Unisys Stealth Solution Release v3.3 Windows Endpoint Security Target

Version 1.1
10 October 2017

Prepared for:

UNISYS

801 Lakeview Drive
Blue Bell, PA 19422

Prepared By:

leidos

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

Table of Contents

List of Tables

1. Introduction

This section introduces the Target of Evaluation (TOE) and provides the Security Target (ST) and TOE identification, ST and TOE conformance claims, ST conventions, glossary and list of abbreviations.

The TOE is the Unisys Stealth™ Solution Release 3.3 Windows Endpoint. The Stealth endpoint for Windows operating systems provides capabilities for protected transmission of private data between Stealth-enabled IPsec VPN endpoints.

The Security Target contains the following additional sections:

- TOE Description (Section 2)—provides an overview of the TOE and describes the physical and logical boundaries of the TOE
- Security Problem Definition (Section 3)—describes the threats and assumptions that define the security problem to be addressed by the TOE and its environment
- Security Objectives (Section 4)—describes the security objectives for the TOE and its operational environment necessary to counter the threats and satisfy the assumptions that define the security problem
- IT Security Requirements (Section 5)—specifies the security functional requirements (SFRs) and security assurance requirements (SARs) to be met by the TOE
- TOE Summary Specification (Section 6)—describes the security functions of the TOE and how they satisfy the SFRs
- Protection Profile Claims (Section 7)—provides rationale for the consistency of the ST with the PPs to which conformance is claimed
- Rationale (Section 8)—provides mappings and rationale for the security problem definition, security objectives, security requirements, and security functions to justify their completeness, consistency, and suitability.

1.1 Security Target, TOE and CC Identification

ST Title – Unisys Stealth Solution Release 3.3 Windows Endpoint Security Target

ST Version – 1.1

ST Date – 10 October 2017

TOE Identification – Unisys Stealth Solution Release 3.3 Windows Endpoint (Version 3.3.011.0)

TOE Developer – Unisys Corporation

Evaluation Sponsor – Unisys Corporation

CC Identification – Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4, September 2012.

1.2 Conformance Claims

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

- Common Criteria for Information Technology Security Evaluation Part 2: Security Functional Components, Version 3.1, Revision 4, September 2012
  - Part 2 Extended
- Common Criteria for Information Technology Security Evaluation Part 3: Security Assurance Components, Version 3.1, Revision 4, September 2012
  - Part 3 Conformant

This ST and the TOE it describes are conformant to the following Protection Profile:

- *Protection Profile for IPsec Virtual Private Network (VPN) Clients*, Version 1.4, 21 October 2013, with CSfC selections for VPN Clients applied. The following NIAP Technical Decisions apply to this PP and have been accounted for in the ST development and the conduct of the evaluation:
    - TD0053: Removal of FCS_IPSEC_EXT.1.12 Test 5 from VPN IPSEC Client v1.4
    - TD0037: IPsec Requirement_DN Verification

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. In this ST, iteration is indicated by a number in parentheses placed at the end of the component. For example, FMT_MTD.1(1) and FMT_MTD.1(2) indicate that the ST includes two iterations of the FMT_MTD.1 requirement.
    - Assignment: allows the specification of an identified parameter. Assignments are indicated using bold and are surrounded by brackets (e.g., [**assignment**]). Note that an assignment within a selection would be identified in italics and with embedded bold brackets (e.g., [***[selected-assignment]***]).
    - Selection: allows the specification of one or more elements from a list. Selections are indicated using bold italics and are surrounded by brackets (e.g., [***selection***]).
    - Refinement: allows the addition of details. Refinements are indicated using bold for additions and double strike-through for deletions (e.g., "… **all** objects …" or "… ~~some~~ **big** things …").
- Other sections of the ST—other sections of the ST use bolding to highlight text of special interest, such as captions.

1.4 Glossary

This ST uses a number of terms that have a specific meaning within the context of the ST and the TOE. This glossary provides a list of those terms and how they are to be understood within this ST.

Community of Interest (COI)	The TOE enables multiple "secure communities" to share the same network without fear of another group accessing their data or their workstations and servers. These are referred to as Communities of Interest (COIs).
Secure Community of Interest Protocol (SCIP)	A Unisys proprietary protocol used in conjunction with IPsec for communication among all Stealth endpoints.
Stealth administrator	An administrator that uses the Stealth Enterprise Manager to configure Stealth endpoint installation packages.
Stealth endpoint	A server or workstation running Stealth endpoint software. The TOE is stealth endpoint software specifically for Windows platforms.

1.5 Abbreviations and Acronyms

The following abbreviations and acronyms are used in this document. A brief definition is provided for abbreviations that are potentially unfamiliar, are specific to the TOE, or not obviously self-explanatory.

AES	Advanced Encryption Standard
CA	Certificate Authority
CBC	Cipher Block Chaining—an AES mode of operation
CC	Common Criteria
COI	Community of Interest (refer to Glossary for definition)
CRL	Certificate Revocation List
CSP	Critical Security Parameter—security-related information whose disclosure or modification can compromise the security of a cryptographic module
DH	Diffie-Hellman
DN	Distinguished Name
ECDSA	Elliptic Curve Digital Signature Algorithm
ESP	Encapsulating Security Payload—a member of the IPsec protocol suite providing origin authenticity, integrity and confidentiality protection of packets
FIPS	Federal Information Processing Standard
GCM	Galois/Counter Mode—an AES mode of operation
HMAC	Keyed-Hash Message Authentication Code
IKE	Internet Key Exchange
IPsec	Internet Protocol security
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
RFC	Request For Comments—an Internet Engineering Task Force memorandum on Internet standards and protocols
SAR	Security Assurance Requirement
SCIP	Secure Community of Interest Protocol (refer to Glossary for definition)
SPD	Security Policy Database—packet filtering rules in an IPsec implementation
SHA	Secure Hash Algorithm
SFR	Security Functional Requirement
ST	Security Target
TCP/IP	Transmission Control Protocol/Internet Protocol
TOE	Target of Evaluation
TSF	TOE Security Functionality
VPN	Virtual Private Network
WFP	Windows Filtering Platform
XML	Extensible Markup Language

2. TOE Description

The TOE is the Unisys Stealth Solution Release 3.0 Windows Endpoint. It is the component of the Unisys Stealth Solution that enables endpoint devices to establish secure IPsec tunnels with each other.

2.1 Overview

The Unisys Stealth Solution is an enterprise networking security solution that utilizes IPsec to protect the confidentiality of data transmitted between devices on the enterprise network. Stealth enables multiple "secure communities" to share the same network while maintaining separation of data and devices. These are referred to as Communities of Interest (COIs).

Stealth achieves this through implementation of the proprietary Secure Community of Interest Protocol (SCIP) combined with standard IPsec. The SCIP/IPsec protocol is used to negotiate COI membership and cryptographic profiles in order to establish COI-based tunnels between Stealth-enabled endpoints. SCIP also controls the OS-native IKE and IPsec implementations used to setup, rekey, and transport data through encrypted tunnels. The operation of SCIP, and the subsequent configuration and management of the resulting IPsec tunnels, is completely transparent to the user and any applications running on the system.

The TOE comprises software installed on Windows-based servers and workstations that enables these devices to participate in the Stealth network as Stealth-enabled endpoints. The TOE functions as an IPsec VPN client that enables the endpoint on which it is installed to establish an IPsec tunnel with another Stealth-enabled endpoint belonging to the same Stealth COI. Note that the TOE implements a client-to-client model of operation—Stealth-enabled endpoints establish IPsec tunnels with each other rather than with a VPN gateway.

