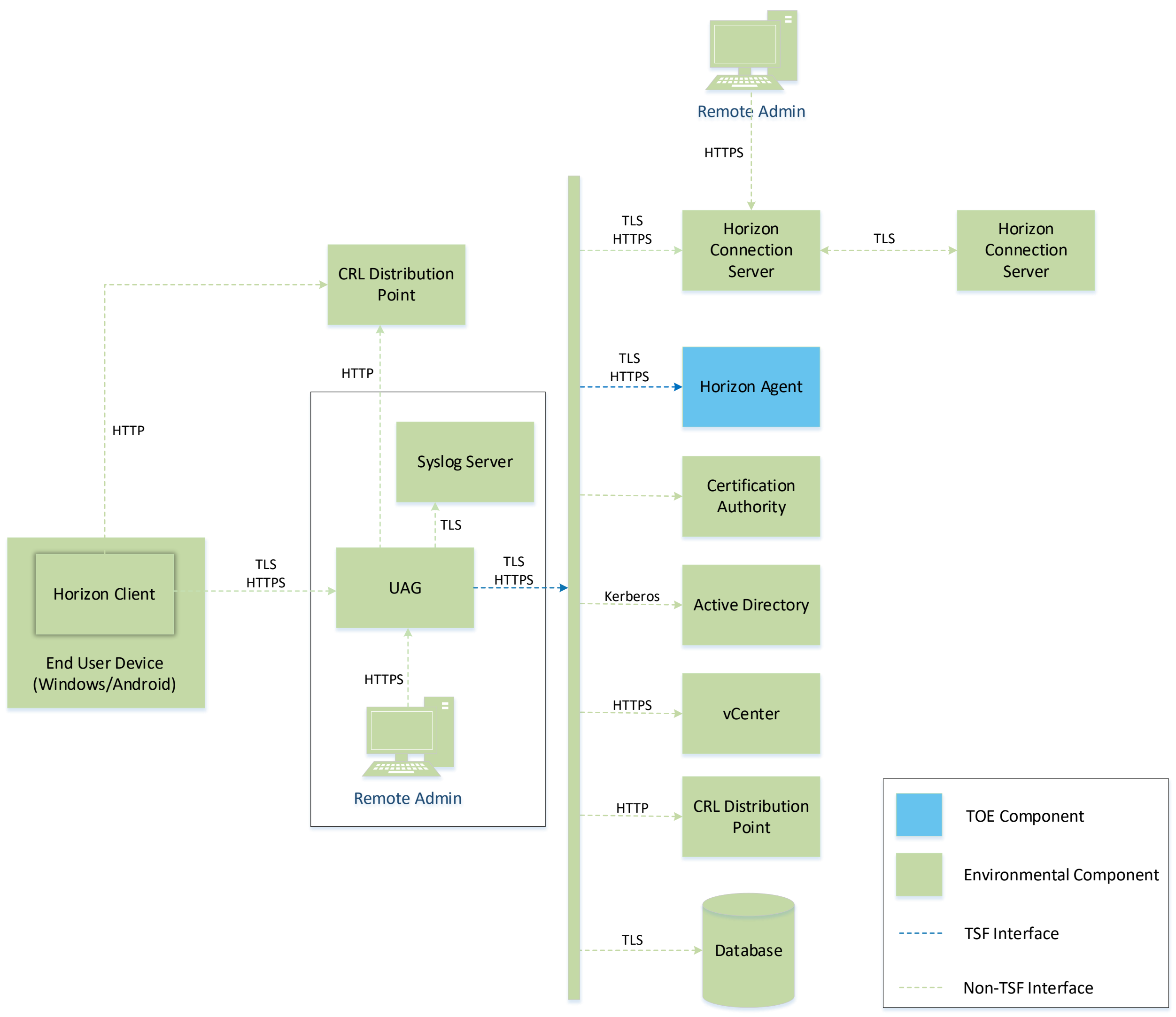

Figure 1 – TOE Boundary

The TOE establishes connectivity to a Horizon Connection Server to report on its status and for configuration and log data. When a user wants to establish a virtual desktop session with the TOE, they will present an authorization token issued by the Connection Server that identifies the resources that user is authorized to access. No separate configuration is needed for this because the TOE has a trust relationship with the Connection Server that requested the authorizations. The TOE will then establish a TLS session with the proxy UAG and serve content to the user via that connection. All content is processed through the VMware Blast protocol and uses TCP.

Interactions between the TOE and the Connection Server involving exchange of security-relevant information relies on data-at-rest encryption and decryption to protect this data prior to its transmission. As a result, the data transmitted between these components is not considered to be sensitive because its disclosure will not result in a compromise of data. Interactions between the TOE and a Horizon Client use one-way TLS, with the TOE acting as the server. This channel is proxied by the environmental UAG so the connection is actually between the UAG and the TOE.

