Appgate SDP Client 6.4

Security Target

Version 1.0

10 January 2025

Prepared for:

Appgate Cybersecurity, Inc.
2 Alhambra Plaza, Suite PH-1-B,
Coral Gables, FL 33134

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 0)
- 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 – Appgate SDP Client 6.4 Security Target

ST Version – Version 1.0

ST Date – 10 January 2025

TOE Identification – Appgate SDP Client 6.4

TOE Developer – Appgate Cybersecurity, Inc.

Evaluation Sponsor – Appgate Cybersecurity, 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.1/SK
    - FCS_CKM.2
    - FCS_COP.1/SKC
    - FCS_COP.1/Hash
    - FCS_COP.1/Sig
    - FCS_COP.1/KeyedHash
    - 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, March 1, 2019* (TLS Package) with the following optional and selection-based SFRs:
    - FCS_TLSC_EXT.1
    - FCS_TLSC_EXT.2
    - FCS_TLSC_EXT.3
    - FCS_TLSC_EXT.5
- 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 not applicable to the TOE because it applies to an SFR the TOE does not claim.

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

TD0628: Addition of Container Image to Package Format

- This TD is applicable to the TOE.

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.

TD0664: Testing activity for FPT_TUD_EXT.2.2

- 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 not applicable to the TOE. The ST does not claim the SFRs that are affected by the TD.

TD0736: Number of elements for iterations of FCS_HTTPS_EXT.1

- This TD is not applicable to the TOE. The ST does not claim the SFRs that are affected by the TD.

TD0739: PKG_TLS_V1.1 has 2 different publication dates

- This TD is not applicable to the TOE; this TD addresses an inconsistency between the HTML and PDF versions of the package but the ST does not claim the SFRs that are affected by the TD.

TD0743: FTP_DIT_EXT.1.1 Selection exclusivity

- This TD is applicable to the TOE.

TD0747: Configuration Storage Options for Android

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

TD0756: Update for platform-provided full disk encryption

- This TD is applicable to the TOE.

TD0770: TLSS.2 connection with no client cert

- This TD is not applicable to the TOE; the TD applies to FCS_TLSS_EXT.2, which the TOE does not claim.

TD0779: Updated Session Resumption Support in TLS package V1.1

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

TD0780: FIA_X509_EXT.1 Test 4 Clarification

- This TD is applicable to the TOE.

TD0798: Static Memory Mapping Exceptions

- This TD is applicable to the TOE.

TD0815: Addition of Conditional TSS Activity for FPT_AEX_EXT.1.5

- This TD is applicable to the TOE.

TD0822: Correction to Windows Manifest File for FDP_DEC_EXT.1

- This TD is applicable to the TOE.

TD0823: Update to Microsoft Windows Exploit Protection link in FPT_AEX_EXT.1.3

- This TD is applicable to the TOE.

TD0844: Addition of Assurance Package for Flaw Remediation V1.0 Conformance Claim

- This TD is applicable to the TOE but the ST does not claim the optional assurance package.

TD0865: Consistency of Cryptographic Key Sizes

- This TD is applicable to the TOE but the ST does not claim the selection that the TD applies to.

TD0893: Addition of Recommended Configuration Locations for Windows in FMT_MEC_EXT.1.1

- 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_CKM.1/AK (for "asymmetric keys") indicates that the FCS_CKM.1 requirement applies specifically to the generation of asymmetric keys.
    - 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.
- The ST does not show refinement operations that have been completed by the PP authors; refinements are used only to show where the ST author has refined text from the PP.

1.3.1 Terminology

The following terms and abbreviations are used in this ST:

Table 1: Terms and Definitions

Term	Definition
Client	An end user application that is used to access protected resources.
Controller	A network endpoint that brokers Client connectivity to a Gateway.
Gateway	A network endpoint that acts as a boundary between a trusted and untrusted network. Clients access protected resources through a Gateway.
Single Packet Authorization	A method for connection establishment where a secure connection cannot even be attempted until evidence is provided that the originator of the connection attempt is valid.
Software Defined Perimeter	The overall Appgate system, designed to provide zero trust access to organizational resources.

1.3.2 Acronyms

Table 2: Acronyms

Term	Definition
AD	Active Directory
ASLR	Address Space Layout Randomization
DRBG	Deterministic Random Bit Generator
ECC	Elliptic Curve Cryptography
FD	File Descriptor
IdP	Identity Provider
OIDC	OpenID Connect
PII	Personally Identifiable Information
PP	Protection Profile
SAN	Subject Alternative Name
SDP	Software Defined Perimeter
SPA	Single Packet Authorization
ST	Security Target
SWID	Software Identification (standard)
TOE	Target of Evaluation
TSF	TOE Security Functionality
UAG	Unified Access Gateway
VPN	Virtual Private Network

2 Product and TOE Description

2.1 Introduction

Appgate Software Defined Perimeter (SDP) is a zero-trust network access solution where users are granted access to enterprise resources using a granular entitlements model. This is done using a single packet authorization (SPA) protocol implemented on top of TLS that allows for the entry point to protected resources to be hidden from public scanning and through the use of granular entitlements to apply additional access control beyond access to the enterprise network itself.

For this Security Target, the Target of Evaluation (TOE) is the Appgate SDP Client application, specifically version 6.4. This is the application that resides on the end user device that is used to access these enterprise resources.

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

2.2 Product Overview

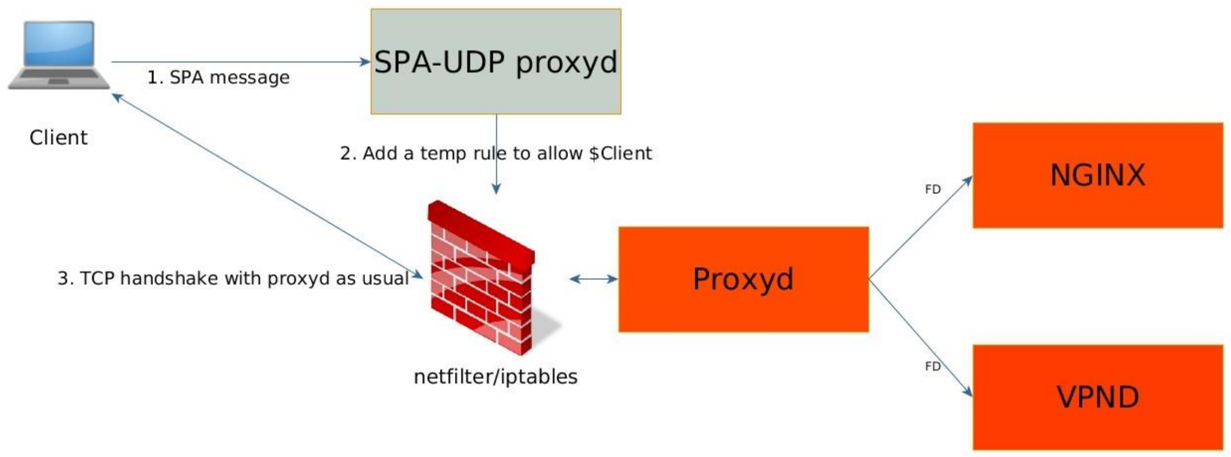
Appgate is a suite of products that control end user access to organizational resources at both the protocol and application layer. At the application layer, Appgate functions as a TLS virtual private network (VPN) that builds a secure tunnel to an organizational network's entry point. Access to resources within the network is based on an encrypted authorization token that is presented over the secure tunnel.