2.2 Stealth Architecture

The architecture of the Stealth solution encompasses the following components:

- Enterprise Manager—a centrally-managed, distributed set of services that allows administrators to create and manage COIs, provision and monitor geographically dispersed components, and authorize, monitor, and license Stealth users. This software runs on the Management Server.
- Endpoints—each Stealth-enabled endpoint has a Stealth agent installed. Endpoint software is responsible for communicating with the Stealth Authorization Service to authorize users, claim a license, and log Stealth-related events. The Stealth Authorization Service is part of the Enterprise Manager and runs on the Management Server. Optional standalone Authorization Servers are also supported but are not covered by the scope of the evaluation. Endpoints also manipulate the OS-native packet handling capabilities to remain "dark" on the network, and to otherwise allow or deny specific traffic flows. Note that endpoint software is also installed on the Management Server and any standalone Authorization Servers.
- Stealth Secure Virtual Gateway—operating systems for which there is no native Stealth agent available can participate in Stealth networks through the use of a Secure Virtual Gateway. The Secure Virtual Gateway is placed topologically in front of the systems to be protected, with as small a clear-text segment between those systems and the Secure Virtual Gateway as possible. Each of the clear-text systems is configured within the Secure Virtual Gateway with its own user credentials. The user credentials determine, through the normal Stealth authorization method, into what Role the clear-text system should be authorized.
- Secure Remote Access Gateway—systems that need to participate in an enterprise's Stealth network, but are physically located outside of the enterprise's intranet, can connect through a Secure Remote Access Gateway. As with the Secure Virtual Gateway, each remote device or wrapped application is represented as a unique Stealth endpoint within the enterprise. The differences are that remote endpoints are not statically configured, their Stealth Roles are determined by credentials passed in from the remote endpoint, and the connection from the remote endpoint to the SRA Gateway is a secure IPsec VPN tunnel.
- Stealth(aware)- automatically discovers network traffic and creates an interactive map of the network topology that enables you to deploy a new Stealth configuration quickly and easily in your network.

Stealth(aware) simplifies the deployment of Stealth onto your network, compared to exclusively using Enterprise Manager. Stealth(aware) functionality is supported but is not covered by the scope of the evaluation.

Note, the scope of this evaluation is specifically the Unisys Stealth Solution Release 3.0 Windows Endpoint. As discussed further below, the Enterprise Manager is required in the operational environment, but other Stealth components described above are not required and are not described further in this ST.

2.3 TOE Architecture

The TOE comprises an endpoint installation package that is created by the Enterprise Manager software in the TOE's operational environment. The Enterprise Manager can create endpoint packages specific for both 32-bit and 64-bit Windows platforms. The installation package includes the Stealth endpoint software and configuration information that specifies the IKE and IPsec cryptographic profiles[1] the TOE will use when negotiating an IPsec tunnel with another endpoint.

The configuration information is contained in two XML files—"*protectionprofile.xml*" and "*crypto.xml*". The *protectionprofile.xml* file contains the settings the administrator configures to enable Stealth to conform to the Protection Profile for IPsec VPN Clients.

The *protectionprofile.xml* file is used in the process of creating endpoint packages. During the endpoint package generation process, the Enterprise Manager reads the *protectionprofile.xml* file, validates it, and inserts the contents into the *crypto.xml* file. Enterprise Manager signs the resulting *crypto.xml* file using a signing certificate and includes it in the endpoint package. The resulting endpoint package is then installed on the endpoints.

The endpoints on which the package is installed read the *crypto.xml* file and validate the signature of the signing certificate using the associated trusted root certificate, which is also installed on these endpoints.

After the signing certificate has been validated, SCIP attempts to establish a tunnel using the information in the *crypto.xml* file. If the Protection Profile mode is enabled in the *protectionprofile.xml* file, the endpoint uses the protection profiles specified in that file, rather than the pre-existing profiles defined in *crypto.xml*. Only the profiles specified in the *protectionprofile.xml* file are used during SCIP processing, and they are used only in the priority order specified in the *protectionprofile.xml* file.

Endpoints are able to communicate using the first matching protection profile that they share. (Endpoints might be running Stealth endpoint software created with identical *protectionprofile.xml* profile priority lists, or they might have been created using a different profile priority list.) The profile that is used during Stealth tunnel establishment depends on the profile priority list specified on the endpoint receiving the Stealth tunnel request.

2.3.1 TOE Physical Boundaries

The physical boundary of the TOE comprises an endpoint installation package that is created by the Enterprise Manager software running on the Management Server. The installation package includes the Stealth endpoint software and configuration information that specifies the IKE and IPsec cryptographic profiles the TOE will use when negotiating an IPsec tunnel with another endpoint. The Stealth endpoint software comprises:

- Kernel-mode drivers (`mlstpgw.sys`, `stealthii.sys`)
- The following services:
  - Unisys Stealth Logon Service (`USSL_Logon`)
  - Unisys PreLogon Service (`USSL_PreLogon`)
  - Unisys Protocol Service (`USSL_Protocol`)
- The Stealth Dashboard, a Windows task-bar program that provides a user interface for obtaining status information on the current Stealth configuration and enabling and disabling the Stealth service (if permitted by the TOE configuration).

[1] These are termed "protection profiles" in the TOE guidance documentation.

The TOE is supported on the following platforms that have all completed Common Criteria evaluations under the US Common Criteria Evaluation and Validation Scheme (CCEVS)—the relevant Validation Identifiers (VIDs) are provided:

- Windows 8 (VID #10520)
- Windows 8.1 (VID #10592)
- Windows Server 2012 R2 (VID #10529).
- Windows 10 (REF: 2016-36-INF-1779 v1)
- Windows Server 2016 (REF: 2016-36-INF-1779 v1)

All Windows endpoints on which the TOE is to be installed must have at least 2 GB of memory and must run the following software:

- .NET Framework version 3.5 with Service Pack 1 or .NET Framework version 4.x
- Java 7.x or later (Java Runtime Platform 1.7 or later).

Additionally, each endpoint on which the TOE is installed must be configured for FIPS mode.

The TOE requires the following in its operational environment:

- A Management Server, which runs the services comprising the Enterprise Manager component
- The Management Server and all endpoints must be part of an Active Directory (AD) domain.

In addition, the Management Server must be Stealth-enabled (i.e., the TOE is required to be installed on the Management Server as well as the VPN endpoints). However, unlike the endpoints on which the TOE is installed, the Management Server must not be configured for FIPS mode.

2.3.1.1 Tested Configurations

During evaluation testing, the TOE was tested on the following specific platforms:

- Windows 8 Pro 64-bit (Version 6.2.9200)
- Windows 8.1 Pro 32-bit (Version 6.3.9600)
- Windows Server Standard 2012 R2 64-bit (Version 6.3.9600).
- Windows 10 Pro 64-bit (Version 10.0.14393)
- Windows Server Standard 2016 64-bit (Version 10.0.14393)

For the purposes of testing, these platforms were installed on a Dell PowerEdge 1950 with:

- Intel Xeon E5430 (2.66 GHz, 64-bit, 12 MB L2 Cache)
- 2GB DDR2 SDRAM
- VMware ESXi 6.0.0.

2.3.2 TOE Logical Boundaries

This section summarizes the security functions provided by the TOE.

2.3.2.1 Cryptographic Support

The TOE enables an end user to establish a point-to-point VPN tunnel with another Stealth-enabled endpoint, using the underlying platform’s implementation of IKE and IPsec.

2.3.2.2 User Data Protection

The TOE ensures that residual information is protected from potential reuse in accessible objects such as network packets.

2.3.2.3 Identification and Authentication

The TOE provides the ability to use, store, and protect X.509v3 certificates. The TOE supports the use of X.509v3 certificates for IKE peer authentication and integrity verification. In addition, the TOE platform uses X.509v3 certificates.

2.3.2.4 Security Management

The TOE provides capabilities necessary to manage most of its security functionality. The TOE platform implements the security management functions not provided by the TOE.

2.3.2.5 TSF Protection

The TOE relies upon its underlying platform to perform self-tests that cover the TOE as well as the functions necessary to securely update the TOE.

2.3.2.6 Trusted Channel/Path

The TOE acts as a VPN client using IPsec to establish point-to-point secure channels with corresponding VPN clients.

2.4 TOE Documentation

This section identifies the guidance documentation included in the TOE:

- *Unisys Stealth Solution Common Criteria Evaluation Guidance Document*, Release 3.3, October 2017 (8205 5823–002)
- *Unisys Stealth Solution Information Center*, Release 3.3, September 2017 (8222 4189-009)
- *Unisys Stealth Solution Simple Implementation Guide*, Release 3.3, September 2017 (8231 0822-008).