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

- Windows platform:
  - Windows Server 2019 and Windows 10, both virtualized on VMware ESXi 7.0
  - Platform must be configured into FIPS-compliant mode of operation
- Linux platform:
  - RHEL 8, virtualized on VMware ESXi 7.0
  - Platform must be configured into FIPS-compliant mode of operation
- VMware ESXi 7.0
- Intel Pentium IV 2.0GHz processor or higher – 2 CPUs minimum, 4 CPUs recommended
  - The TOE's tested configuration uses an Intel Xeon 6230R (Cascade Lake)
- 4 GB RAM – at least 40GB recommended for deployments of 50 or more remote desktops
- 100 Mbps NIC – 1 Gbps recommended

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

- TCP/22443 (for inbound Blast protocol connectivity to Horizon Client via UAG)

The TOE's operational environment includes the following:

- Other VMware Horizon components (at least one each of Horizon Connection Server and Horizon Client).
- Network access between "outer" and "protected" networks mediated through VMware UAG.
- Platform (hardware and software) on which the TOE is hosted, described above in the list of system requirements.
- VMware VM Encryption is required for the TOE platform to ensure adequate data-at-rest protection.
- Authentication server (Active Directory) – optional for configuration.

The TOE has the following logical exclusions, in addition to any functionality that is not directly related to any of the SFR claims made in this ST:

- Tunnel Channel – The Windows platform version of the Horizon Agent has a separate tunnel channel that allows for communications of Microsoft RDP and Windows Media MMR through HTTPS. This channel also allows a USB device connected to a remote Horizon Client system to be accessible by the TOE platform as if it was plugged in to the remote device (USB redirection), and it allows for the Horizon Client system's file system to be similarly accessible on the virtual desktop (Client Drive Redirection, or CDR). In the evaluated configuration, the communications that use the tunnel channel are configured to use Blast instead.
- PcoIP – The Windows platform version of the Horizon Agent supports PC over IP (PcoIP) for remote communications with Horizon Clients. In the evaluated configuration, this is disabled and Blast is used instead (the Linux platform version only supports Blast).

The TOE has multiple editions with different features that are activated by licensing. The security functionality claimed within the TOE boundary is not affected by which license is used. The highest tier edition (Enterprise) was used for the tested configuration.

2.4.2 Logical Boundary

This section summarizes the security functions provided by the TOE:

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

2.4.2.1 Cryptographic Support

The TOE makes use of cryptography to protect data at rest and in transit.

For data at rest, the Windows platform version of the TOE relies on its operational environment to control access to stored credential data stored as certificates. All other credential data for both the Windows and Linux platform versions are protected by TSF-provided cryptographic functions.

For protection of sensitive data in transit, the TOE implements TLS/HTTPS as a server. The TOE implements all cryptography used for these functions using its own implementation of OpenSSL with CAVP validated algorithms. The TOE also implements cryptography through its own implementation Bouncy Castle BC-FJA. This is used to decrypt and encrypt data that is transmitted between the environmental Connection Server and the TOE. The TOE's DRBG is seeded using entropy from the underlying OS platform.

2.4.2.2 User Data Protection

The TOE relies on volume encryption via VMware VM Encryption to protect sensitive data at rest in addition to the mechanisms listed in section 2.4.2.1 above that are used to protect credential data at rest.

The TOE relies on the network connectivity of its host OS platform. The TOE can also access the system clipboard, audio/video capture devices, and file system resources.

2.4.2.3 Security Management

The TOE itself and the configuration settings it uses are stored in locations recommended by the platform vendor. The TOE is launched by an authenticated OS user and runs in the session context of that user; there is no interface to the TSF to act as an administrator through separate authentication. Changes to the product configuration are initiated from the Operational Environment.

2.4.2.4 Privacy

The TOE does not have a mechanism to retrieve or transmit personally identifiable information (PII) of any individuals.

2.4.2.5 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 through the application itself or by leveraging its OS platform, depending on the platform version of the TOE. All updates are digitally signed to guarantee their authenticity and integrity.

2.4.2.6 Trusted Path/Channels

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

2.5 TOE Documentation

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

- VMware Horizon 2209 Installation and Upgrade (https://docs.vmware.com/en/VMware-Horizon/2209/horizon-installation.pdf), 2022
- VMware Horizon 2209 Windows Desktops and Applications in Horizon (https://docs.vmware.com/en/VMware-Horizon/2209/virtual-desktops.pdf), 2022
- VMware Horizon 2209 Linux Desktops and Applications in Horizon (https://docs.vmware.com/en/VMware-Horizon/2209/linux-desktops-setup.pdf), 2022
- VMware Horizon 2209 Horizon Security (https://docs.vmware.com/en/VMware-Horizon/2209/horizon-security.pdf), 2022
- VMware Horizon 2209 Horizon Overview and Deployment Planning (https://docs.vmware.com/en/VMware-Horizon/2209/horizon-architecture-planning.pdf), 2022
- VMware Horizon Agent 8 2209 (Horizon 8.7) Common Criteria (CC) Evaluated Configuration Guidance, Version 1.0, May 17, 2023

3 Security Problem Definition

This ST includes by reference the Security Problem Definition, composed of threats and assumptions, from the 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 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 1.4, October 7, 2021
- *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_EXT.1/PBKDF Password Conditioning
- 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 (iterated by ST author)
- 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

Defined in TLS Package:

- FCS_TLS_EXT.1 TLS Protocol
- FCS_TLSS_EXT.1 TLS Server Protocol (iterated by ST author)

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_EXT.1/PBKDF/L Password Conditioning
	FCS_CKM.1/AK Cryptographic Asymmetric Key Generation
	FCS_CKM.1/SK Cryptographic Symmetric Key Generation
	FCS_CKM.2 Cryptographic Key Establishment
	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_COP.1/SKC Cryptographic Operation – Encryption/Decryption
	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/L Storage of Credentials (Linux Agent)
	FCS_STO_EXT.1/W Storage of Credentials (Windows Agent)
	FCS_TLS_EXT.1 TLS Protocol (TLS Package)
	FCS_TLSS_EXT.1/L TLS Server Protocol (Linux Agent) (TLS Package)
	FCS_TLSS_EXT.1/W TLS Server Protocol (Windows Agent) (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
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[1]

FCS_CKM_EXT.1.1 The application shall [

- Implement asymmetric key generation

].