Appgate SDP consists of several components:

- Clients are applications that are installed on end user devices and used to access protected network resources.
- Controllers are network appliances that are used to broker the initial Client connection to a Gateway. A Controller issues the Client an authorization token, which attests that the Client's requested access to resources is authorized, and a TLS client certificate that allows the Client authentication to the Gateway to take place. The Controller functions as a Certification Authority (CA) for the issuance of these certificates. The certificate infrastructure for this operates as a closed loop, independent of organizational PKI.
- Gateways are network appliances that act as the interface to a protected network. Gateways only permit access to Clients that first successfully connected to Controllers to receive the requisite authorizations.

The SPA functionality of Appgate can be implemented solely over TCP or through a combination of UDP and TCP. UDP+TCP SPA uses a crafted UDP packet sent to a Controller or Gateway to request an inbound connection. If the UDP packet is recognized as a valid SPA packet, the Controller or Gateway will apply a short-lived firewall rule allowing access from the same source over TCP port 443. This relationship is shown in figure 1 below, where a Client interacts with a Controller or Gateway's proxy service (shown in grey) before authorization is given to communicate with the actual backend destination of the TLS connection (shown in orange). Within daemons, this is handled through the file descriptor (FD), i.e. UNIX domain socket.

Figure 1 – SPA UDP Data Flow


SPA-UDP proxyd
1. SPA message
Client
2. Add a temp rule to allow $Client
3. TCP handshake with proxyd as usual
netfilter/iptables
Proxyd
FD
NGINX
FD
VPND

The Client then sends a TLS Client Hello that includes SPA information as part of a custom extension (defined by RFC 5246 section 7.4.1.4). If this SPA extension is valid, a standard TLS 1.2 connection is then established. Communications between the Client and the Controller use one-way authentication, while communications between the Client and the Gateway use mutually-authenticated TLS, with the Client certificate being issued by the Controller as part of the initial connectivity. Authorization tokens, which are used to grant access to specific resources on the protected network, are also issued by the Controller and validated by the Gateway; these are handled at the application layer separate from any TLS connectivity.

UDP+TCP SPA is used in the evaluated configuration to facilitate connectivity to environmental IT entities because it is inseparable from the core TLS protocol functionality; the TLS protocol functionality conforms to the TLS Package as written.

2.3 TOE Overview

The Target of Evaluation (TOE) is the Appgate SDP Client 6.4 application. The specific tested builds of the TOE were 6.4.2-39991 (Windows) and 6.4.2-40808 (macOS). All references to “Appgate SDP Client” throughout the ST refer to this specific version. The TOE includes the Windows and macOS platform versions of this application. The user-facing functionality for both platform versions is fundamentally the same.

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 within the scope of the TOE:

- **Protection of sensitive data at rest:** the TOE leverages secure platform storage and encryption mechanisms to protect credentials and other sensitive data at rest.
- **Protection of data in transit:** the TOE secures data in transit between itself and its operational environment using TLS.
- **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 wolfCrypt with CAVP validated algorithm services that it uses to secure data in transit.
- **Self-protection:** the TOE is designed to interact with its underlying host operating system platform in a secure manner. This includes the use of anti-exploitation techniques that prevent the TOE from being used as way to access system memory, the invocation of well-defined platform APIs to ensure that platform functions and data are being accessed in their intended manner, and the use of documented third-party libraries so that the TOE developer is responsive to potential security flaws introduced by those libraries.

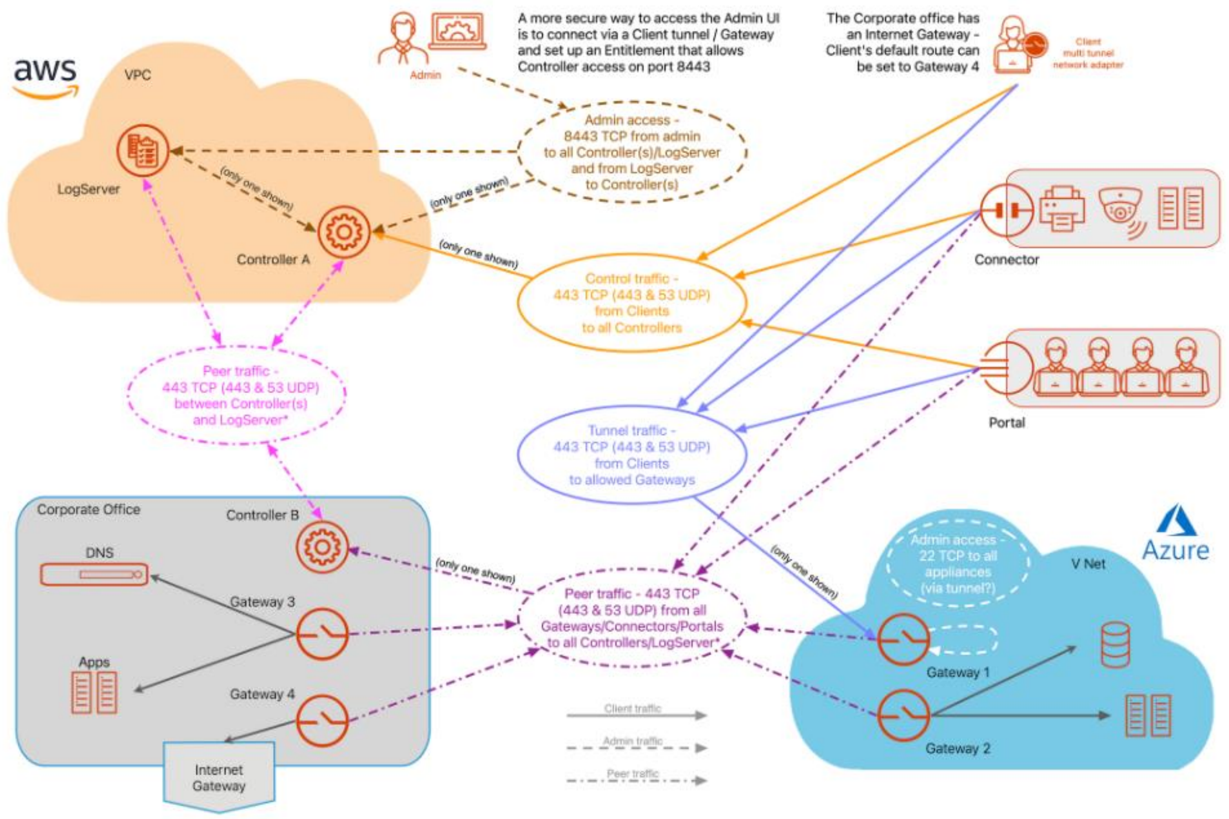
Notably, there are no standardized security requirements in the claimed PP (or any other published PP at the time of this ST's publication) for application layer authentication and authorization. Therefore, the security of these interfaces is only assessed with respect to the ability of the TOE to protect these communications from unauthorized modification or disclosure. Similarly, the SPA implementation is outside the scope of the evaluation but is present in the TOE as non-interfering functionality as successful SPA authorization is a necessary condition of negotiating a TLS connection.

2.4 TOE Architecture

The Appgate SDP Client TOE consists of the Appgate SDP Client application. The TOE has both Windows and macOS platform versions. For both platform versions, the UI is written in Javascript using the React framework, business logic is written in C# using .NET, and the driver is written in C and Rust. Third-party components used by the TOE (see Appendix A.2) are either linked into the TOE binaries or exist as standalone DLLs, depending on the library.

A sample real-world deployment of the TOE in a larger organizational environment is shown in Figure 2 below. Organizational users with Client applications connect to one or more Controllers (solid line to orange) which then allows connectivity to corporate resources that exist either on-premise or in the cloud (solid line to blue, peer dashed lines to purple). Admin traffic and peer traffic relate to the configuration of the Controller and Gateway components, which is outside the scope of the Client.