3. Security Problem Definition

This ST includes by reference the Security Problem Definition (comprising threat statements and assumptions) from the *Protection Profile for IPsec Virtual Private Network Clients*, Version 1.4, 21 October 2013. The PP offers additional information about the identified threats, but that has not been reproduced here and the PP should be consulted if there is interest in that material.

In general, the PP has presented a Security Problem Definition appropriate for IPsec VPN clients, and as such is applicable to the Unisys Stealth VPN Client.

4. Security Objectives

This ST includes by reference the Security Objectives for the TOE specified in the *Protection Profile for IPsec Virtual Private Network Clients*, Version 1.4, 21 October 2013. The security objectives for the operational environment are reproduced below, since these objectives characterize technical and procedural measures each consumer must implement in their operational environment.

4.1 Security Objectives for the Operational Environment

OE.NO_TOE_BYPASS	Information cannot flow onto the network to which the VPN client's host is connected without passing through the TOE.
OE.PHYSICAL	Physical security, commensurate with the value of the TOE and the data it contains, is assumed to be provided by the operational environment.
OE.TRUSTED_CONFIG	Personnel configuring the TOE and its operational environment will follow the applicable security configuration guidance.

5. IT Security Requirements

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

All the SFRs have been drawn from *Protection Profile for IPsec Virtual Private Network (VPN) Clients*, Version 1.4, 21 October 2013 ([VPNPP]). As such, operations already performed in that PP are not identified here. Instead, the requirements have been copied from the PP and any incomplete selections or assignments have been performed herein. Of particular note, the PP makes a number of refinements and completes some SFR operations defined in the CC, so it should be consulted if necessary to identify those changes.

The SARs are the set of SARs specified in [VPNPP].

5.1 Extended Requirements

All of the extended requirements in this ST have been drawn from the [VPNPP]. The [VPNPP] defines the following extended SFRs and since they are not redefined in this ST, the [VPNPP] should be consulted for more information in regard to those CC extensions.

- FCS_CKM_EXT.2: Cryptographic key storage
- FCS_CKM_EXT.4: Cryptographic key zeroization
- FCS_IPSEC_EXT.1: Extended: Internet Protocol Security (IPsec) communications
- FCS_RBG_EXT.1: Extended: Cryptographic operation (random bit generation)
- FIA_X509_EXT.1: Extended: X.509 certificate validation
- FIA_X509_EXT.2: Extended: X509 use and management
- FPT_TST_EXT.1: Extended: TSF self test
- FPT_TUD_EXT.1: Extended: Trusted update

5.2 Security Functional Requirements

This section specifies the SFRs for the TOE.

Requirement Class	Requirement Component
FCS: Cryptographic Support	FCS_CKM.1(1): Cryptographic key generation (asymmetric keys)
	FCS_CKM.1(2): Cryptographic key generation (asymmetric keys - IKE)
	FCS_CKM_EXT.2: Cryptographic key storage
	FCS_CKM_EXT.4: Cryptographic key zeroization
	FCS_COP.1(1): Cryptographic operation (data encryption/decryption)
	FCS_COP.1(2): Cryptographic operation (for cryptographic signature)
	FCS_COP.1(3): Cryptographic operation (cryptographic hashing)
	FCS_COP.1(4): Cryptographic operation (keyed-hash message authentication)
	FCS_IPSEC_EXT.1: Extended: Internet Protocol Security (IPsec) Communications

Requirement Class	Requirement Component
	FCS_RBG_EXT.1: Extended: Cryptographic operation (random bit generation)
FDP: User Data Protection	FDP_RIP.2(1): Full residual information protection (provided by the TOE)
	FDP_RIP.2(2): Full residual information protection (provided by the TOE platform)
FIA: Identification and Authentication	FIA_X509_EXT.1: Extended: X.509 certificate validation
	FIA_X509_EXT.2: Extended: X.509 use and management
FMT: Security Management	FMT_SMF.1(1): Specification of management functions (required of TOE)
	FMT_SMF.1(2): Specification of management functions (provided by TOE)
	FMT_SMF.1(3): Specification of management functions (provided by platform)
FPT: Protection of the TSF	FPT_TST_EXT.1: Extended: TSF self test
	FPT_TUD_EXT.1: Extended: Trusted update
FTP: Trusted path/channels	FTP_ITC.1: Inter-TSF trusted channel

Table 1: TOE Security Functional Components

5.2.1 Cryptographic Support (FCS)

FCS_CKM.1(1) – Cryptographic key generation (asymmetric keys)

FCS_CKM.1.1(1) Refinement: The [***TOE platform***] shall generate asymmetric cryptographic keys used for key establishment in accordance with

- NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography” for finite field-based key establishment schemes;
- NIST Special Publication 800-56A, “Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography” for elliptic curve-based key establishment schemes and implementing “NIST curves” P-256, P-384 and [***no other curves***] (as defined in FIPS PUB 186-4, “Digital Signature Standard”)
- [***no other***]

and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits. See NIST Special Publication 800-57, “Recommendation for Key Management” for information about equivalent key strengths.

FCS_CKM.1(2) – Cryptographic key generation (for asymmetric keys – IKE)

FCS_CKM.1.1(2) Refinement: The [***TOE platform***] shall generate asymmetric cryptographic keys used for IKE peer authentication in accordance with a:
[

- ***FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4 for ECDSA schemes and implementing “NIST curves” P-256, P-384 and [no other curves]***]

and specified cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

FCS_CKM_EXT.2 – Cryptographic key storage

FCS_CKM_EXT.2.1 The [***TOE platform***] shall store persistent secrets and private keys when not in use in platform-provided key storage.

FCS_CKM_EXT.4 – Cryptographic key zeroization

FCS_CKM_EXT.4.1 Refinement: The [***TOE platform***] shall zeroize all plaintext secret and private cryptographic keys and CSPs when no longer required.

FCS_COP.1(1) – Cryptographic operation (data encryption/decryption)

FCS_COP.1.1(1) Refinement: The [***TOE platform***] shall perform encryption and decryption in accordance with a specified cryptographic algorithm AES operating in GCM and CBC mode with cryptographic key sizes 128-bits and 256-bits that meets the following:

- FIPS PUB 197, "Advanced Encryption Standard (AES)"
- NIST SP 800-38D, NIST SP 800-38A.

FCS_COP.1(2) – Cryptographic operation (for cryptographic signature)

FCS_COP.1.1(2) Refinement: The [***TOE platform***] shall perform cryptographic signature services in accordance with a specified cryptographic algorithm:
[

- ***FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.3 for RSA scheme***
- ***FIPS PUB 186-4, "Digital Signature Standard (DSS)", Appendix B.4 for ECDSA schemes and implementing "NIST curves" P-256, P-384 and [no other curve]***]

and cryptographic key sizes equivalent to, or greater than, a symmetric key strength of 112 bits.

Application Note: *The TOE platform uses RSA for integrity verification.*

FCS_COP.1(3) – Cryptographic operation (cryptographic hashing)

FCS_COP.1.1(3) Refinement: The [***TOE platform***] shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [***SHA-1, SHA-256, SHA-384***] and message digest sizes [***160, 256, 384***] bits that meet the following: FIPS Pub 180-4, "Secure Hash Standard."

Application Note: *The TOE platform uses SHA-1 for integrity verification.*

FCS_COP.1(4) – Cryptographic operation (keyed-hash message authentication)

FCS_COP.1.1(4) Refinement: The [***TOE platform***] shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-**[*SHA-256***] key size [**256**], and message digest size of [***256***] bits that meet the following: FIPS PUB 198-1, "The Keyed-Hash Message Authentication Code", and FIPS PUB 180-4, "Secure Hash Standard".

FCS_IPSEC_EXT.1[2] – Extended: Internet Protocol Security (IPsec) Communications

FCS_IPSEC_EXT.1.1 The [***TOE platform***] shall implement the IPsec architecture as specified in RFC 4301.

FCS_IPSEC_EXT.1.2 The [***TOE***] shall implement [***tunnel mode***].