5.2.1.2 FCS_CKM_EXT.1/PBKDF/L Password Conditioning[2]

FCS_CKM_EXT.1.1/PBKDF/L A password/passphrase shall perform [*PBKDF2*] in accordance with a specified cryptographic algorithm as specified in FCS_COP.1/KeyedHash, with [*16384*] iterations, and output cryptographic key sizes [128] that meet the following [NIST SP 800-132].

FCS_CKM_EXT.1.2/PBKDF/L 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_EXT.1.1/PBKDF**/L**.

Application Note: *This function is only implemented on the Linux platform version of the TOE.*

5.2.1.3 FCS_CKM.1/AK Cryptographic Asymmetric Key Generation[3]

FCS_CKM.1.1/AK The application shall [

- implement functionality

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

- [RSA schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3],
- [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],
- [FFC Schemes] using cryptographic key sizes of [2048-bit or greater] that meet the following: FIPS PUB 186-4, "Digital Signature Standards (DSS)", Appendix B.1]

].

[1] Modified by TD0717

[2] Modified by TD0717

[3] Modified by TD0717

5.2.1.4 FCS_CKM.1/SK Cryptographic Symmetric Key Generation

FCS_CKM.1.1/SK The application shall generate symmetric cryptographic keys using a Random Bit Generator as specified in FCS_RBG_EXT.1 and specified cryptographic key sizes [128 bit, 256 bit].

5.2.1.5 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"]

].

5.2.1.6 FCS_COP.1/Hash Cryptographic Operation – Hashing[4]

FCS_COP.1.1/Hash The application shall perform [cryptographic hashing services] in accordance with a specified cryptographic algorithm [

- SHA-256,
- SHA-384,
- SHA-512

] and message digest sizes [

- 256,
- 384,
- 512

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

5.2.1.7 FCS_COP.1/KeyedHash Cryptographic Operation – Keyed-Hash Message Authentication[5]

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

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

[4] Modified by TD0717

[5] Modified by TD0717

- HMAC-SHA-512

] and [

- no other algorithms]

with key sizes [*256, 384, 512 bits*] and message digest sizes [256, 384, 512] 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_COP.1/Sig Cryptographic Operation – Signing[6]

FCS_COP.1.1/Sig 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]

].

5.2.1.9 FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption[7]

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

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

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

5.2.1.10 FCS_HTTPS_EXT.1/Server HTTPS Protocol[8]

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 Functional Package for TLS.

FCS_HTTPS_EXT.1.3/Server The application shall [not process peer certificates] if the peer certificate is deemed invalid.

5.2.1.11 FCS_RBG_EXT.1 Random Bit Generation Services

FCS_RBG_EXT.1.1 The application shall [

[6] Modified by TD0717

[7] Modified by TD0717

[8] Modified by TD0709

- implement DRBG functionality

] for its cryptographic operations.

5.2.1.12 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 [Hash_DRBG (any), 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.

Application Note: *The TOE includes both the BC-FJA and OpenSSL cryptographic libraries. BC-FJA uses Hash_DRBG while OpenSSL uses CTR_DRBG(AES).*

5.2.1.13 FCS_STO_EXT.1/L Storage of Credentials (Linux Agent)

FCS_STO_EXT.1.1/L The application shall [

- implement functionality to securely store [*server certificate, key pairs for remote users*] according to [FCS_CKM_EXT.1/PBKDF/**L**]

] to non-volatile memory.

5.2.1.14 FCS_STO_EXT.1/W Storage of Credentials (Windows Agent)

FCS_STO_EXT.1.1/W The application shall [

- invoke the functionality provided by the platform to securely store [*server certificate, key pairs for remote users*]

] to non-volatile memory.

5.2.1.15 FCS_TLS_EXT.1 TLS Protocol (TLS Package)

FCS_TLS_EXT.1.1 The product shall implement [

- TLS as a server

].

5.2.1.16 FCS_TLSS_EXT.1/L TLS Server Protocol (Linux Agent) (TLS Package)

FCS_TLSS_EXT.1.1/L[9] 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_GCM_SHA256 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/L 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/L[10] The product shall perform key establishment for TLS using [

- ECDHE parameters using elliptic curves [secp256r1, secp384r1, secp521r1] and no other curves

].

5.2.1.17 FCS_TLSS_EXT.1/W TLS Server Protocol (Windows Agent) (TLS Package)

FCS_TLSS_EXT.1.1/W[11] 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_GCM_SHA256 as defined in RFC 5289,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

] and also supports functionality for [

- Session resumption based on session IDs according to ~~RFC 4346 (TLS 1.1) or~~ RFC 5246 (TLS 1.2),
- Session resumption based on session tickets according to RFC 5077

].

FCS_TLSS_EXT.1.2/W 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/W[12] The product shall perform key establishment for TLS using [

[9] Modified by TD0442

[10] Modified by TD0726

[11] Modified by TD0442 and TD0588

[12] Modified by TD0726

- ECDHE parameters using elliptic curves [secp256r1, secp384r1, secp521r1] and no other curves

].