Figure 2 – Sample Appgate Environment


aws
VPC
LogServer
Controller A
Admin
A more secure way to access the Admin UI is to connect via a Client tunnel / Gateway and set up an Entitlement that allows Controller access on port 8443
The Corporate office has an Internet Gateway - Client's default route can be set to Gateway 4
Client multi tunnel network adapter
Admin access - 8443 TCP from admin to all Controller(s)/LogServer and from LogServer to Controller(s)
(only one shown)
Control traffic - 443 TCP (443 & 53 UDP) from Clients to all Controllers
Connector
Portal
Peer traffic - 443 TCP (443 & 53 UDP) between Controller(s) and LogServer*
Tunnel traffic - 443 TCP (443 & 53 UDP) from Clients to allowed Gateways
Corporate Office
Controller B
DNS
Gateway 3
Apps
Gateway 4
Internet Gateway
Peer traffic - 443 TCP (443 & 53 UDP) from all Gateways/Connectors/Portals to all Controllers/LogServer*
Azure
V Net
Admin access - 22 TCP to all appliances (via tunnel?)
Gateway 1
Gateway 2
Client traffic
Admin traffic
Peer traffic

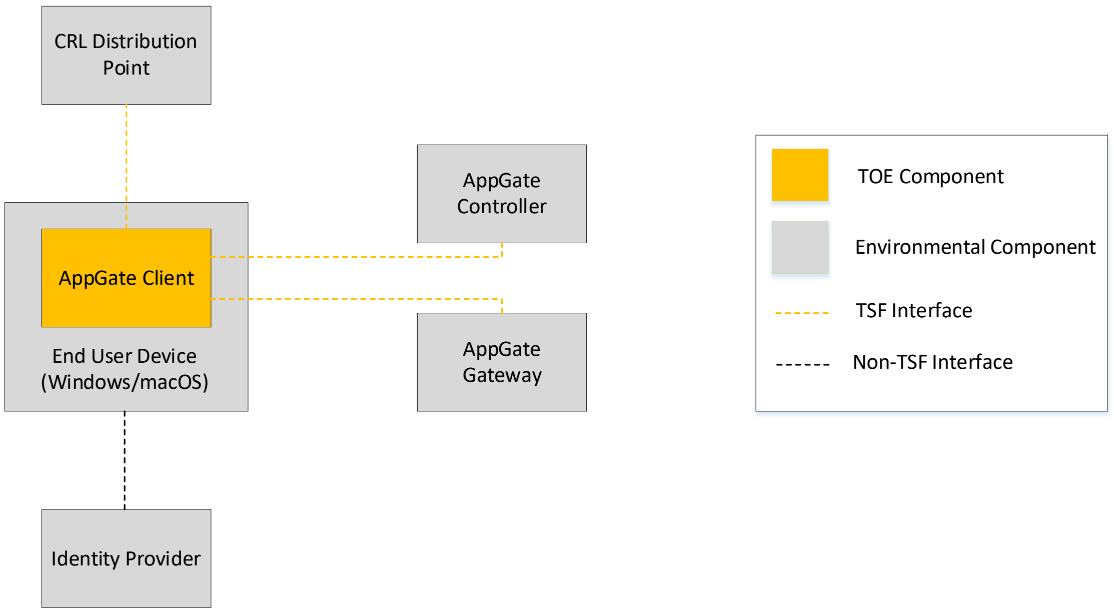
2.4.1 Physical Boundary

The TOE consists of the following component, as shown in Figure 3 below:

- Appgate SDP Client 6.4

Figure 3 shows the TOE in a high-level architecture with respect to its external interfaces. Interfaces external to environmental components (e.g. log servers, remote administration, protected resources) are not shown. An Appgate deployment supports multiple Controllers and multiple Gateways but the interfaces to these are identical. A CRL distribution point is required to validate the revocation status of TLS server certificates presented by the Appgate Controller and Gateway components. An Identity Provider (IdP) is used to provide third-party verification of user identity (e.g. when accessing resources that require authentication). When an OIDC IdP is used, the TOE facilitates IdP interaction via browser redirection, so the actual interface to the IdP is performed by the underlying platform; the Client passes any assertion returned by the IdP to the Controller via its single interface with it. When an AD IdP is used, the Controller itself interfaces with the IdP for user identity verification.

Figure 3 - TOE Boundary



Connectivity to the Controller uses TLS without client authentication. Connectivity to the Gateway uses mutually-authenticated TLS.

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

- Windows:
  - Windows 11
  - 64-bit Intel hardware
  - 400MB disk space
  - 256MB RAM
- macOS:
  - macOS 14.4
  - Apple M1 hardware
  - 600MB disk space
  - 256MB RAM

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

- TCP/443 (for TCP SPA and TLS connectivity)
- UDP/53 (for initial user session via UDP SPA)

The TOE's operational environment includes the following:

- Other Appgate components (at least one each of Controller and Gateway).
- Identity Provider (AD or OIDC)
- Platform (hardware and software) on which the TOE is hosted.
  - The TOE is capable of running on a general-purpose Windows or macOS operating system on standard consumer-grade hardware using Apple M1 and 64-bit Intel

processors. For the evaluated configuration, the TOE was tested on the following environments:

- Windows: Windows 11 using an Intel® Core™ i7-1255U @ 1.70G GHz with PAA (Alder Lake)
- macOS: MacOS 14.4 using an Apple M1 Max processor with PAA

- Full disk encryption is required for the TOE platform to ensure adequate data-at-rest protection.
- Access to a Certification Authority and corresponding revocation checking mechanism is needed to validate presented X.509 certificates.

The TOE boundary excludes any logical functionality that does not enforce or support the SFR claims made in section 5 of this ST. Non-interfering functionality is present if it is an integral part of the TOE's operation. Specifically, the TOE uses UDP+TCP SPA as the first step in opening a connection to a Controller or Gateway. This functionality is a prerequisite to establishing a TLS channel (which is part of the TSF) but it does not relate to any SFR claims so it is non-interfering with respect to security. Similarly, any authorization tokens are outside the TOE boundary with respect to their ability to grant access to resources since the claimed PP does not have requirements for access control in that manner. However, since the tokens themselves are encrypted using 256-bit AES, the TOE's ability to encrypt and decrypt these tokens is examined through the claim of FCS_COP.1/DataEncryption.

2.4.2 Logical Boundary

This section summarizes the security functions provided by the TOE:

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

2.4.2.1 Cryptographic Support

The TOE implements cryptography to protect data in transit. For data in transit, the TOE implements TLS as a client. The TOE supports TLS connections both with and without mutual authentication.

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

For data at rest, the TOE relies on its operational environment to control access to stored credential data.

2.4.2.2 User Data Protection

The TOE relies on platform full disk encryption and credential storage mechanisms to protect sensitive data at rest.

The TOE relies on the network connectivity, credential repository, and log functions of its host OS platform. All uses of network connectivity are user-initiated.

2.4.2.3 Identification and Authentication

The TOE supports X.509 certificate validation as part of establishing TLS connections. The TOE implements functionality to support various certificate validity checking methods, including the checking of certificate revocation status using CRL. If the validity status of a certificate cannot be determined, the certificate will be rejected.

2.4.2.4 Security Management

The TOE itself and the configuration settings it uses are stored in locations recommended by the platform vendor. The TOE is launched by an authenticated OS user and runs in the session context of that user; there is no interface for a non-administrator to act as an administrator through separate authentication. When connecting to a Controller in the operational environment, the TOE receives a policy which may restrict the user to a subset of the available management functions. The TOE's primary management function is to add and remove the Controller-provided profiles and to configure behavior related to credential storage.

2.4.2.5 Privacy

The TOE does not have an interface to request or transmit requested PII from a user; PII is only transmitted over the network if initiated by the user.