FCS_IPSEC_EXT.1.3 The [***TOE***] shall have a nominal, final entry in the SPD that matches anything that is otherwise unmatched, and discards it.

[2] FCS_IPSEC_EXT.1 has been modified to comply with TD 0037.

FCS_IPSEC_EXT.1.4 The [***TOE, TOE platform***] shall implement the IPsec protocol ESP as defined by RFC 4303 using the cryptographic algorithms AES-GCM-128, AES-GCM-256 as specified in RFC 4106, [***no other algorithms***].

FCS_IPSEC_EXT.1.5 The [***TOE, TOE platform***] shall implement the protocol: [***IKEv1 as defined in RFCs 2407, 2408, 2409, RFC 4109, [no other RFCs for extended sequence numbers], and [no other RFCs for hash functions]***].

FCS_IPSEC_EXT.1.6 The [***TOE***] shall ensure the encrypted payload in the [***IKEv1***] protocol uses the cryptographic algorithms AES-CBC-128, AES-CBC-256 as specified in RFC 6379 and [***no other algorithm***].

FCS_IPSEC_EXT.1.7 The [***TOE***] shall ensure that IKEv1 Phase 1 exchanges use only main mode.

FCS_IPSEC_EXT.1.8 The [***TOE***] shall ensure that [***IKEv1 SA lifetimes can be configured by an [Administrator] based on [length of time, where the time values can be limited to: 24 hours for Phase 1 SAs and 8 hours for Phase 2 SAs]***].

FCS_IPSEC_EXT.1.9 The [***TOE platform***] shall generate the secret value x used in the IKE Diffie-Hellman key exchange ("x" in g^x mod p) using the random bit generator specified in FCS_RBG_EXT.1, and having a length of at least [**512**] bits.

FCS_IPSEC_EXT.1.10 The [***TOE platform***] shall generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life a specific IPsec SA is less than 1 in 2^[**256**] .

FCS_IPSEC_EXT.1.11 The [***TOE***] shall ensure that all IKE protocols implement DH Groups 14 (2048-bit MODP), 19 (256-bit Random ECP), and [***20 (384-bit Random ECP)***].

FCS_IPSEC_EXT.1.12 The [***TOE***] shall ensure that all IKE protocols perform peer authentication using a [***ECDSA***] that use X.509v3 certificates that conform to RFC 4945 and [***no other method***].

FCS_IPSEC_EXT.1.13 The [***TOE platform***] shall support peer identifiers of the following types: [***Distinguished Name (DN)***] and [***no other reference identifier type***].

FCS_IPSEC_EXT.1.14 The [***TOE platform***] shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

FCS_IPSEC_EXT.1.15 The [***TOE***] shall be able to ensure by default that the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 1***] connection is greater than or equal to the strength of the symmetric algorithm (in terms of the number of bits in the key) negotiated to protect the [***IKEv1 Phase 2***] connection.

FCS_RBG_EXT.1 – Extended: Cryptographic operation (random bit generation)

FCS_RBG_EXT.1.1 The [***TOE platform***] shall perform all deterministic random bit generation services in accordance with [***NIST Special Publication 800-90A using [CTR_DRBG (AES)]***].

FCS_RBG_EXT.1.2 The deterministic RBG shall be seeded by an entropy source that accumulates entropy from [***a platform-based RBG***] with a minimum of [***256 bits***] of entropy at least equal to the greatest security strength (according to NIST SP 800-57) of the keys and hashes that it will generate.

5.2.2 User Data Protection (FDP)

FDP_RIP.2 – Full residual information protection

FDP_RIP.2.1(1) The [***TOE***] shall enforce that any previous information content of a resource is made unavailable upon the [***deallocation of the resource from***] all objects.

FDP_RIP.2.1(2) The [***TOE platform***] shall enforce that any previous information content of a resource is made unavailable upon the [***allocation of the resource to***] all objects.

5.2.3 Identification and Authentication (FIA)

FIA_X509_EXT.1 – Extended: X.509 certificate validation

FIA_X509_EXT.1.1 The [***TOE platform***] shall validate certificates in accordance with the following rules:

- Perform RFC 5280 certificate validation and certificate path validation.
- Validate the revocation status of the certificate using [***a Certificate Revocation List (CRL) as specified in RFC 5759***].
- Validate the certificate path by ensuring the basicConstraints extension is present and the cA flag is set to TRUE for all CA certificates.
- Validate the extendedKeyUsage field according to the following rules:
    - Certificates used for [***trusted updates***, ***integrity verification***] shall have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3).

FIA_X509_EXT.1.2 The [***TOE platform***] shall only treat a certificate as a CA certificate if the following is met: the basicConstraints extension is present and the CA flag is set to TRUE.

Application Note: *The TOE uses certificates in support of trusted update and integrity verification while the TOE platform uses certificates in support of integrity verification.*

FIA_X509_EXT.2 – Extended: X.509 certificate use and management

FIA_X509_EXT.2.1 The TSF shall use X.509v3 certificates as defined by RFC 5280 to support authentication for IPsec exchanges, and [***digital signatures for FPT_TUD_EXT.1***, ***integrity checks for FPT_TST_EXT.1.2***].

FIA_X509_EXT.2.2 When a connection to determine the validity of a certificate cannot be established, the [***TOE platform***] shall [***accept the certificate***, ***not accept the certificate***].

FIA_X509_EXT.2.3 The [***TOE platform***] shall not establish an SA if a certificate or certificate path is deemed invalid.

Application Note: *The administrator can configure the TOE either to accept certificates when their validity cannot be established, or to reject certificates when their validity cannot be established. Once configured, there is no further administrator involvement.*

5.2.4 Security Management (FMT)

FMT_SMF.1(1) – Specification of Management Functions (required of TOE)

FMT_SMF.1.1(1) The TOE shall be capable of performing the following management functions:

- Specify VPN ~~gateways~~ **endpoints** to use for connections,
- Specify client credentials to be used for connections,
- **[*see FMT_SMF.1(2)*]**.

Application Note: *The TOE implements a client-to-client model of operation. The TOE establishes IPsec tunnels with other TOE-equipped clients rather than with a VPN gateway. Therefore, the security management function implemented by the TOE is to specify VPN clients to use for connections.*

FMT_SMF.1(2) – Specification of Management Functions (provided by TOE)

FMT_SMF.1.1(2) The [***TOE***] shall be capable of performing the following management functions:

- Configuration of IKE protocol version(s) used,
- Configure IKE authentication techniques used,
- Configure the cryptoperiod for the established session keys. The unit of measure for configuring the cryptoperiod shall be no greater than an hour,
- Configure certificate revocation check,
- **Configure the reference identifier for the peer,**[3]

[3] FMT_SMF.1(2) has been modified to comply with TD 0037.

- Specify the algorithm suites that may be proposed and accepted during the IPsec exchanges,
- ~~load X.509v3 certificates used by the security functions in this PP,~~
- ~~ability to update the TOE, and to verify the updates,~~
- ability to configure all security management functions identified in other sections of this PP,
- **[*action to be taken when a connection to determine the validity of a certificate cannot be established*]**.

FMT_SMF.1(3) – Specification of Management Functions (provided by platform)

FMT_SMF.1.1(3) The [***TOE platform***] shall be capable of performing the following management functions:

- ~~Configuration of IKE protocol version(s) used,~~
- ~~Configure IKE authentication techniques used,~~
- ~~Configure the cryptoperiod for the established session keys. The unit of measure for configuring the cryptoperiod shall be no greater than an hour,~~
- ~~Configure certificate revocation check,~~
- ~~Specify the algorithm suites that may be proposed and accepted during the IPsec exchanges,~~
- load X.509v3 certificates used by the security functions in this PP,
- ability to update the TOE, and to verify the updates,
- ~~ability to configure all security management functions identified in other sections of this PP,~~
- **[*configure peer identifier verification*]**.

5.2.5 Protection of the TOE Security Functions (FPT)

FPT_TST_EXT.1 – Extended: TSF self test

FPT_TST_EXT.1.1 The [***TOE platform***] shall run a suite of self tests during initial start-up (on power on) to demonstrate the correct operation of the TSF.