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 [

- protect sensitive data in accordance with FCS_STO_EXT.1,
- leverage platform-provided functionality to encrypt sensitive data

] in non-volatile memory.

Application Note: *This ST iterates FCS_STO_EXT.1. Both iterations apply to the selection above.*

5.2.2.2 FDP_DEC_EXT.1 Access to Platform Resources

FDP_DEC_EXT.1.1 The application shall restrict its access to [

- network connectivity,
- [*audio/video interface*]

].

FDP_DEC_EXT.1.2 The application shall restrict its access to [

- system logs,
- [*file system,*
- *clipboard*]

].

5.2.2.3 FDP_NET_EXT.1 Network Communications

FDP_NET_EXT.1.1 The application shall restrict network communication to [

- respond to [*Blast connectivity*],
- [*application-initiated communications with Connection Server*]

].

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.

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].

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 [

- *no management functions*

].

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 built with stack-based buffer overflow protection enabled.

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 [[*date-based versioning, major/minor release versioning*]].

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 substantial 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.

Application Note: *The Windows platform version of the TOE has both TSF-provided and platform provided mechanisms to identify the TOE version. The Linux platform version of the TOE has only a platform mechanism for this.*

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 [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.

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

5.2.6 Trusted Path/Channels (FTP)

5.2.6.1 FTP_DIT_EXT.1[13] Protection of Data in Transit

FTP_DIT_EXT.1.1 The application shall [

- encrypt all transmitted [sensitive data] with [HTTPS as a server in accordance with FCS_HTTPS_EXT.1/Server, TLS as a server as defined in the Functional Package for TLS and also supports functionality for [none]]

] 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.

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.

[13] Modified by TD0655

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

VMware uses an internal classification system to categorize product security flaws by severity level. The classifications and their respective service-level agreements for mitigation are as follows:

- Critical:
    - Vulnerabilities that can be exploited by an unauthenticated attacker from the Internet or those that break the guest/host Operating System isolation. The exploitation results in the complete compromise of confidentiality, integrity, and availability of user data and/or processing resources without user interaction. Exploitation could be leveraged to propagate an Internet worm or execute arbitrary code between Virtual Machines and/or the Host Operating System.
    - A fix or corrective action is begun immediately and will be made available in the shortest commercially reasonable time.
- Important:
    - Vulnerabilities that are not rated critical but whose exploitation results in the complete compromise of confidentiality and/or integrity of user data and/or processing resources through user assistance or by authenticated attackers. This rating also applies to those vulnerabilities which could lead to the complete compromise of availability when exploitation is by a remote unauthenticated attacker from the Internet or through a breach of virtual machine isolation.
    - A fix will be delivered as part of the next planned maintenance release of the product and will be released as a patch if appropriate to do so.
- Moderate:
    - Vulnerabilities where the ability to exploit is mitigated to a significant degree by configuration or difficulty of exploitation, but in certain deployment scenarios could still lead to the compromise of confidentiality, integrity, or availability of user data and/or processing resources.
    - A fix will be delivered with the next planned major or minor release of the product.
- Low:
    - All other issues that have a security impact. Vulnerabilities where exploitation is believed to be extremely difficult, or where successful exploitation would have minimal impact.
    - A fix will be delivered with the next planned major or minor release of the product.

The standard release cycle for VMware products is quarterly, so all Moderate and Low findings are typically resolved within a maximum of 90 days, while more significant findings are generally resolved in less time. Both quarterly releases and mid-cycle patches can be obtained from https://customerconnect.vmware.com.

VMware provides an email address (security@vmware.com) that is used for the reporting of potential security findings. VMware encourages the use of Pretty Good Privacy (PGP) to encrypt any communications sent to this email address and provides a copy of their PGP public key at https://kb.vmware.com/s/article/1055.

VMware staff identifies potential vulnerabilities through third-party researchers reporting potential flaws via email, reports from field personnel, reports from customers, and monitoring of public vulnerability sites. When a report is received, VMware attempts to reproduce the finding and determine its severity. If a finding is discovered for which there is no current fix, VMware will publish a Knowledge Base article about the finding as well as any potential workarounds that may be used until an updated version of the product can be delivered.

6.2 Cryptographic Support

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

TSF cryptographic services are implemented by two cryptographic libraries within the TOE boundary. The Java Message Service (JMS) channel between the TOE and an environmental Connection Server uses VMware's BC-FJA (Bouncy Castle FIPS Java API) 1.0.2.3 with JDK 11 to protect data prior to transmission over the JMS channel, and the Blast channel from an inbound Horizon Client (via Unified Access Gateway) uses VMware's OpenSSL FIPS Object Module 2.0.20-vmw. The cryptographic algorithms supplied by the TOE are CAVP 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 Algorithm Claims

Functions		**Standards**	**Certificates**
FCS_CKM.1/AK Cryptographic Asymmetric Key Generation			
ECC key pair generation (NIST curves P-256, P-384, P-521)	OpenSSL	FIPS PUB 186-4	A1292 (OpenSSL)
FFC key pair generation (2048 bit)	BC-FJA	FIPS PUB 186-4	A2841 (BC-FJA)
RSA key pair generation (3072 bit)	BC-FJA	FIPS PUB 186-4	A2841 (BC-FJA)
FCS_CKM.2 Cryptographic Key Establishment			
Elliptic curve-based key establishment	OpenSSL	NIST SP 800-56A	A1292 (OpenSSL)
FCS_COP.1/Hash Cryptographic Operation – Hashing			
SHA-256 and SHA-384 (digest sizes 256 and 384 bits)	OpenSSL	FIPS PUB 180-4	A1292 (OpenSSL)
SHA-256, SHA-384, SHA-512 (digest sizes 256, 384, 512 bits)	BC-FJA		A2841 (BC-FJA)