2.4.2.6 Protection of the TSF

The TOE enforces various mechanisms to protect itself against unauthorized modification and use. 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 are acquired through the application's connection to the Controller in the operational environment and subsequently applied using the TOE platform. All updates are digitally signed to guarantee their authenticity and integrity.

2.4.2.7 Trusted Path/Channels

The TOE encrypts sensitive data in transit between itself and its operational environment using TLS. These interfaces are used to secure all data in transit between the TOE and its operational environment.

2.5 TOE Documentation

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

- Appgate SDP User Guide, version 6.4
- Appgate SDP Client 6.4 Common Criteria Evaluated Configuration Guide (CCECG), Version 1.0, November 6, 2024

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 an executable 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_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 Package:

- FCS_TLS_EXT.1 TLS Protocol
- FCS_TLSC_EXT.1 TLS Client Protocol

- FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication
- FCS_TLSC_EXT.3 TLS Client Support for Signature Algorithms Extension
- 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.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_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/C TLS Client Protocol (Controller) (TLS Package)
	FCS_TLSC_EXT.1/G TLS Client Protocol (Gateway) (TLS Package)
	FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication (TLS Package)
	FCS_TLSC_EXT.3 TLS Client Support for Signature Algorithms Extension (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

Requirement Class	**Requirement Component**
	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.1/AK Cryptographic Asymmetric Key Generation[2]

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]

].

5.2.1.3 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 [

- 256 bit

].

[1] Modified by TD0717

[2] Modified by TD0717

5.2.1.4 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”]

].

5.2.1.5 FCS_COP.1/Hash Cryptographic Operation – Hashing[3]

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

- SHA-384
- SHA-512

] and message digest sizes [

- 384
- 512

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

5.2.1.6 FCS_COP.1/KeyedHash Cryptographic Operation – Keyed-Hash Message Authentication[4]

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

- HMAC-SHA-384

] and [

- no other algorithms

] with key sizes [*384 bits*] and message digest sizes [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’].

[3] Modified by TD0717

[4] Modified by TD0717

5.2.1.7 FCS_COP.1/Sig Cryptographic Operation – Signing[5]

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, 3072-bit, 4096-bit**] that meet the following: [FIPS PUB 186-4, "Digital Signature Standard (DSS)", Section 5]

].

5.2.1.8 FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption[6]

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 [256-bit].

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 [Hash_DRBG (any)].

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] Modified by TD0717

[6] Modified by TD0717

5.2.1.11 FCS_STO_EXT.1 Storage of Credentials

FCS_STO_EXT.1.1 The application shall [

- invoke the functionality provided by the platform to securely store [*TLS client certificate private key, OIDC refresh token, onboarding cookie, AD credential, cached PIN (Windows only)*]

] 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/C TLS Client Protocol (Controller) (TLS Package)[7]

FCS_TLSC_EXT.1.1/C 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_256_GCM_SHA384 as defined in RFC 5289

] and also supports functionality for [

- none

].

FCS_TLSC_EXT.1.2/C The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

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

- with no exceptions

].

5.2.1.14 FCS_TLSC_EXT.1/G TLS Client Protocol (Gateway) (TLS Package)[8]

FCS_TLSC_EXT.1.1/G 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_256_GCM_SHA384 as defined in RFC 5289

[7] Modified by TD0442

[8] Modified by TD0442

] and also supports functionality for [

- mutual authentication

].

FCS_TLSC_EXT.1.2/G The product shall verify that the presented identifier matches the reference identifier according to RFC 6125.

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

- with no exceptions

].

5.2.1.15 FCS_TLSC_EXT.2 TLS Client Support for Mutual Authentication (TLS Package)

FCS_TLSC_EXT.2.1 The product shall support mutual authentication **for FCS_TLSC_EXT.1/G** using X.509v3 certificates.

Application Note: ***This SFR only applies to the interface between the TOE and the environmental Gateway.***

5.2.1.16 FCS_TLSC_EXT.3 TLS Client Support for Signature Algorithms Extension (TLS Package)

FCS_TLSC_EXT.3.1 The product shall present the signature_algorithms extension in the Client Hello with the supported_signature_algorithms value containing the following hash algorithms: [SHA384, SHA512] and no other hash algorithms.

5.2.1.17 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 [

- secp256r1,
- secp384r1,
- secp521r1

].

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 [

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

] 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 [

- network connectivity

].

FDP_DEC_EXT.1.2 The application shall restrict its access to [

- system logs,
- [*system credential repository*]

].

5.2.2.3 FDP_NET_EXT.1 Network Communications

FDP_NET_EXT.1.1 The application shall restrict network communication to [

- user-initiated communication for [*initiating Controller connection, initiating Gateway connection, initiating service connection (via Gateway), initiating OIDC redirect*]

].

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 [implement functionality] 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, 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 [CRL as specified in RFC 5280 Section 6.3, CRL as specified in RFC 8603].
- 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.

Application Note: *There are no cases where the TOE is presented with a code signing, TLS client, S/MIME, OCSP, or EST certificate in its evaluated configuration so certificates designated for these purposes will never be accepted by the TSF.*

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 [TLS].

FIA_X509_EXT.2.2 When the application cannot establish a connection to determine the validity of a certificate, the application shall [not 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's 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 [

- [add and remove profiles,
- select active profile,
- Enable/disable PIN caching (Windows only)]

].

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 [

- <u>not transmit PII over a network</u>

].

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 [*cryptographic module*].

FPT_AEX_EXT.1.2 The application shall [

- <u>not allocate any memory region with both write and execute permissions</u>

].

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 [[<u>*major.minor.release-build versioning*</u>]].

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 substantial 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 [provide the ability] to check for updates and patches to the application software.

FPT_TUD_EXT.1.2 The application shall [provide the ability] 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 [as an additional software package to the platform OS].

5.2.6.6 FPT_TUD_EXT.2 Integrity for Installation and Update[9]

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.

5.2.7 Trusted Path/Channels (FTP)

5.2.7.1 FTP_DIT_EXT.1 Protection of Data in Transit[10]

FTP_DIT_EXT.1.1 The application shall [

- encrypt all transmitted [data] with [TLS as a client as defined in the Functional Package for TLS for [*protection of traffic between the TOE and an AppGate Controller in the operational environment (per FCS_TLSC_EXT.1/C), mutually-authenticated protection of traffic between the TOE and an AppGate Gateway in the operational environment (per FCS_TLSC_EXT.1/G)*]]

] 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.

[9] Modified by TD0628

[10] Modified by 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

Appgate uses a classification system to categorize product security flaws by severity level based on CVSS scoring (https://nvd.nist.gov/vuln-metrics/cvss). 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 gain access to protected network resources or execute arbitrary code on the user's operating system that operates the Appgate client or on the Appgate appliance itself.
  - 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.
  - 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, or already with one of the next patch releases.
- 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, or already with one of the next patch releases.

The standard release cycle for Appgate Major or Minor versions is typically done every 6 months, patch versions are typically released once a Month, 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 Major and Minor releases and mid-cycle patches can be obtained for desktop Clients from https://www.appgate.com/support/software-defined-perimeter-support.

Appgate provides an HTTPS form on their corporate website that is used for the reporting of potential security findings.