FPT_TST_EXT.1.2 The [***TOE platform***] shall provide the capability to verify the integrity of stored TSF executable code when it is loaded for execution through the use of the [**digital signature services specified in FCS_COP.1(2) (RSA) and cryptographic hashing services specified in FCS_COP.1(3)**].

FPT_TUD_EXT.1 – Extended: Trusted update

FPT_TUD_EXT.1.1 The [***TOE***] shall provide the ability to query the current version of the TOE firmware/software.

FPT_TUD_EXT.1.2 The [***TOE platform***] shall provide the ability to initiate updates to TOE firmware/software.

FPT_TUD_EXT.1.3 The [***TOE platform***] shall provide a means to verify firmware/software updates to the TOE using a digital signature mechanism and [***published hash***] prior to installing those updates.

5.2.6 Trusted Path/Channels (FTP)

FTP_ITC.1 – Inter-TSF trusted channel

FTP_ITC.1.1 Refinement: The [***TOE***] shall use IPsec to provide a trusted communication channel between itself and a VPN ~~Gateway~~ **endpoint** that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from disclosure and detection of modification of the channel data.

FTP_ITC.1.2 The [***TOE***] shall permit the TSF to initiate communication via the trusted channel.

FTP_ITC.1.3 The [***TOE***] shall initiate communication via the trusted channel for all traffic traversing that connection.

Application Note: *The TOE implements a client-to-client model* of operation. The TOE establishes *IPsec tunnels with other TOE-equipped clients rather than with a VPN gateway.*

5.3 Security Assurance Requirements

This section specifies the SARs for the TOE. The SARs are included by reference from [VPNPP].

Requirement Class	Requirement Component
ADV: Development	ADV_FSP.1: Basic functional specification
AGD: Guidance documents	AGD_OPE.1: Operational user guidance
	AGD_PRE.1: Preparative procedures
ALC: Life-cycle support	ALC_CMC.1: Labelling of the TOE
	ALC_CMS.1: TOE CM coverage
ATE: Tests	ATE_IND.1: Independent testing – conformance
AVA: Vulnerability assessment	AVA_VAN.1: Vulnerability survey

Table 2: Assurance Components

6. TOE Summary Specification

This chapter describes the following security functions implemented by the TOE or its underlying platform to satisfy the SFRs claimed in Section 5.2 of this ST:

- Cryptographic Support
- User Data Protection
- Identification and Authentication
- Security Management
- TSF Protection
- Trusted Channel/Path.

The following platforms, as described by the identified Security Target documents, are supported in the evaluated configuration (the VIDnnnnn identifies the NIAP evaluation reference for each platform):

- Windows 8—*Microsoft Windows 8 and Windows Server 2012 Security Target*, Version 1.0; December 19, 2014 (VID10520)
- Windows 8.1—*Microsoft Windows 8.1and Windows Phone Security Target*, Version 1.0, March 6, 2015 (VID10592)
- Windows Server 2012 R2—*IPsec VPN Client Security Target*, Version 1.0; January 23, 2014, in conjunction with *Assurance Continuity Maintenance Report for Microsoft Windows 8.1, Microsoft Windows RT 8.1, Microsoft Windows Server 2012 R2* (VID10529).
- Windows 10 and Windows Server 2016 - *Microsoft Windows Common Criteria Evaluation Microsoft Windows 10 (Anniversary Update) Microsoft Windows Server 2016*, Version 0.06, December 2, 2016 (REF: 2016-36-INF-1779 v1)

As described in Section 2.3.1, the TOE comprises an endpoint installation package that is created by the Enterprise Manager software in the TOE's operational environment. The installation package includes the Stealth endpoint software and configuration information that specifies the IKE and IPsec cryptographic profiles the TOE will use when negotiating an IPsec tunnel with another endpoint. The cryptographic profiles are termed "protection profiles" in the TOE's guidance documentation.

6.1 Cryptographic Support

6.1.1 IPsec Implementation

The TOE enables an end user to establish a point-to-point VPN tunnel with another Stealth-enabled endpoint, using the underlying platform's implementation of IKEv1 and IPsec.

Each of the supported Windows platforms provides an implementation of IPsec that conforms to RFC 4301. The platforms implement the IPsec Security Policy Database (SPD) via the Windows Filtering Platform (WFP). The WFP allows for filtering, monitoring and/or modification of TCP/IP packets, as well as filtering of IPsec traffic. The WFP allows for access to TCP/IP processing at different layers and can be used to filter on incoming or outgoing traffic.

The TOE creates WFP filters and directs the Windows Base Filter Engine to use IPsec for specific endpoint to endpoint traffic. The TOE uses the WFP to build conditions and filters that map to filter definitions provided to the TOE by the Authorization Service.

To configure filters, the Stealth administrator specifies the following:

- Filter Qualifier—a filter qualifier specifies a list of IP address exceptions and the port or protocol ranges allowed for a qualified filter.

- Filter List—a filter list is a list of qualified filters (IP addresses or ranges, with optional qualifiers), a list of filter lists, or a mix of qualified filters and filter lists.
- Filter Set—a filter set is a group of filter lists, with each list configured in a certain order and each set defined as either Allow or Deny. The order of filter lists in the filter set determines the order in which the filter lists are processed.

Filter sets work differently depending on whether they are applied to COIs or roles, as follows:

- A filter set applied to a COI is designated as a *Stealth Filter*. Stealth Filters control Stealth-enabled network traffic, allowing or denying information passed between Stealth endpoints that share COIs.
- A filter set applied to a role is designated as a *Clear Text Filter*. Clear Text Filters control clear text network traffic, allowing or denying information passed between Stealth endpoints and non-Stealth-enabled (clear text) components.

The Stealth administrator can apply either Allow filter sets or Deny filter sets to a COI or a role. Filter sets applied to roles are checked before filter sets applied to COIs.

- Allow Filter—an Allow filter set applied to a COI (Stealth Filter) operates as a set of IPsec PROTECT rules. The Stealth administrator specifies rules for filtering the network traffic to be protected. Messages that meet the filter criteria are transmitted using an IPsec tunnel. If the filter criteria are not met, then the data is discarded. An Allow filter set applied to a role (Clear Text Filter) operates as a set of IPsec BYPASS rules. The Stealth administrator specifies rules for filtering the network traffic to be allowed through without modification. Messages that meet the filter criteria are transmitted to or from the clear text network. If the filter criteria are not met, then the data is transmitted using an IPsec tunnel. No frames are discarded.
- Deny Filter—a Deny filter set applied to a COI (Stealth Filter) operates as a set of IPsec DISCARD rules. The Stealth administrator specifies rules to filter network traffic to be denied. Messages that meet the filter criteria are discarded. All other traffic is sent using an IPsec tunnel. A Deny filter set applied to a role (Clear Text Filter) operates as a set of IPsec PROTECT rules. The Stealth administrator specifies rules for filtering the network traffic to be denied passing through the network without modification. Messages that meet the filter criteria (that match the deny filter) are transmitted using an IPsec tunnel. All others are transmitted to or from the clear text network.

An Allow filter set has a default behavior of deny; that is, any condition that does not match the allow filter is denied by default. In the same way, a Deny filter set has a default behavior of allow, so that any condition that does not match the deny filter is allowed.

All IPsec connections established by the TOE operate in tunnel mode.

6.1.2 Supported Profiles

The TOE can be configured to support each of the IKEv1/IPsec cryptographic protection profiles specified in Table 3 below. Note that all profiles specify the use of ECDSA-signed X.509v3 certificates for IKE peer authentication. The TOE supports the use of NIST curves P-256 and P-384with ECDSA X.509v3 certificates. All IKEv1 Phase 1 exchanges use main mode only—aggressive mode is not used in these exchanges.