Functions		Standards	Certificates
FCS_COP.1/KeyedHash Cryptographic Operation – Keyed Hash Message Authentication			
HMAC-SHA-256 and HMAC-SHA-384	OpenSSL	FIPS PUB 198-1 FIPS PUB 180-4	A1292 (OpenSSL)
HMAC-SHA-512	BC-FJA		A2841 (BC-FJA)
FCS_COP.1/Sig Cryptographic Operation – Signing			
RSA (2048, 3072-bit)	OpenSSL	FIPS PUB 186-4, Section 5	A1292 (OpenSSL)
FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption			
AES-GCM (128 bits, 256 bits)	OpenSSL	GCM as defined in NIST SP 800-38D	A1292 (OpenSSL)
AES-CTR (128 bits)	BC-FJA	CTR as defined in NIST SP 800-38A	A2841 (BC-FJA)
FCS_RBG_EXT.2 Random Bit Generation from Application			
Hash_DRBG DRBG (128 bits, 256 bits)	BC-FJA	NIST SP 800-90A NIST SP 800-57	A2841 (BC-FJA)
CTR_DRBG DRBG (128 bits, 256 bits)	OpenSSL		A1292 (OpenSSL)

The TOE generates keys in support of trusted communications. The TSF generates ECC keys using P-256, P-384, and P-521 for ECDHE key establishment schemes for use in TLS/HTTPS communications. The TSF also generates 2048-bit DSA keys for data integrity and 3072-bit RSA keys that are used as credentials for environmental applications. The TOE also generates symmetric keys; it generates 128-bit and 256-bit AES-GCM keys in support of TLS communications. The TOE also generates 128-bit AES-CTR keys that are used as message keys for communications between the TOE and an environmental Connection Server. Message keys are single-use keys that are wrapped using the recipient's public key. To ensure sufficient key strength, the TOE also implements DRBG functionality for key generation, using the AES-CTR_DRBG or Hash_DRBG, depending on OS platform version and the specific cryptographic library that is invoked to perform the key generation function. 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 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. Specifically, random numbers are obtained from the Windows BCryptGenRandom platform API. The Linux platform version of the TOE relies on the Linux kernel implementation of the /dev/random pseudo-device. In both cases, it is assumed that the platform provides at least 256 bits of entropy.

The TOE uses digital signature functions in support of TLS/HTTPS communications.

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

The TOE through BC-FJA uses SHA-256 and SHA-384 in support of digital signatures and SHA-512 in support of password-based key derivation. The TOE through BC-FJA uses HMAC-SHA-512 in support of password-based key derivation. Only the Linux platform version of the TOE uses password-based key derivation.

The TOE uses TLS 1.2 as part of HTTPS for server communications; older SSL and TLS versions are not accepted. The TOE's implementation of TLS conforms to RFC 5246 and its implementation of HTTPS conforms to RFC 2818. The specific TOE network interfaces are documented below in section 6.3. The TOE offers the following ciphersuites in the evaluated configuration:

- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

All supported ciphersuites use elliptic curve ephemeral Diffie-Hellman as the method of key establishment. The TSF presents secp256r1, secp384r1, and secp521r1 in the Supported Groups extension as the parameter used for key establishment. 2048-bit and 3072-bit RSA certificates are supported.

The Windows Agent supports session resumption based on both session IDs (RFC 5246) and session tickets (RFC 5077); the Linux Agent does not support any session resumption.

The TOE's TLS server interface is used for inbound HTTPS connectivity for environmental Horizon Clients to access the TOE for interacting with its underlying host OS platform; these connections are mediated by the environmental Unified Access Gateway.

The TOE maintains two types of credential data: the TOE's own TLS server certificate and RSA key pairs that can be used to issue just-in-time credentials to remote users accessing the TOE platform system through single sign-on. For the Windows platform version of the TOE, the server certificate is stored in the Windows Certificate Store, and the key pairs are stored locally and encrypted using DPAPI. For the Linux platform version of the TOE, both the server certificate and the key pairs are stored using Bouncy Castle's key store. The server certificate is used by OpenSSL but the private key is stored in Bouncy Castle's key store at rest and loaded into a Linux keyring on boot. The Bouncy Castle key store uses the TOE's implementation of PBKDF2 to protect data at rest. This implementation conforms to NIST SP 800-132, uses HMAC-SHA-512 as the pseudorandom function, and executes 16,384 iterations to generate a 256-bit output. The password input is the combination of a master key and random salt generated using the TOE's Hash_DRBG. The resulting 256-bit output is split into a 128-bit AES-CTR key and 128-bit initialization vector. The data that is encrypted and decrypted with this key is stored in the Bouncy Castle key store with the random salt that is used to derive the key for it.

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

- FCS_CKM_EXT.1 – The TOE implements its own cryptographic functionality.
- FCS_CKM_EXT.1/PBKDF/L – The Linux platform version of the TOE uses a password-based key derivation function to protect stored credential data at rest.
- FCS_CKM.1/AK – The TOE uses CAVP validated implementations to generate asymmetric keys in support of TLS communications.
- FCS_CKM.1/SK – The TOE uses its DRBG to generate symmetric keys in support of TLS communications and in encrypting sensitive data.