The Appgate staff identifies potential vulnerabilities through third-party researchers reporting potential flaws via email, our own security pen testers, reports from field personnel, reports from customers, and monitoring of public vulnerability sites. When a report is received, Appgate attempts to reproduce the finding and determine its severity. If a finding is discovered for which there is no current fix, Appgate 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 the wolfCrypt cryptographic library included within the TOE boundary. The Windows version of the TOE uses wolfCrypt 5.2.1 and the macOS version of the TOE uses wolfCrypt 5.2.3. 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	**Libraries**	**Standards**	**Certificates**
FCS_CKM.1/AK Cryptographic Asymmetric Key Generation			
ECC key pair generation (NIST curve P-256, P-384, P-521)	wolfCrypt	FIPS PUB 186-4	#A5866 (macOS) #A5865 (Windows)
RSA key pair generation (2048-bit, 3072-bit, 4096-bit)	wolfCrypt	FIPS PUB 186-4	#A5866 (macOS) #A5865 (Windows)
FCS_CKM.1/SK Cryptographic Symmetric Key Generation			
N/A – symmetric key generation is performed using the DRBG; refer to FCS_RBG_EXT.2 below.			
FCS_CKM.2 Cryptographic Key Establishment			
Elliptic curve-based key establishment	wolfCrypt	NIST SP 800-56A	#A5866 (macOS) #A5865 (Windows)
FCS_COP.1/Hash Cryptographic Operation – Hashing			
SHA-384 (digest size 384 bits) SHA-512 (digest size 512 bits)	wolfCrypt	FIPS PUB 180-4	#A5866 (macOS) #A5865 (Windows)
FCS_COP.1/KeyedHash Cryptographic Operation – Keyed Hash Message Authentication			
HMAC-SHA-384 (digest size 384 bits, key size 256 bits, block size 128 bits)	wolfCrypt	FIPS PUB 198-1 FIPS PUB 180-4	#A5866 (macOS) #A5865 (Windows)

Functions	Libraries	Standards	Certificates
FCS_COP.1/Sig Cryptographic Operation – Signing			
RSA (2048-bit, 3072-bit, 4096 bit)	wolfCrypt	FIPS PUB 186-4, Section 4	#A5866 (macOS) #A5865 (Windows)
FCS_COP.1/SKC Cryptographic Operation – Encryption/Decryption			
AES-GCM (256 bits) AES-CTR (256 bits)	wolfCrypt	GCM as defined in NIST SP 800-38D CTR as defined in NIST SP 800-38A	#A5866 (macOS) #A5865 (Windows)
FCS_RBG_EXT.2 Random Bit Generation from Application			
Hash_DRBG (256 bits)	wolfCrypt	NIST SP 800-90A NIST SP 800-57	#A5866 (macOS) #A5865 (Windows)

The TOE generates asymmetric keys in support of trusted communications and authentication. The TOE generates ECDHE keys in support of the ECDHE key establishment scheme used for TLS communications. The TOE generates RSA keys in support of X.509 certificate signing request generation. The TOE generates 256-bit AES keys; AES-GCM for TLS, and AES-CTR for encryption and decryption of authorization tokens at the application layer. To ensure sufficient key strength, the TOE also implements DRBG functionality for key generation, using a 256-bit Hash_DRBG. The proprietary Entropy Analysis Report (EAR) describes how the TSF extracts random data from hardware-based or software-based sources, depending on platform, 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. In the evaluated configuration, the Windows platform version of the TOE runs on an Intel processor that supports the hardware RDSEED interface, which functions as the platform noise source from the TOE. The macOS version of the TOE obtains its entropy from the platform pseudorandom number generator, which uses the Apple M1 processor noise source.

Specifically, random numbers are obtained from the following platform APIs, depending on the platform used:

- Windows: RDSEED
- macOS: invocation of /dev/urandom and /dev/random pseudo-devices (falls back to the blocking /dev/random source if insufficient entropy is available in /dev/urandom)

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

The TOE uses TLS 1.2 as a client for secure communications with environmental IT entities. The TOE's implementation of TLS conforms to RFC 5246. The specific TOE network interfaces are documented below in section 6.3. The TLS client offers the following cipher suite in its evaluated configuration:

- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289

The supported ciphersuite uses 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. By default, the TSF also presents SHA-384 and SHA-512 in the signature_algorithms extension as the hash parameter used for digital signatures. If a presented server certificate is invalid, the TSF will automatically reject it. In the evaluated configuration, 2048-bit, 3072-bit, or 4096-bit RSA certificates are configured for TLS servers with which the TOE communicates.

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 comparison of the DNS name or IP address presented in the Subject Alternative Name (SAN) certificate field to the hostname of the server. Wildcards are not supported. Certificate pinning is not supported.

The TOE supports TLS client functionality for connectivity to an Appgate Controller and Gateway in its operational environment. Initial connectivity with the Controller is used to obtain an authorization token for access to resources behind a Gateway. It is also used to obtain an authentication credential to the Gateway in the form of an X.509 client certificate issued to it by the Controller. The initial TLS connection to the Controller does not use mutual authentication, while the subsequent Gateway connection does. These are only used for initial connectivity and not persistently stored.

The TOE relies on platform-provided storage mechanisms for credential data. Both the Windows and macOS platform versions store the private key for the TLS client certificate. The client certificate and corresponding private key reside in the Windows Certificate Store on Windows and the Keychain on macOS.

The TOE optionally supports the use of an environmental IdP for user authentication. In the evaluated configuration, either of AD or an OIDC compatible IdP is used. When AD is used, an AD credential is stored. When OIDC is used, a refresh token is stored. On Windows, the AD credential is stored in the Windows Credential Manager and the refresh token is stored using DPAPI. On macOS, both credentials are stored in the Keychain. The TOE also stores an onboarding cookie for a given Controller connection to verify that a successful connection was previously made to that Controller. For the Windows Client, if PIN caching for certificate authentication is enabled, the PIN is cached in the Windows Credential Manager.

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.1/AK – The TOE uses a CAVP validated implementation to generate asymmetric keys in support of TLS communications and X.509 certificate signing request generation.
- FCS_CKM.1/SK – The TOE uses its DRBG to generate symmetric keys used for AES.
- FCS_CKM.2 – The TOE performs CAVP validated key establishment in support of TLS communications.
- FCS_COP.1/Hash – The TOE uses a CAVP validated implementation to perform cryptographic hashing in support of TLS communications.
- FCS_COP.1/KeyedHash – The TOE uses a CAVP validated implementation to perform HMAC functions in support of TLS communications.
- FCS_COP.1/Sig – The TOE uses a CAVP validated implementation to generate and verify RSA digital signatures in support of TLS communications.
- FCS_COP.1/SKC – The TOE uses a CAVP validated implementation to perform AES encryption and decryption in support of TLS communications and for encrypting and decrypting an authorization token used for application layer access control.

- FCS_RBG_EXT.1 – The TOE implements its own random bit generation services.
- FCS_RBG_EXT.2 – The TOE uses a CAVP validated implementation to generate pseudo-random bits and this implementation is seeded with sufficiently strong entropy collected from the operational environment.
- FCS_STO_EXT.1 – The TOE uses platform-provided mechanisms to secure credential data at rest.
- FCS_TLS_EXT.1 – The TOE implements TLS to secure data in transit.
- FCS_TLSC_EXT.1/C – The TOE implements TLS as a client without mutual authentication for communications to the environmental Controller.
- FCS_TLSC_EXT.1/G – The TOE implements TLS as a client with mutual authentication for communications to the environmental Gateway.
- FCS_TLSC_EXT.2 – The TOE's TLS client implementation supports mutual authentication for some TLS functions.
- FCS_TLSC_EXT.3 – The TOE's TLS client implementation presents supported hash algorithms to the server in the signature_algorithms extension.
- FCS_TLSC_EXT.5 – The TOE's TLS client implementation presents supported elliptic curves to the server in the Supported Groups extension.

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 storage mechanisms identified in FCS_STO_EXT.1 to protect credential data at rest in non-volatile storage. Other sensitive data consists of system log data that is generated by the application. For that data, platform-provided full disk encryption is used to protect data at rest.