	IPsec	**IKEv1 Algorithms**		
Profile ID	**ESP Algorithm**	**Encryption**	**Integrity**	**DH Group**
0x08	AES-GCM-128	AES-CBC-128	SHA-256	14
0x10	AES-GCM-128	AES-CBC-128	SHA-256	19
0x20	AES-GCM-256	AES-CBC-256	SHA-384	14
0x40	AES-GCM-256	AES-CBC-256	SHA-384	19
0x80	AES-GCM-256	AES-CBC-256	SHA-384	20

Table 3: Supported Protection Profiles

The TOE configuration includes one or more of the supported protection profiles, arranged in priority order. The highest priority profile shared by two endpoints will be used to establish the IPsec tunnel between them and will determine which algorithms are used for IKE and ESP.

As indicated in Table 3, each supported protection profile ensures the strengths, in terms of the number of bits in the symmetric key, of the algorithms allowed for the IKE exchange are always equal to the strengths of the encryption algorithms allowed for ESP.

The TOE platform is responsible for generating nonces and the secret value *x* used in the IKE Diffie-Hellman key exchange (i.e., “*x*” in g^x mod *p*). When a random number is needed for either a nonce or for key agreement, the platform uses its NIST-approved random bit generator. When requested, the platform’s random bit generator can generate 256 or 512 bits for the caller. The TOE platform can generate nonces used in IKE exchanges in a manner such that the probability that a specific nonce value will be repeated during the life of a specific IPsec SA is less than 1 in 2^{256}. A 256-bit random value provides sufficient strength for nonces for all of the Diffie-Hellman groups supported by the TOE, while a 512-bit random value for *x* in g^x mod *p* is at least twice the “bits of security” value associated with each supported Diffie-Hellman group.

When the TOE platform processes X.509 certificates for IPsec, it follows RFC 5280 (Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile), which implies that if the platform deems that a certificate is invalid, such as for a DN mismatch, a revoked certificate, or an expired certificate, it will not establish an IPsec association.

The underlying Windows platform supports use of the Distinguished Name (DN) as a peer identifier. The Stealth administrator configures the reference identifier in the *protectionprofile.xml* file, although the only part of the DN available to set as a reference identifier is the Common Name. The underlying Windows platform compares the Common Name in the presented certificate to the configured reference identifier and fails the connection if they do not match.

6.1.3 Cryptographic Mechanisms

The TOE relies on the underlying Windows platform for the basic cryptographic functions used to support its operations.

Each Windows platform supported in the evaluated configuration implements the following cryptographic modules:

- Boot Manager—the system boot manager, called by the bootstrapping code that resides in the boot sector
- Winload OS Loader—the binary executable for loading the Windows operating system
- Code Integrity—a dynamically-linked library used to verify the integrity of other binary executable code files
- Kernel Mode Cryptographic Primitives Library—a kernel mode export driver that provides cryptographic services, through its documented interfaces, to Windows kernel components
- Cryptographic Primitives Library—provides cryptographic services, through its documented interfaces, to components and applications running on the Windows platform.

The Boot Manager, Winload OS Loader and Code Integrity modules provide functionality in support of code integrity verification (FPT_TST_EXT.1.2 – see Section 6.5.2). The Kernel Mode Cryptographic Primitives Library and Cryptographic Primitives Library provide cryptographic support to software running in kernel mode and in user mode respectively. Software requiring the cryptographic services specified by FCS_COP.1(*) requirements make calls into the appropriate cryptographic primitives library.

The implementation of all algorithms required by the TOE and the TOE platform to support the functional requirements specified in this ST has been NIST-validated on all supported platforms. This is summarized in the following table.

	Windows 8	Windows 8.1	Server 2012 R2	Windows 10	Server 2016
RSA 2048 bit	#1134	#1493, #1494	#1493, #1494	#2192	#2192
ECDSA P-256, P-384	#341	#505	#505	#911, #920	#911, #920
DSA 2048, 3072	#687	#855	#855	#1098	#1098
Key Agreement	#36	#47	#47	#92	#92
AES CBC 128, 256	#2197	#2832	#2832	#4063, #4074	#4063, #4074
AES GCM 128 256	#2216	#2832	#2832	#4064	#4064
SHA 1, 256, 384	#1903	#2396	#2396	#3346, #3347	#3346, #3347
HMAC-SHA-256	#1345	#1773	#1773	#2651, #2661	#2651, #2661
CTR_DRBG (AES)	#258	#489	#489	#1217, #1222	#1217, #1222

Table 4: Windows Cryptographic Algorithm Validation Program Certifications

The cryptographic hash functions (SHA-1, SHA-256, SHA-384) are associated with the following cryptographic functions:

- Keyed-hash message authentication code (HMAC) operations
- RSA and ECDSA digital signature verification services
- Diffie-Hellman and Elliptic Curve Diffie-Hellman key agreement.

The keyed-hash message authentication function (HMAC-SHA-256) is used as a pseudo-random function in IKEv1 for generating key material and for authentication of the IKEv1 Security Association. It is also used as a data origin authentication and integrity verification algorithm in ESP.

The TOE relies on the underlying platform for generation of asymmetric keys used for key establishment purposes and for IKE peer authentication. The Kernel Mode Cryptographic Primitives Library (cng.sys) cryptomodule included in each of the underlying platforms is invoked for this purpose.

Table 5 lists the keys and critical security parameters (CSPs) managed by the underlying Windows platform that are needed to meet the requirements in this ST.

Security Relevant Data Item	Description	Persistent?
Symmetric encryption/decryption keys	Keys used for AES encryption/decryption for IKEv1 and IPsec ESP.	N
HMAC keys	Keys used for HMAC-SHA-256.	N
Asymmetric ECDSA Public Keys	Keys used for the verification of ECDSA digital signatures for IPsec traffic and peer authentication.	Y
Asymmetric ECDSA Private Keys	Keys used for the calculation of ECDSA digital signatures for IPsec traffic and peer authentication.	Y
Asymmetric RSA Public Keys	Keys used for the verification of RSA digital signatures for signed product updates and code integrity verification.	Y
AES-CTR DRBG Seed	A secret value maintained internal to the module that provides the seed material for AES-CTR DRBG output.	N
AES-CTR DRBG Entropy	A secret value maintained internal to the module that provides the	N

Security Relevant Data Item	**Description**	**Persistent?**
Input	entropy material for AES-CTR DRBG output.	
AES-CTR DRBG V	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output.	N
AES-CTR DRBG Key	A secret value maintained internal to the module that provides the entropy material for AES-CTR DRBG output.	N
DH Private and Public values	Private and public values used for Diffie-Hellman key establishment.	N
ECDH Private and Public values	Private and public values used for EC Diffie-Hellman key establishment.	N

Table 5: Keys and CSPs

Each Windows platform includes a key isolation service designed specifically to host secret and private keys in a protected process to mitigate tampering or access to sensitive key materials. As such, they are stored in plaintext in the underlying Windows platform. The Windows platforms perform a key error detection check on each transfer of key (internal, intermediate transfers). The Windows platforms prevent archiving of expired (private) signature keys. The Windows platforms destroy non-persistent cryptographic keys after a cryptographic administrator-defined period of time of inactivity. The Windows platforms overwrite each intermediate storage area for plaintext key/critical cryptographic security parameter (i.e., any storage, such as memory buffers, that is included in the path of such data). This overwriting is performed as follows:

- For non-volatile memories other than EEPROM and Flash, the overwrite is executed three or more times using a different alternating data pattern each time upon the transfer of the key/critical cryptographic security parameter to another location.
- For volatile memory and non-volatile EEPROM and Flash memories, the overwrite is a single direct overwrite consisting of a pseudo random pattern, followed by a read-verify upon the transfer of the key/critical cryptographic security parameter to another location.

Each Windows platform implements a deterministic random bit generation (DRBG) function in accordance with NIST Special Publication 800-90. Windows generates random bits by taking the output of a cascade of two SP800-90 AES-256 counter mode based DRBGs in kernel-mode and four cascaded SP800-90 AES-256 DRBGs in user-mode. Programmatic callers can choose to obtain either 128 or 256 bits from the DRBG, which is seeded from the Windows entropy pool. The entropy pool is populated using the following values:

- An initial entropy value from a seed file provided to the Windows OS Loader at boot time (512 bits of entropy).
- A calculated value based on the high-resolution CPU cycle counter which fires after every 1024 interrupts (a continuous source providing 16384 bits of entropy).
- Random values gathered periodically from the Trusted Platform Module (TPM), if one is available on the system (320 bits of entropy on boot, 384 bits thereafter).
- Random values gathered periodically by calling the RDRAND CPU instruction, if supported by the CPU (256 bits of entropy).