- FCS_CKM.2 – The TOE performs CAVP validated key establishment in support of TLS communications.
- FCS_COP.1/Hash – The TOE uses CAVP validated implementations to perform cryptographic hashing in support of TLS communications.
- FCS_COP.1/Sig – The TOE uses CAVP validated implementations to generate and verify RSA digital signatures in support of TLS communications.
- FCS_COP.1/KeyedHash – The TOE uses CAVP validated implementations to perform HMAC functions in support of TLS communications and protection of data at rest.
- FCS_COP.1/SKC – The TOE uses CAVP validated implementations to perform AES encryption and decryption in support of TLS communications, to encrypt/decrypt data transmitted over an unprotected channel.
- 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 CAVP validated implementations to generate pseudo-random bits and this implementation is seeded with sufficiently strong entropy collected from the operational environment.
- FCS_STO_EXT.1/L – The Linux platform version of the TOE uses its own cryptographic functions to secure credential data at rest.
- FCS_STO_EXT.1/W – The Windows platform version of the TOE uses platform-provided mechanisms to secure credential data at rest.
- FCS_TLS_EXT.1 – The TOE implements TLS to secure data in transit.
- FCS_TLSS_EXT.1/L – The TOE implements TLS as a server. Specifically for the Linux Agent, session resumption is not supported.
- FCS_TLSS_EXT.1/W – The TOE implements TLS as a server. Specifically for the Windows Agent, session resumption is supported based on both session IDs and session tickets.

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 TSF relies on platform and TOE-provided storage mechanisms identified in both iterations of FCS_STO_EXT.1 to protect all credential data at rest in non-volatile storage. VMware VM Encryption is used to protect log file data at rest as well as the master key used for the PBKDF2 implementation of the Linux client.

The TOE requires system privileges (Local Security Authority and ‘manage system services’ for Windows, root for Linux) to run. As the TOE’s purpose is to make resources on its underlying OS platform available to remote users via the virtual desktop, it will always require use of network connectivity in order to function. Depending on the functionality an individual Agent makes available, the TSF also requires access to the underlying platform’s audio/video interfaces to serve interactive content to remote clients.

It may also require access to the host platform's file system and clipboard. The TOE also accesses the host platform's log facilities to generate audit records of its activity.

The TOE interfaces with external components in its operational environment to satisfy its core functionality. The following network interfaces are present in the TSF:

Table 6: TSF Network Usage

Function	**Invoked By**	**Network Port**	**Secured By**
Connection Server status and event channel	TOE	TCP/4001	No encryption in transit; data is not sensitive
TCP Client access over Blast	UAG	TCP/22443	One-way TLS (TOE acts as HTTPS server)

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 in turn by the platform's use of VMware VM Encryption and the TSF's use of both platform credential storage repositories and its own protection mechanisms.
- 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 initiated from external applications or by the TOE itself.

6.4 Security Management

The TOE is run locally as an application on the host platform. There is no direct local or remote administrative access to the TOE so there are no interactive administrative credentials used for it.

The Windows platform version of the TOE is installed by default to %ProgramFiles%\VMware\VMware View\Agent owned by SYSTEM and the Linux platform version of the TOE is installed by default to /usr/lib/vmware owned by root/root with 755 permissions (i.e. not world-writable). Locally stored security-relevant configuration data consists of the following:

- Supported TLS cipher suites
- Supported TLS groups
- Supported TLS version
- NoManagedCertificate flag (in the evaluated configuration, this flag is enabled to require the TOE to have a certificate signed by a CA)
- Name and public key of Connection Server
- Log level setting

On the Windows platform version of the TOE, this data is stored in the Windows Registry. On the Linux platform version of the TOE, this data is stored in /etc, except for the log level setting which is stored in the user's home directory.

The TOE is managed through the JMS interface from an external Connection Server so it does not provide its own management functionality. Reference the Security Target for VMware Horizon Connection Server 8 2209 (Horizon 8.7) for more details on this interface.

In the evaluated configuration, the TOE is deployed in a VMware vSphere deployment on the same vSphere instance as an authorized Connection Server. An administrator in the TOE's operational environment can use the Connection Server to modify the TOE's configuration through initiation of vSphere commands that can modify the .vmx file of the VM on which the TOE resides. Changes to this file are propagated onto the actual system as modifications to the TOE's configuration files. The Connection Server can also issue management commands directly over the JMS channel between it and the TOE. The Windows platform version of the TOE can also be managed through Active Directory GPOs. However, if these are used, management occurs entirely through the OS platform via direct configuration of registry values.

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

- FMT_CFG_EXT.1 – 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 – The TOE is managed by an external Connection Server.

6.5 Privacy

The TOE's primary function is to serve interactive desktop or application content to a remote Horizon Client based on the specific resources that are authorized to the user who is operating the client. This interface may be used to disclose PII but any such disclosure would be user-supplied and not in the context of a specific prompt for PII (e.g. a user may open an interactive desktop session and willingly enter PII into a text document). Because there is no situation where the TOE requires user-supplied PII to perform its designed functionality, the TSF does not transmit (or require the transmission of) PII over a network.

- FPR_ANO_EXT.1 – The TOE does not transmit PII.

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 /dynamicbase compiler flag in Windows and the –fPIC compiler flag in Linux 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++, and Java (Windows) and C, C++, Java, Python, JavaScript, and shell code (Linux). It is compiled with stack overflow protection through the use of the /GS (Windows) and –fstack-protector (Linux) compiler flags.