The TOE has access to physical and logical system resources from the host platform. For physical resources, the TOE always requires network connectivity to satisfy its intended purpose. For logical resources, the TOE accesses the platform logging function and repositories for stored credential data (Windows Credential Manager, Certificate Store, and DPAPI; macOS Keychain).

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
Connectivity to Controller	User	UDP/53 UDP/443	N/A
		TCP/443	TLS (TOE acts as a client)
Connectivity to Gateway (both initial connectivity	User	UDP/53 UDP/443	N/A
		TCP/443	Mutual TLS (TOE acts as client)

and service connectivity)			
Connectivity to OIDC IdP*	User	TCP/29001 localhost**	N/A

* when AD is used as the IdP, connectivity is entirely between the Controller and AD so any interaction between AD and the user goes over the existing TLS channel.

** the TOE itself does not use this port; the TOE initiates a browser redirect so that the user can use their browser to be authenticated by the IdP. The assertion returned by the IdP is subsequently transmitted to the Controller via the TLS interface. This is therefore user-initiated in the sense that the interface is invoked as a result of the user attempting to use the TOE to authenticate to a Controller.

Both the Controller and Gateway interfaces use the same UDP+TCP SPA connection as a prerequisite to allowing a TLS connection. This works as follows for both interfaces:

1. UDP SPA
    a. Client sends UDP DNS packet (port 53) and UDP DTLS Client Hello packet (port 443) to destination.
    b. The DTLS Client Hello packet is parsed as a DNS lookup request, from which data about the Client is extracted. The DTLS Client Hello is used as the vehicle for the SPA data; no actual DTLS connection attempt occurs.
    c. A successful SPA attempt allows the destination to open a temporary firewall rule on TCP port 443 for the same source as the original SPA request to allow a TLS connection to be attempted.
2. TCP SPA
    a. Client sends TLS 1.2 Client Hello message which includes a 94 byte extension of type 0x0043 (an unreserved extension type per RFC 5246).
    b. Regardless of the rest of the Client Hello, the TLS Server Hello will not be returned by the destination unless the TCP SPA extension is present and valid.

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 the platform's use of full disk encryption and by the TSF's use of platform credential storage repositories.
- 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 user-initiated directly through the TSF or indirectly as a consequence of using the TOE.

6.4 Identification and Authentication

The TOE uses X.509 to validate the TLS server certificates of the Appgate components that it communicates with (Controller and Gateway). The TOE implements its own X.509 validation functionality for this behavior, as follows:

- Certificate validation and certificate path validation is performed in accordance with RFC 5280.

- The certificate path is checked to ensure that it terminates with a trusted CA certificate.
- The certificate path is validated by ensuring the presence of the basicConstraints extension, that the CA flag is set to TRUE for all CA certificates, and that any path constraints are met.
- Any CA certificate is validated by ensuring that the key usage field includes the caSigning purpose.
- Revocation status is checked using CRL (RFC 5280 section 6.3, RFC 8603).
- 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.

The TOE supports a maximum trust depth of two in the certificate chain. In the event that the revocation status of a certificate cannot be verified (i.e. the CRL cannot be reached), the certificate is treated as if it is invalid. This behavior is automatic and not configurable.

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 client certificate that the TOE presents to a Gateway when performing TLS mutual authentication is issued by a Controller. The connection to the Gateway is always mutually authenticated; this is the default behavior of the TOE and is not configurable.

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 will accept or reject the certificate based on configuration.

6.5 Security Management

The TOE is run locally as an application on the host platform. A user must input a valid credential to be authenticated by the system as part of accessing protected resources, but this credential is supplied to the server and is based on an organizational credential (i.e. a user defined by an organization's IdP). The TOE itself is launched in the context of the user session on the host OS platform so no separate authentication is required to access the application itself.

The Windows platform version of the TOE is installed by default to %program files%\Appgate SDP. All configuration data is stored in %appdata%\Appgate using the .NET User.config file.

The macOS platform version of the TOE is installed by default to /Applications/Appgate SDP. All configuration data is stored using NSUserDefaults.

Management of the TOE can be performed by local users through the TOE itself. The security-relevant management functions are specified below.

Table 7: Management Functions

Management Function	**Purpose**
Add and remove profiles	Allows the user to define different connection settings for different accounts.
Select active profile	Allows the user to switch between profiles as needed when multiple profiles are available.
Pin caching (Windows only)	Saves PIN for certificate-based user authentication if enabled.

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 includes limited management functionality for connecting to environmental entities.

6.6 Privacy

The TOE's primary function is to facilitate a user's remote access to enterprise computing resources. The user may use the TOE to access resources that may require the input of PII, but any mechanism to do this would be user-initiated; the TSF is not designed or intended for the capture of PII. The risk of inadvertent PII disclosure is assumed by the user through consent to allow the TOE to access computing interfaces over which PII could be transmitted if the user initiated an operation to do so (whether intentionally or not). The TOE accepts credentials from the user that are used by the operational environment to validate the user's identity in order to grant them access to their authorized enterprise computing resources, but user account information is not considered to be PII.

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

- FPR_ANO_EXT.1 – The TOE does not have an interface to request PII from a user; PII is only transmitted over the network if initiated by the user.

6.7 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 on Windows (enabled by default for .NET) and the -fPIE compiler flag on macOS, and it relies fully on its underlying host platforms to perform memory mapping. The only exception to the use of dynamic address space is for the cryptographic module, which is a linked library that requires a static mapping for integrity self-testing. The TOE also does not use both PROT_WRITE and PROT_EXEC on the same memory regions. The TOE UI is written in Javascript using the React framework, business logic is written in C# using .NET, and the driver is written in C and Rust. Stack overflow protection is enforced through the use of the /GS compiler flag on Windows and -fstack-protector-all on macOS.

The Windows platform version of the TOE is compatible with the security features of Windows Defender Exploit Guard. Similarly, the macOS version of the TOE can run as intended without disabling any platform security features. 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 a major.minor.release-build versioning scheme (e.g., "6.4.2-39991-release"); SWID is not used. The TOE is a standalone application that is not natively bundled as part of a host OS. The current running version of the application can be checked through the application UI, under About.

The TOE checks for updates to itself via its connection to the Controller. Any time a new login to the controller is performed or when an entitlement token expires (24 hours by default), a check is made over the TCP/443 interface to the Controller to determine if an updated application is available. Once acquired, installation is handled through the OS platform itself. 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 macOS platform version of the TOE is packaged as a DMG. All installation packages are signed by Appgate 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 only documented platform APIs.
- FPT_IDV_EXT.1 – The TOE uses a well-defined versioning scheme.
- 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

In the evaluated configuration, the TOE uses its own cryptographic implementation to encrypt data in transit. Data is transmitted to the Controller and Gateway using an identical mechanism, except that Gateway connectivity is mutually authenticated (i.e. the TOE provides a TLS client certificate to the Gateway). In both cases, TLS is used over TCP port 443. A UDP SPA that uses both UDP ports 53 and 443 is used to allow the TCP port 443 SYN to be accepted. The initial TLS connection includes a TCP SPA presented as a custom TLS extension 0x0043; acceptance of the SPA is a prerequisite to establishment of the TLS connection, which otherwise conforms to RFC 5246.

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 all 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, March 1, 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 macOS 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 0, 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 8 demonstrates the relationship between security requirements and functions.

Table 8: 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.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_RBG_EXT.1	X						
FCS_RBG_EXT.2	X						
FCS_STO_EXT.1	X						
FCS_TLS_EXT.1	X						
FCS_TLSC_EXT.1/C	X						
FCS_TLSC_EXT.1/G	X						
FCS_TLSC_EXT.2	X						
FCS_TLSC_EXT.3	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			

	Cryptographic Support	User Data Protection	Identification and Authentication	Security Management	Privacy	Protection of the TSF	Trusted Path/Channels
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 Appgate SDP Client product.

