FireEye xAgent Application Security Target

Acumen Security, LLC.

Document Version: 1.0

Table Of Contents

Revision History

Version	Date	Description
1.0	July 2016	Publication

1 Security Target Introduction

1.1 Security Target and TOE Reference

This section provides information needed to identify and control this ST and its TOE.

Category	Identifier
ST Title	FireEye Endpoint Agent Security Target
ST Version	1.0
ST Date	July 2016
ST Author	Acumen Security, LLC.
TOE Identifier	FireEye Endpoint Agent
TOE Software Version	21
TOE Developer	FireEye, Inc.
Key Words	Software

Table 1 TOE/ST Identification

1.2 TOE Overview

The TOE is a software agent that resides on a host platform. The software exclusively interacts with the NIAP validated FireEye HX Series Appliances (NIAP VID 10675). This interaction consists of the TOE receiving policies from an external HX series appliance (validated separately) and sending any alerts that are found as a result of these scans. This is done via polling. The TOE is an enterprise managed agent that runs in the background of an endpoint platform. It is intended that the user will have no interaction with the software and will not be alerted of communications with the external HX appliance.

The frequency at which the agent communicates with the HX appliance is set by the enterprise. By default, each agent polls the HX appliance every 600 seconds (10 minutes) to obtain information and task requests and polls the appliance every 30 minutes to obtain the latest indicators. When new policies are received, they are used to identify potential intrusions on the host platform.

1.3 TOE Architecture

1.3.1 Physical Boundaries

The TOE boundary is the application software which runs on the host platform. The software is pushed to the host platform from a FireEye HX series and installs natively as a kernel and user space application. The software runs on Microsoft Operating Systems. The following Operating Systems are included in this evaluation,

- Windows 7 (SP1) x64 running on an Intel Xeon processor
- Windows 7 (SP1) x32 running on an Intel Xeon processor
- Windows Server 2012R2 x64 running on an Intel Xeon processor
- Windows Server 2008R2 (SP1) x64 running on an Intel Xeon processor
- Windows 10 x64 running on an Intel Xeon processor
- Windows 10 x32 running on an Intel Xeon processor

1.3.2 Security Functions provided by the TOE

The TOE provides the security functionality required by [SWAPP].

1.3.2.1 Cryptographic Support

The TOE provides cryptographic support for the following features,

- TLS connectivity with the following entities:
  - HX Series Appliance (NIAP VID 10675)
- Digital certificate generation

The cryptographic services provided by the TOE are described below.

Cryptographic Method	Use within the TOE
RSA Signature Services	Used in TLS session establishment. Used in secure software update.
SP 800-90 DRBG	Used in TLS session establishment. Used in digital certificate generation.
SHS	Used in secure software update. Used in digital certificate generation.
HMAC-SHS	Used to provide TLS traffic integrity verification.
AES	Used to encrypt TLS traffic Secure certificate storage

Table 2 TOE Provided Cryptography

Each of these cryptographic algorithms have been validated for conformance to the requirements specified in their respective standards, as identified below. Each of these algorithms are implemented as part of the OpenSSL cryptographic library, version 1.0.1.

Algorithm	Standard	CAVP Certificate #	Processor
RSA	FIPS PUB 186-4 (Signature generation/verification)	Cert. #1976, 1977	Intel Xeon
SP 800-90 DRBG	SP 800-90	Cert. #1103, 1104	Intel Xeon
SHS	FIPS Pub 180-4	Cert. #3194, 3195	Intel Xeon
HMAC-SHS	FIPS Pub 198-1, FIPS Pub 180-4	Cert. #2517, 2518	Intel Xeon
AES	NIST SP 800-38A	Cert. #3873, 3874	Intel Xeon

Table 3 CAVP Algorithm Testing References

1.3.2.2 Secure Software Update

The TOE is distributed as a Microsoft .MSI file providing a consistent and reliable versioning. After initial installation, all updates to the xAgent are distributed as .MSI. Each TOE installation and update is signed by FireEye and can only come from the HX Series appliance associated with the TOE.

1.3.2.3 Protection of the TSF

The TOE employs several mechanisms to ensure that it is secure on the host platform. The TOE never allocates memory with both write and execute permission. The TOE is designed to operate in an environment in which the following security techniques are in effect, Data execution prevention,

Mandatory address space layout randomization (no memory map to an explicit address), Structured exception handler overwrite protection, Export address table access filtering, Anti-Return Oriented Programming, and SSL/TLS certificate trust pinning. This allows the TOE to operate in an environment in which the Enhanced Mitigation Experience Toolkit is also running. During compilation the TOE is built with several flags enabled that check for engineering flaws. The TOE is built with the /GS flag enabled. This reduces the possibilities of stack-based buffer overflows in the product.

1.3.2.4 Trusted Path/Channels

The TOE receives scanning policies from the associated HX Series appliance over the network which it uses on the host platform. This connection is always secured using TLS.

1.3.3 TOE Documentation

- [ST] FireEye xAgent Application Security Target, version 1.0
- [AGD] Common Criteria FireEye Endpoint Agent Addendum, Release 21

1.3.4 Other References

Protection Profile for Application Software, version 1.1, dated, 05 November 2014 [SWAPP].

2 Conformance Claims

2.1 CC Conformance

This TOE is conformant to:

- Common Criteria for Information Technology Security Evaluations Part 1, Version 3.1, Revision 4, September 2012
- Common Criteria for Information Technology Security Evaluations Part 2, Version 3.1, Revision 4, September 2012: Part 2 extended
- Common Criteria for Information Technology Security Evaluations Part 2, Version 3.1, Revision 4, September 2012: Part 3 extended

2.2 Protection Profile Conformance

This TOE is conformant to:

- Protection Profile for Application Software, version 1.1, dated, 05 November 2014 [SWAPP].