The Windows platform version of the TOE is compatible with the security features of Windows Defender Exploit Guard. The Linux platform version of the TOE is compatible with SElinux configured into an enabled and enforcing state. The TOE uses only documented platform APIs. Appendix A.1 lists the APIs used by each platform version of the TOE. The TOE also makes use of third-party libraries. Appendix A.2 lists the libraries used by each platform version of the TOE. The TOE is versioned using both YYMM date-based versioning to correspond to the approximate release of a particular version and major/minor release versioning, e.g. 2209 refers to the TOE version released on or around September of 2022 and is also synonymous with version 8.7; SWID is not used. The TOE is a standalone application that is not natively bundled as part of a host OS.

The TOE does not have automatic checking for updates or automatic updates; updates to both TOE platform versions are obtained through the operational environment (e.g. through the VMware support site). The TOE identifies its version information through the following mechanisms, depending on platform version:

- Windows:
  - TSF-provided: version information is embedded in the binary and can be accessed via API
  - Platform-provided: version information is written to Windows Registry and to log files
- Linux:
  - Platform-provided: version information is stored in a read-only Product.txt file.

The TOE will not download, modify, replace, or update its own binary code. The Windows platform version of the TOE is packaged as an .exe file and the Linux platform version of the TOE is packaged as an RPM. All installation packages are signed by VMware 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 YYMM date-based and major/minor versioning.
- 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 its own cryptographic implementation to encrypt sensitive data in transit. The TOE has the following external interface that relies on trusted communications:

- Between UAG and TOE
  - Communications use TLS/HTTPS (TOE is server)
  - Not mutually authenticated
  - TCP port 22443
  - Used to serve interactive desktop capabilities (file system, application, audio/video, etc. access) to a remote client via UAG

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 sensitive 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, October 7, 2021* (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 Windows or 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. Some 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 6, the 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 5 demonstrates the relationship between security requirements and functions.

Table 5: Security Functions vs. Requirements Mapping

	Cryptographic Support	User Data Protection	Security Management	Privacy	Protection of the TSF	Trusted Path/Channels
FCS_CKM_EXT.1	X					
FCS_CKM_EXT.1/PBKDF/L	X					
FCS_CKM.1/AK	X					
FCS_CKM.1/SK	X					
FCS_CKM.2	X					
FCS_COP.1/Hash	X					
FCS_COP.1/KeyedHash	X					
FCS_COP.1/Sig	X					
FCS_COP.1/SKC	X					
FCS_HTTPS_EXT.1/Server	X					
FCS_RBG_EXT.1	X					
FCS_RBG_EXT.2	X					
FCS_STO_EXT.1/L	X					
FCS_STO_EXT.1/W	X					
FCS_TLS_EXT.1	X					

	Cryptographic Support	User Data Protection	Security Management	Privacy	Protection of the TSF	Trusted Path/Channels
FCS_TLSS_EXT.1/L	X					
FCS_TLSS_EXT.1/W	X					
FDP_DAR_EXT.1		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	
FPT_LIB_EXT.1					X	
FPT_TUD_EXT.1					X	
FPT_TUD_EXT.2					X	
FTP_DIT_EXT.1						X

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 Horizon Agent product.

A.1.1 Windows Platform

ntdll.dll
KERNEL32.DLL
KERNELBASE.dll
ucrtbase.dll
USER32.dll
win32u.dll
GDI32.dll
gdi32full.dll
msvcp_win.dll
ADVAPI32.dll
msvcrt.dll
sechost.dll
RPCRT4.dll
ole32.dll
combase.dll
bcryptPrimitives.dll
OLEAUT32.dll
bcrypt.dll
SHELL32.dll
cfgmgr32.dll
shcore.dll
windows.storage.dll
profapi.dll
powrprof.dll
shlwapi.dll
kernel.appcore.dll
cryptsp.dll
WS2_32.dll
CRYPT32.dll
MSASN1.dll
MSVCP140.dll
VCRUNTIME140_1.dll
VCRUNTIME140.dll
Secur32.dll
WinSCard.dll
VERSION.dll

ncrypt.dll
CRYPTUI.dll
DEVOBJ.dll
NTASN1.dll
SSPICLI.DLL
ntmarta.dll
PSAPI.DLL
dwmapi.dll
clbcatq.dll
WLDAP32.dll
MPR.dll
NETAPI32.dll
ACTIVEDS.dll
adsldpc.dll
NTDSAPI.dll
WININET.dll
WKSCLI.DLL
SAMCLI.DLL
LOGONCLI.DLL
NETUTILS.DLL
MSCTF.dll
WTSAPI32.dll
atlthunk.dll
CRYPTBASE.DLL
msv1_0.DLL
NtlmShared.dll
cryptdll.dll
Windows.Globalization.dll
Bcp47Langs.dll
bcp47mrm.dll
globinputhost.dll
TextInputFramework.dll
CoreUIComponents.dll
PROPSYS.dll
WINSTA.dll
CoreMessaging.dll
wintypes.dll
Windows.UI.dll
InputHost.dll
twinapi.appcore.dll
d2d1.dll
d3d11.dll
MSFTEDIT.DLL
WindowsCodecs.dll