A.1.1 macOS Platform

Service
libsystem_asl.dylib
Security.framework
libdl
libSystem
CoreGraphics
Foundation
Foundation.NSUserDefaults
Metal
Microsoft
Microsoft.Bcl.AsyncInterfaces
Microsoft.NETCore.Platforms
Microsoft.NETCore.Targets
Microsoft.VisualStudio.Threading.Analyzers
NuGetDefense
ObjCRuntime
Security.SecStatusCode.Success
System.Action
System.AppContext
System.Buffers
System.Data
System.Diagnostics
System.Diagnostics.Contracts
System.Diagnostics.Debug
System.Diagnostics.Tools
System.Dynamic.Runtime
System.Enum
System.Environment.NewLine.ToCharArray
System.Exception
System.Globalization
System.Globalization.Calendars
System.Globalization.Extensions
System.Int32.MaxValue
System.IO
System.IO.FileSystem.Primitives
System.Net
System.Numerics.Vectors
System.OperatingSystem.IsMacOS
System.OperatingSystem.IsWindows

System.Reflection
System.Reflection.Extensions
System.Reflection.TypeExtensions
System.Resources.ResourceManager
System.Runtime.CompilerServices
System.Runtime.Extensions
System.Runtime.Handles
System.Runtime.Serialization
System.Runtime.Versioning.TargetFrameworkAttribute
System.Runtime.WindowsRuntime
System.Security
System.Security.AccessControl
System.Security.Claims
System.Security.Cryptography.Algorithms
System.Security.Cryptography.Cng
System.Security.Cryptography.Csp
System.Security.Cryptography.Encoding
System.Security.Cryptography.OpenSsl
System.Security.Cryptography.Primitives
System.Security.Cryptography.X509Certificates
System.Security.Principal
System.Security.Principal.Windows
System.Text
System.Text.Encoding
System.Threading.Tasks
System.Threading.Tasks.Extensions
System.Threading.Timer
System.ValueTuple
System.Version
System.Web
System.Xml.ReaderWriter
System.Xml.XmlDocument

Driver
Carbon.framework

Both service and driver
SystemConfiguration.framework
CoreFoundation.framework
IOKit.framework
AppKit

A.1.2 Windows Platform

Service

kernel32.lib

netapi32.lib
system32.lib
winbrand.lib
wininet.lib
wscapi.lib
Microsoft
Microsoft.Bcl.AsyncInterfaces
Microsoft.NETCore.Platforms
Microsoft.NETCore.Targets
Microsoft.VisualStudio.Editors.SettingsDesigner.SettingsSingleFileGenerator
Microsoft.VisualStudio.Threading.Analyzers
Microsoft.Win32
NuGetDefense
runtime.native.System
runtime.native.System.IO.Compression
runtime.native.System.Net.Http
runtime.native.System.Security.Cryptography.Apple
runtime.native.System.Security.Cryptography.OpenSsl
System.Action
System.AppContext
System.Buffers
System.CodeDom.Compiler.GeneratedCodeAttribute
System.Configuration
System.Data
System.Diagnostics
System.Diagnostics.Contracts
System.Diagnostics.Debug
System.Diagnostics.Tools
System.Dynamic.Runtime
System.Enum
System.Globalization
System.Globalization.Calendars
System.Globalization.Extensions
System.Int32.MaxValue
System.IO
System.IO.FileSystem
System.IO.FileSystem.Primitives
System.Net
System.OperatingSystem.IsMacOS
System.OperatingSystem.IsWindows
System.Reflection
System.Reflection.Extensions
System.Reflection.TypeExtensions
System.Resources.ResourceManager
System.Runtime.CompilerServices
System.Runtime.CompilerServices.Unsafe
System.Runtime.Extensions
System.Runtime.Handles

System.Runtime.InteropServices
System.Runtime.Serialization
System.Runtime.Versioning
System.Runtime.WindowsRuntime
System.Security
System.Security.Cryptography.Algorithms
System.Security.Cryptography.Cng
System.Security.Cryptography.Csp
System.Security.Cryptography.Encoding
System.Security.Cryptography.OpenSsl
System.Security.Cryptography.Primitives
System.Security.Cryptography.X509Certificates
System.ServiceProcess
System.Text
System.Text.Encoding
System.Threading.Overlapped
System.Threading.Tasks
System.Threading.Tasks.Extensions
System.Threading.Timer
System.TypeCode
System.ValueTuple
System.Version
System.Volume.BitLockerProtection
System.Web
System.Windows.Extensions

Driver
crypt32.lib
dbghelp.lib
dnsapi.lib
fwpuclnt.lib
iphlpapi.lib
ole32.lib
propsys.lib
psapi.lib
shell32.lib
version.lib
windowsapp.lib

Both service and driver
advapi32.lib
bcrypt.lib
ntdll.lib
user32.lib
userenv.lib
wintrust.lib
wtsapi32.lib
ws2_32.lib

A.2 Third-Party Libraries

Listed below are the third-party libraries used by the Appgate SDP Client product.

A.2.1 Service

A.2.1.1 macOS Platform

CommandLine.dll
CommandLineParser
INIFileParserDotNetCore.dll
libclrgc.dylib
libclrjit.dylib
libcoreclr.dylib
libempty-pkcs11-universal.dylib
libfido2
libhostfxr.dylib
libhostpolicy.dylib
libMono.Unix.dylib
libmscordaccore.dylib
libmscordbi.dylib
libopensc.12.dylib
libSystem.Globalization.Native.dylib
libSystem.IO.Compression.Native.dylib
libSystem.Native.dylib
libSystem.Net.Security.Native.dylib
libSystem.Security.Cryptography.Native.Apple.dylib
libSystem.Security.Cryptography.Native.OpenSsl.dylib
libwolfssl.42.dylib
MessagePack.Annotations.dll
MessagePack.dll
Microsoft.CSharp.dll
Microsoft.macOS.dll
Microsoft.NET.StringTools.dll
Microsoft.VisualStudio.Threading.dll
Microsoft.VisualStudio.Threading.resources.dll
Microsoft.VisualStudio.Validation.dll
Microsoft.VisualStudio.Validation.resources.dll
Microsoft.Win32.Primitives.dll
Microsoft.Win32.Registry.dll
Mono.Unix.dll
Nerdbank.Streams.dll
netstandard.dll
Newtonsoft.Json.dll
Pkcs11Interop.dll
StreamJsonRpc.dll

StreamJsonRpc.resources.dll
System.Collections.Concurrent.dll
System.Collections.dll
System.Collections.Immutable.dll
System.Collections.NonGeneric.dll
System.Collections.Specialized.dll
System.ComponentModel.dll
System.ComponentModel.Primitives.dll
System.ComponentModel.TypeConverter.dll
System.Console.dll
System.Data.Common.dll
System.Diagnostics.DiagnosticSource.dll
System.Diagnostics.FileVersionInfo.dll
System.Diagnostics.Process.dll
System.Diagnostics.StackTrace.dll
System.Diagnostics.TraceSource.dll
System.Diagnostics.Tracing.dll
System.dll
System.Drawing.dll
System.Drawing.Primitives.dll
System.Formats.Asn1.dll
System.IO.Compression.dll
System.IO.Compression.ZipFile.dll
System.IO.FileSystem.dll
System.IO.MemoryMappedFiles.dll
System.IO.Pipelines.dll
System.IO.Pipes.dll
System.Linq.dll
System.Linq.Expressions.dll
System.Memory.dll
System.Net.Http.dll
System.Net.NameResolution.dll
System.Net.NetworkInformation.dll
System.Net.Primitives.dll
System.Net.Quic.dll
System.Net.Requests.dll
System.Net.Security.dll
System.Net.Sockets.dll
System.Net.WebSockets.dll
System.Numerics.Vectors.dll
System.ObjectModel.dll
System.Private.CoreLib.dll
System.Private.Uri.dll
System.Private.Xml.dll