The interface for the Windows random number generator is BCryptGenRandom.

The Cryptographic Support function satisfies the following security functional requirements:

- FCS_CKM.1(1)—the TOE platform generates asymmetric cryptographic keys used for key establishment in accordance with NIST Special Publication 800-56A.
- FCS_CKM.1(2)—the TOE platform generates asymmetric cryptographic keys used for IKE peer authentication in accordance with FIPS PUB 186-4 for ECDSA schemes and implements NIST curves P-256 and P-384.

- FCS_CKM_EXT.2—the TOE platform stores persistent secrets and private keys in platform-provided key storage.
- FCS_CKM_EXT.4—the TOE platform zeroizes cryptographic keys when they are no longer needed.
- FCS_COP.1(1)—the TOE platform implements AES using CBC and GCM modes to support the IPsec protocol.
- FCS_COP.1(2)—the TOE platform provides digital signature generation and verification services using RSA with 2048-bit modulus and ECDSA with NIST curves P-256 and P-384.
- FCS_COP.1(3)—the TOE platform implements SHA-1, SHA-256 and SHA-384 cryptographic hash algorithms to support digital signature verification and IPsec operations.
- FCS_COP.1(4)—the TOE platform implements HMAC-SHA-256 for use with IPsec.
- FCS_IPSEC_EXT.1—the TOE platform implements IKE and IPsec while the TOE contributes to satisfying elements of FCS_IPSEC_EXT.1 by ensuring the TOE platform is configured appropriately (e.g., by ensuring only the specified algorithms will be used in Stealth tunnels).
- FCS_RBG_EXT.1—the TOE platform implements a CTR_DRBG(AES) random bit generator to support generation of symmetric keys and asymmetric key pairs.

6.2 User Data Protection

Both the TOE and the underlying Windows platform are designed to ensure that no residual information exists in network packets. After processing either an incoming or outgoing network packet, the TOE overwrites with zeros the contents of each memory buffer it used in packet processing, prior to releasing the buffer to the memory pool of the underlying platform. The Windows platform ensures that previous information contents of resources used for new objects are not discernible in the new object via zeroing or overwriting of memory and tracking read/write pointers for disk storage. Every process is allocated new memory and an execution context. Memory is zeroed or overwritten before allocation.

The User Data Protection function satisfies the following security functional requirement:

- FDP_RIP.2(*)—both the TOE and the underlying Windows platform are designed to ensure that no residual information exists in network packets.

6.3 Identification and Authentication

The TOE ensures the use of X.509v3 certificates, as defined in RFC 5280, for authentication of IPsec exchanges and to support integrity verification. For certificates used to support IPsec authentication, the certificate to use is identified by its Common Name, which is specified in the protectionprofile.xml file used when configuring the Stealth endpoint installation package. The certificate must be stored in the Local Computer Personal certificate store on the Windows platform. The underlying Windows platform additionally uses X.509v3 certificates for trusted updates and verifying the integrity of TSF executable code.

The Windows platform processes X.509v3 certificates for IPsec in accordance with RFC 5280 and uses Certificate Revocation Lists (CRLs), as specified in RFC 5759, to determine revocation status. The Windows platform validates all applicable usage constraints, including:

- Ensuring the basicConstraints extension is present and the cA flag is set to TRUE for all CA certificates
- Validating the extendedKeyUsage field to ensure certificates used for integrity verification and trusted updates have the Code Signing purpose (id-kp 3 with OID 1.3.6.1.5.5.7.3.3).

The Windows platform validates the full path of X.509v3 certificates by validating it can construct a certificate path from the certificate through any intermediary CAs to the trusted root CA certificate. If the Windows platform can successfully build the certificate path, then it will next check the validity of the certificates in the path. Assuming the certificates are valid, the Windows platform finally checks the revocation status of all certificates.

If the Windows platform deems that a certificate is invalid, such as for a distinguished name (DN) mismatch, a revoked certificate, or an expired certificate, it will not establish an IPsec association. The Stealth administrator can configure a value in the protectionprofile.xml file that controls CRL checking. If strong CRL checking is enabled and the Windows platform cannot establish a connection to determine the validity of a certificate, the Windows platform will not accept the certificate. If weak CRL checking is enabled and the Windows platform cannot establish a connection to determine the validity of a certificate, the Windows platform will accept the certificate. The Windows platform administrator can also configure the platform to verify the specific DN in the remote machine's IPsec certificate and fail the connection if the DN is incorrect.

The TOE uses X.509v3 certificates to support integrity verification of the endpoint installation package. During the endpoint package generation process, the Enterprise Manager reads the protectionprofile.xml file, validates it, and inserts the contents into the crypto.xml file. Enterprise Manager signs the resulting crypto.xml file using a signing certificate and includes it in the endpoint package. The resulting endpoint package is then installed on the endpoints.

The endpoints on which the package is installed read the crypto.xml file and validate the signature of the signing certificate using the associated trusted root certificate, which is also installed on these endpoints.

The Windows platform uses X.509v3 certificates to support integrity verification of TSF kernel-mode executables when they are loaded for execution, while the TOE uses X.509v3 certificates to support integrity verification of TSF user-mode services. This is described fully in Section 6.5.2 below.

The X.509v3 certificate used for IKE peer authentication (termed the Protection Profile Authentication certificate in the TOE guidance documentation) is imported to the Local Computer Personal certificate store on the TOE platform. The physical location of the certificate store is the registry hive for the computer. Certificates can only be loaded into the certificate store by an authorized administrator who has write (i.e., modify) and delete access rights to the registry keys that serve as the certificate store. Certificates can be loaded using GUI administrator tools, command line tools, or through local and group policy.

The Identification and Authentication function satisfies the following security functional requirement:

- FIA_X509_EXT.1—the underlying Windows platform performs all needed validation of X.509v3 certificates.
- FIA_X509_EXT.2—the TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for IPsec exchanges, code integrity verification, and trusted updates. The underlying Windows platform shall not establish an SA if a certificate or certificate path is deemed invalid, but can be configured to accept or not accept a certificate in the event the Windows platform is unable to establish a connection to determine certificate validity.

6.4 Security Management

The TOE provides the following security management functions:

- Specification of VPN endpoints to use for connections
- Specification of the client credentials (i.e., X.509v3 certificate) to be used for IPsec connections
- Configuration of the IKE protocol version to be used by the platform when establishing the IPsec VPN tunnel
- Configuration of the IKE authentication techniques used by the platform
- Configuration of the cryptoperiod for the established session keys. The cryptoperiod is specified in terms of hours
- Configuration of the certificate revocation check
- Configuration of the reference identifier for the peer
- Specification of the algorithm suites that can be proposed and accepted during the IPsec exchanges
- Configuration of the action to be taken when a connection to determine the validity of a certificate cannot be established.

The underlying platform provides the following security management functions:

- Loading X.509v3 certificates used for IKE peer authentication and trusted updates.
- Update of the TOE and verification of updates
- Configure the capability to verify peer identifiers.

The Security Management function satisfies the following security functional requirements:

- FMT_SMF.1(1)—the TOE is capable of performing the required security management functions of specifying VPN endpoints and client credentials.
- FMT_SMF.1(2)—the TOE is capable of performing additional security management functions described in this ST, including configuring the IKE and IPsec protocols.
- FMT_SMF.1(3)—the underlying Windows platform is capable of performing security management functions to load X.509v3 certificates, update the TOE, and verify peer identifiers.

6.5 TSF Protection

6.5.1 Self-Tests

The underlying Windows platform performs a set of self-tests to verify that Windows is operating correctly.

The relevant kernel-mode startup self-tests are:

- AES encrypt/decrypt Known Answer Test for CBC and GCM modes
- RSA Known Answer Test
- ECDSA sign/verify tests on supported NIST curves
- ECDH secret agreement Known Answer Test on supported NIST curves
- HMAC-SHA-256 Known Answer Tests
- SP800-56A concatenation KDF Known Answer Tests (same as Diffie-Hellman KAT)
- SP800-90 AES-256 counter mode DRBG Known Answer Tests (instantiate, generate and reseed).