uxtheme.dll
IMM32.DLL
RMCLIENT.dll
dxgi.dll
gdiplus.dll
tdh.dll
IPHLPAPI.DLL
mintdh.dll
SAS.dll
NSI.dll
SAMLIB.dll
rsaenh.dll
IconCodecService.dll
DPAPI.dll
ncryptprov.dll
SETUPAPI.dll
WINTRUST.dll
LINKINFO.dll
schannel.DLL
mskeyprotect.dll
ncryptsslp.dll
dhcpcsvc6.DLL
dhcpcsvc.DLL
DNSAPI.dll
FirewallAPI.dll
fwbase.dll
FWPolicyIOMgr.dll
sxs.dll
pdh.dll
wshqos.dll
wshtcpip.DLL
wship6.dll
rasadhlp.dll
fwpuclnt.dll
kerberos.DLL
perfos.dll
perfdisk.dll
WMICLNT.dll
WINHTTP.dll
msi.dll
DSPARSE.DLL
mswsock.dll
SysNtfy.dll
WINMM.dll

ODBC32.dll
WINMMBASE.dll
rasadhlp.dll
qwave.dll
TRAFFIC.dll
NtlmShared.dll
newdev.dll
DEVRTL.dll
napinsp.dll
winrnr.dll
NLAapi.dll
wshbth.dll
DSOUND.dll
dxva2.dll
DBGHELP.DLL
dbgcore.DLL
gpapi.dll
paho-mqttas.dll

A.1.2 Linux Platform

awk
adduser
base64
cat
cd
chcon
checkmodule
chgrp
chkconfig
chmod
chown
comm
cp
cut
dconf
depmod
dhclient
dmesg
dmidecode
domainname
dpkg
echo
egrep
exec
exit
export

expr
find
flock
fusermount
gdbus
gdm3
gpasswd
grep
gsettings
head
hostname
iconv
id
ifconfig
initctl
install
ip
iptables
kill
kinit
ktutil
kvno
ldconfig
ldd
ln
loginctl
ls
lsmod
lpadmin
lspci
mkdir
modinfo
modprobe
mount
mountpoint
mv
net
netstat
openssl
pactl
pax11publish
pbis
pidof
ping
pkaction
printf
prlimit
ps

read
reboot
restorecon
rm
rotate
rpm
sed
semanage
semodule
service
setfacl
setsebool
setxkbmap
shutdown
sleep
sort
ss
stat
stty
su
sysctl
systemctl
tail
tar
tdbtool
tee
test
touch
tr
trap
type
typeset
udevadm
ulimit
umask
umount
unlink
unset
useradd
userdel
usermod
uuid
vmtoolsd
wbinfo
wc
which
xargs
xauth

xinput
xprop
xrandr

A.2 Third-Party Libraries

Listed below are the third-party libraries used by the Horizon Agent product.

A.2.1 Windows Platform

Library	Version (If applicable)
openssl	1.0.2zg
bc-fips	1.0.2.3
openjdk-11-BellSoft	11.0.17
speex	1.2.rc3
libexpat	2.5.0
curl	7.88
x264	157.6.1
xerces	2.12.2
org.slf4j:slf4j-api	1.7.30
commons-lang	3.8.1
io.jsonwebtoken:jjwt	0.9.1
hidapi	0.9.12
libxml2	2.10.3
libopus	1.3.1
Intel IPP	8.2.1
libjpeg-turbo	2.1.0
org.apache.logging.log4j:log4j-slf4j-impl	2.17.2
7zip	22.0.1
log4cxx	0.11
beamr h264 codec sdk	5.0.0.4
libyuv	r1788
boost	1.78.2101
org.bouncycastle:bcpkix-jdk15on	1.68
org.apache.logging.log4j:log-api	2.17.2
org.bouncycastle:bcprov-jdk15on	1.68
zlib	1.2.13
libwebrtc	90
dxsdk	N/A
commons-net:commons-net	3.9
org.apache.logging.log4j:log4j-core	2.17.2
xml-apis	1.4.01
snappy	1.1.7
libpng	1.6.37
libogg	1.3.2
gson	2.2.2
libyuv	r1714
libiconv	1.14

nvdia-grid-sdk	N/A
jsoncpp	1.2.1
glib	2.71.1
glibmm	2.70.0
libtheora	1.1.1.115
guava-30.1-jre-jar	30.1
SwiftMQ JMS/Mq	9.3.1
Libidn.dll, SecurityGateway.exe (from Teradici)	PSG-2.23
json-java	20220924

A.2.2 Linux Platform

Library	Version (If applicable)
bc-fips	1.0.2.3
openjdk-11-BellSoft	11.0.17
openssl	1.0.2zg
libappindicator	12.10.0
atk	2.28.1
at-spi2-atk	2.10.2
at-spi2-core	2.10.2
atkmm	2.22.7
boost	1.78
cairo	1.17.4
cairomm	1.10.0
cvt	1.1
libfuse	2.9.4
gdk-pixbuff	2.36.11
glib	2.68.3
glibmm	2.70
gtk	3.14.15
gtkmm	3.10.1
ffmpeg	4.4.1
juds	0.95
libevent	12.10.0
libffi	3.3-rc2
libffi	3.2.1
pango	1.40.13
pangomm	2.34.0
pcre	8.44
png	1.6.37
protobuf	3.18.3
protobuf-c	1.3.3
libsigc++	2.4.1
libstdc++	6.0.28
uriparser	0.9.6
libXScrnSaver	1.2.2
x264	157.6.1

zlib	1.2.13