System.Private.Xml.Linq.dll
System.Reflection.Emit.dll
System.Reflection.Emit.ILGeneration.dll
System.Reflection.Metadata.dll
System.Reflection.Primitives.dll
System.Runtime.CompilerServices.Unsafe.dll
System.Runtime.dll
System.Runtime.InteropServices.dll
System.Runtime.InteropServices.RuntimeInformation.dll
System.Runtime.Intrinsics.dll
System.Runtime.Loader.dll
System.Runtime.Numerics.dll
System.Runtime.Serialization.Formatters.dll
System.Runtime.Serialization.Primitives.dll
System.Security.Cryptography.dll
System.Text.Encoding.Extensions.dll
System.Text.Encodings.Web.dll
System.Text.Json.dll
System.Text.RegularExpressions.dll
System.Threading.Channels.dll
System.Threading.dll
System.Threading.Tasks.Dataflow.dll
System.Threading.Thread.dll
System.Threading.ThreadPool.dll
System.Web.HttpUtility.dll
System.Xml.Linq.dll
System.Xml.XDocument.dll
treebitmap
zlib

A.2.1.2 Windows Platform

clretwrc.dll
clrgc.dll
clrjit.dll
CommandLine.dll
CommandLineParser
coreclr.dll
CredentialManagement.dll
hostfxr.dll
hostpolicy.dll
libfido2
MessagePack.Annotations.dll
MessagePack.dll
Microsoft.CSharp.dll

Microsoft.DiaSymReader.Native.amd64.dll
Microsoft.NET.StringTools.dll
Microsoft.VisualStudio.Threading.dll
Microsoft.VisualStudio.Threading.resources.dll
Microsoft.VisualStudio.Validation.dll
Microsoft.VisualStudio.Validation.resources.dll
Microsoft.Win32.Primitives.dll
Microsoft.Win32.Registry.dll
Microsoft.Win32.SystemEvents.dll
mscordaccore.dll
mscordaccore_amd64_amd64_8.0.1124.51707.dll
mscordbi.dll
mscorlib.dll
mscorrc.dll
msquic.dll
Nerdbank.Streams.dll
netstandard.dll
Newtonsoft.Json.dll
Optional.dll
StreamJsonRpc.dll
StreamJsonRpc.resources.dll
System.Collections.Concurrent.dll
System.Collections.dll
System.Collections.Immutable.dll
System.Collections.NonGeneric.dll
System.Collections.Specialized.dll
System.ComponentModel.dll
System.ComponentModel.EventBasedAsync.dll
System.ComponentModel.Primitives.dll
System.ComponentModel.TypeConverter.dll
System.Configuration.ConfigurationManager.dll
System.Console.dll
System.Core.dll
System.Data.Common.dll
System.Diagnostics.DiagnosticSource.dll
System.Diagnostics.EventLog.dll
System.Diagnostics.FileVersionInfo.dll
System.Diagnostics.Process.dll
System.Diagnostics.StackTrace.dll
System.Diagnostics.TextWriterTraceListener.dll
System.Diagnostics.TraceSource.dll
System.Diagnostics.Tracing.dll
System.DirectoryServices.AccountManagement.dll
System.DirectoryServices.dll

System.DirectoryServices.Protocols.dll
System.dll
System.Drawing.Common.dll
System.Drawing.dll
System.Drawing.Primitives.dll
System.Formats.Asn1.dll
System.IO.Compression.Brotli.dll
System.IO.Compression.dll
System.IO.Compression.Native.dll
System.IO.Compression.ZipFile.dll
System.IO.FileSystem.AccessControl.dll
System.IO.MemoryMappedFiles.dll
System.IO.Pipelines.dll
System.IO.Pipes.AccessControl.dll
System.IO.Pipes.dll
System.Linq.dll
System.Linq.Expressions.dll
System.Memory.dll
System.Net.Http.dll
System.Net.NameResolution.dll
System.Net.NetworkInformation.dll
System.Net.Primitives.dll
System.Net.Quic.dll
System.Net.Requests.dll
System.Net.Security.dll
System.Net.ServicePoint.dll
System.Net.Sockets.dll
System.Net.WebClient.dll
System.Net.WebHeaderCollection.dll
System.Net.WebSockets.dll
System.Numerics.Vectors.dll
System.ObjectModel.dll
System.Private.CoreLib.dll
System.Private.Uri.dll
System.Private.Xml.dll
System.Private.Xml.Linq.dll
System.Reflection.Emit.dll
System.Reflection.Emit.ILGeneration.dll
System.Reflection.Metadata.dll
System.Reflection.Primitives.dll
System.Runtime.CompilerServices.Unsafe.dll
System.Runtime.dll
System.Runtime.InteropServices.dll
System.Runtime.InteropServices.RuntimeInformation.dll

System.Runtime.Intrinsics.dll
System.Runtime.Loader.dll
System.Runtime.Numerics.dll
System.Runtime.Serialization.Formatters.dll
System.Runtime.Serialization.Primitives.dll
System.Security.AccessControl.dll
System.Security.Claims.dll
System.Security.Cryptography.dll
System.Security.Cryptography.ProtectedData.dll
System.Security.Permissions.dll
System.Security.Principal.Windows.dll
System.ServiceProcess.ServiceController.dll
System.Text.Encoding.Extensions.dll
System.Text.Encodings.Web.dll
System.Text.Json.dll
System.Text.RegularExpressions.dll
System.Threading.Channels.dll
System.Threading.dll
System.Threading.Tasks.Dataflow.dll
System.Threading.Thread.dll
System.Threading.ThreadPool.dll
System.Web.HttpUtility.dll
System.Xml.Linq.dll
System.Xml.ReaderWriter.dll
System.Xml.XDocument.dll
System.Xml.XmlSerializer.dll
treebitmap
WmiLight.dll
WmiLight.Native.dll
wolfssl-fips.dll
zlib

A.2.2 Driver

A.2.2.1 macOS Platform

WolfSSL
cJSON
libuv
zlib
eglib
treebitmap

A.2.2.2 Windows Platform

Wolfssl-fips.dll

cJSON
libuv
Wintun
zlib
eglib
treebitmap

A.2.3 User Interface

A.2.3.1 macOS Platform

@colors/colors
@fingerprintjs
@fontsource/roboto
@types/uuid
classnames
core-js
css-loader
dompurify
electron-context-menu
electron-store
history
native-reg
node-polyglot
Query-string
react
react-dom
react-loading-skeleton
react-responsive
react-router-dom
react-slick
sass
sass-loader
semver
slick-carousel
style-loader
swipe-js-iso
ua-parser-js
uuid
winston
winston-daily-rotate-file
Yargs

libEGL.dylib
libffmpeg.dylib
libGLESv2.dylib
libvk_swiftshader.dylib

A.2.3.2 Windows Platform

@colors/colors
@fingerprintjs
@fontsource/roboto
@types/uuid
classnames
core-js
css-loader
dompurify
electron-context-menu
electron-store
history
native-reg
node-polyglot
Query-string
react
react-dom
react-loading-skeleton
react-responsive
react-router-dom
react-slick
sass
sass-loader
semver
slick-carousel
style-loader
swipe-js-iso
ua-parser-js
uuid
winston
winston-daily-rotate-file
Yargs

d3dcompiler_47.dll
libEGL.dll
ffmpeg.dll
libGLESv2.dll
vk_swiftshader.dll
vulkan-1.dll