The Windows kernel-mode cryptographic module (the Kernel Mode Cryptographic Primitives Library) also performs pair-wise consistency checks upon each invocation of RSA, ECDH, and ECDSA key-pair generation and import. SP 800-56A conditional self-tests are also performed. A continuous RNG test (CRNGT) is used for the random number generators of this cryptographic module. A pair-wise consistency test is done for Diffie-Hellman.

The Kernel Mode Cryptographic Primitives Library is loaded into the kernel's memory early during the boot process. If there is a failure in any startup self-test, the Kernel Mode Cryptographic Primitives Library DriverEntry function will fail to return the STATUS_SUCCESS status to its caller. The only way to recover from the failure of a startup self-test is to attempt to invoke DriverEntry again, which will rerun the self-tests, and will only succeed if the self-tests pass.

By thoroughly exercising the cryptography that is the basis for IPsec, Windows will avoid situations where data is not transmitted through a non-trusted channel in cases where the authorized administrator has deemed that a trusted channel is necessary.

6.5.2 Code Integrity Checking

The underlying Windows platform verifies the integrity of Windows program code using its Code Integrity capability. Kernel-mode code signing (KMCS) prevents kernel-mode device drivers from loading unless they are digitally signed by trusted developers. KMCS, using public-key cryptography technologies, requires that kernel-mode code include a digital signature generated by one of the trusted certificate authorities. The TOE includes two kernel-mode device drivers (`mlstpgw.sys` and `stealthii.sys`) that must be digitally signed in this fashion. When either of these kernel device drivers tries to load, Windows decrypts the hash included with the driver using the public key stored in the certificate, then verifies that the hash matches the one computed with the code. The authenticity of the certificate is checked in the same way, using the certificate authority's public key, which is trusted by Windows. The root public key of the certificate chain used to verify the signature must match one of Microsoft's root public keys. These Microsoft root public keys are hardcoded in the Windows Boot Manager.

When each of the TOE's user-mode services is started, the integrity of the service is verified by invoking services of the Windows platform and using the digital signature of the executable. If the digital signature of a service cannot be verified, the following occurs:

- The service enters the paused state
- An error is written to the Windows System Event log.

In addition, the TOE cannot be enabled, and the Stealth Dashboard Stealth Shield icon appears yellow in the Windows taskbar.

In contrast, if the integrity verification is successful, the services start. The Stealth Dashboard Stealth Shield icon appears blue if the TOE is installed as Always On, or red if the TOE is installed as On Demand (when the TOE is enabled, the icon then turns blue).

6.5.3 Trusted Update

The TOE endpoint installation package includes the Stealth Dashboard. The Stealth Dashboard is accessible from the Windows task bar and can be used to obtain status information on the current Stealth configuration, including the software version number.

The TOE software is updated by generating a new endpoint installation package on the Enterprise Manager and installing the new installation packages on the Windows endpoint. A signing certificate is used to protect the integrity of the endpoint software package. The XML files in the endpoint software package are signed, and that signature is verified by the signing certificate. Prior to installation of the endpoint package, the signing certificate's trust chain is installed on the endpoint, including the trusted root certificate (stored in the Windows Local Computer/Trusted Root Certification Authorities store) and any intermediate certificates (stored in the Windows Local Computer/Intermediate Certification Authorities store). The signing certificate itself is automatically included in the endpoint package.

The endpoint user is able to use the capabilities of the underlying Windows platform to verify the integrity of the endpoint installation package by verifying a SHA-256 hash calculated over the entire installation package and by verifying the validity of the digital signatures on the executables within the installation package. The steps to be performed by administrators and users are described in *Unisys Stealth Solution Common Criteria Evaluation Guidance Document*. The underlying Windows platform reports to the user if the digital signature is OK (for successful signature verification) or invalid (for unsuccessful signature verification).

The TSF Protection function satisfies the following security functional requirements:

- FPT_TST_EXT.1—the underlying Windows platform performs various cryptographic self-tests during startup to demonstrate correct operation of the mechanisms relied on by the TOE. The Windows platform provides the capability to verify the integrity of stored executable code when it is loaded for execution.
- FPT_TUD_EXT.1—the underlying Windows platform provides capabilities to initiate updates to the TOE and to verify the integrity of TOE updates prior to installation. The TOE provides a means for the user to query its current version.

6.6 Trusted Channel/Path

See Section 6.1 for a description of how the TOE establishes IPsec VPN connections with configured VPN endpoints. The resulting VPNs ensure that both ends of the channel are authenticated and the channel protects data from disclosure and modification.

The Trusted Channel/Path function satisfies the following security functional requirement:

- FTP_ITC.1— TOE uses IPsec to provide a protected communication channel between itself and an IPsec VPN endpoint. The channel provides assured identification of its end points and protects transmitted data from disclosure and modification.

7. Protection Profile Claims

This ST is conformant to the *Protection Profile for IPsec Virtual Private Network (VPN) Clients*, Version 1.4, 21 October 2013.

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

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

The SFRs in this ST are reproduced from [VPNPP] and assignment and selection operations completed as appropriate, with the following variations:

- FCS_IPSEC_EXT.1 has been modified in accordance with NIAP Technical Decision 0037, which replaces FCS_IPSEC_EXT.1.13 with the following two elements:

FCS_IPSEC_EXT.1.13	The [selection: TOE, TOE platform] shall support peer identifiers of the following types: [selection: IP address, Fully Qualified Domain Name (FQDN), user FQDN, Distinguished Name (DN)] and [selection: no other reference identifier type, [assignment: other supported reference identifier types]].
FCS_IPSEC_EXT.1.14	The [selection: TOE, TOE platform] shall not establish an SA if the presented identifier does not match the configured reference identifier of the peer.

  Additionally, FCS_IPSEC_EXT.1.14 from [PPVPNC] is renumbered FCS_IPSEC_EXT.1.15 in this ST.

- FDP_RIP.1 is iterated to capture the contrasting behaviors of the TOE and the TOE platform with regards to residual information protection.

- FMT_SMF.1 is iterated to distinguish between the security management functions that are: required of the TOE; provided by the TOE; and provided by the TOE platform. Additionally, the following security management function as specified in TD0037 is added to FMT_SMF.1(2): “configure the reference identifier for the peer”.

- FMT_SMF.1(1) and FTP_ITC.1 are refined to specify “endpoint” instead of “gateway”, since the TOE implements a client-to-client model of operation.

8. Rationale

This ST includes by reference Security Problem Definition, Security Objectives, and Security Assurance Requirements from *Protection Profile for IPsec Virtual Private Network (VPN) Clients*, Version 1.4, 21 October 2013. The ST makes no additions to the PP assumptions. The PP security functional requirements have been reproduced with the PP operations completed. Operations on the security requirements follow the PP application notes and assurance activities. Consequently, the PP rationale applies, but is incomplete. The TOE Summary Specification rationale below serves to complete the rationale required for the ST.

8.1 TOE Summary Specification Rationale

Section 6, the TOE Summary Specification, describes how the security functions of the TOE meet the claimed SFRs. The following table provides a mapping of the SFRs to the security function descriptions to support the TOE Summary Specification.

	Cryptographic Support	User Data Protection	Identification and Authentication	Security Management	TSF Protection	Trusted Channel/Path
FCS_CKM.1(1)	X					
FCS_CKM.1(2)	X					
FCS_CKM_EXT.2	X					
FCS_CKM_EXT.4	X					
FCS_COP.1(*)	X					
FCS_IPSEC_EXT.1	X					
FCS_RBG_EXT.1	X					
FDP_RIP.2(*)		X				
FIA_X509_EXT.1			X			
FIA_X509_EXT.2			X			
FMT_SMF.1(*)				X		
FPT_TST_EXT.1					X	
FPT_TUD_EXT.1					X	
FTP_ITC.1						X

Table 6: Security Functions vs. Requirements Mapping