2.3 Conformance Rationale

This Security Target provides exact conformance to Version 1.1 of the Protection Profile for Application Software, version 1.1. The security problem definition, security objectives and security requirements in this Security Target are all taken from the Protection Profile performing only operations defined there.

2.3.1 Technical Decisions

The following Technical Decisions have been considered for this evaluation:

- TD0073: Additional Option to meet FPT_TUD_EXT.1.2 in App PP v1.1
- TD0072: FIA_X509_EXT.1.1 Certificate Depth in App PP v1.1
- TD0070: Assurance Activity Clarification for FCS_RGB_EXT.1 in Software Application PP
- TD0054: Clarification of FPT_API_EXT.1.1 Requirement in APP PP v1.1
- TD0051: Android Implementation of TLS in App PP v1.1
- TD0050: FMT_CFG_EXT.1.2 Change in APP SW PPv1.1
- TD0025: Update to FCS_COP.1(2)
- TD0024: Application Settings Clarification for FMT_MEC_EXT.1

3 Security Problem Definition

The security problem definition has been taken from [SWAPP] and is reproduced here for the convenience of the reader. The security problem is described in terms of the threats that the TOE is expected to address, assumptions about the operational environment, and any organizational security policies that the TOE is expected to enforce.

3.1 Threats

The following threats are drawn directly from the SWAPP.

ID	Threat
T.NETWORK_ATTACK	An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may engage in communications with the application software or alter communications between the application software and other endpoints in order to compromise it.
T.NETWORK_EAVESDROP	An attacker is positioned on a communications channel or elsewhere on the network infrastructure. Attackers may monitor and gain access to data exchanged between the application and other endpoints.
T.LOCAL_ATTACK	An attacker can act through unprivileged software on the same computing platform on which the application executes. Attackers may provide maliciously formatted input to the application in the form of files or other local communications.
T.PHYSICAL_ACCESS	An attacker may try to access sensitive data at rest.

Table 4 Threats

3.2 Assumptions

The following assumptions are drawn directly from the SWAPP.

ID	Assumption
A.PLATFORM	The TOE relies upon a trustworthy computing platform for its execution. This includes the underlying platform and whatever runtime environment it provides to the TOE.
A.PROPER_USER	The user of the application software is not willfully negligent or hostile, and uses the software in compliance with the applied enterprise security policy.
A.PROPER_ADMIN	The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

Table 5 OSPs

3.3 Organizational Security Policies

There are no OSPs for the application

4 Security Objectives

The security objectives have been taken from [SWAPP] and are reproduced here for the convenience of the reader.

4.1 Security Objectives for the TOE

The following security objectives for the TOE were drawn directly from the SWAPP.

ID	TOE Objective
O.INTEGRITY	Conformant TOEs ensure the integrity of their installation and update packages, and also leverage execution environment-based mitigations. Software is seldom if ever shipped without errors, and the ability to deploy patches and updates to fielded software with integrity is critical to enterprise network security. Processor manufacturers, compiler developers, execution environment vendors, and operating system vendors have developed execution environment-based mitigations that increase the cost to attackers by adding complexity to the task of compromising systems. Application software can often take advantage of these mechanisms by using APIs provided by the runtime environment or by enabling the mechanism through compiler or linker options. Addressed by: FDP_DEC_EXT.1, FMT_CFG_EXT.1, FPT_AEX_EXT.1, FPT_TUD_EXT.1
O.QUALITY	To ensure quality of implementation, conformant TOEs leverage services and APIs provided by the runtime environment rather than implementing their own versions of these services and APIs. This is especially important for cryptographic services and other complex operations such as file and media parsing. Leveraging this platform behavior relies upon using only documented and supported APIs. Addressed by: FMT_MEC_EXT.1, FPT_API_EXT.1, FPT_LIB_EXT.1
O.MANAGEMENT	To facilitate management by users and the enterprise, conformant TOEs provide consistent and supported interfaces for their security-relevant configuration and maintenance. This includes the deployment of applications and application updates through the use of platform-supported deployment mechanisms and formats, as well as providing mechanisms for configuration. Addressed by: FMT_SMF.1, FPT_IDV_EXT.1, FPT_TUD_EXT.1.5
O.PROTECTED_STORAGE	To address the issue of loss of confidentiality of user data in the event of loss of physical control of the storage medium, conformant TOEs will use data-at-rest protection. This involves encrypting data and keys stored by the TOE in order to prevent unauthorized access to this data. Addressed by: FDP_DAR_EXT.1, FCS_STO_EXT.1, FCS_RBG_EXT.1
O.PROTECTED_COMMS	To address both passive (eavesdropping) and active (packet modification) network attack threats, conformant TOEs will use a trusted channel for sensitive data. Sensitive data includes cryptographic keys, passwords, and any other data specific to the application that should not be exposed outside of the application. Addressed by: FTP_DIT_EXT.1, FCS_TLSC_EXT.1, FCS_DTLS_EXT.1, FCS_RBG_EXT.1

Table 6 Objectives for the TOE

4.2 Security Objectives for the Operational Environment

The following security objectives for the operational environment assist the TOE in correctly providing its security functionality. These track with the assumptions about the environment.

ID	Objective for the Operation Environment
OE.PLATFORM	The TOE relies upon a trustworthy computing platform for its execution. This includes the underlying operating system and any discrete execution environment provided to the TOE.
OE.PROPER_USER	The user of the application software is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.
OE.PROPER_ADMIN	The administrator of the application software is not careless, willfully negligent or hostile, and administers the software within compliance of the applied enterprise security policy.

Table 7 Objectives for the environment

5 Security Requirements

This section identifies the Security Functional Requirements for the TOE and/or Platform. The Security Functional Requirements included in this section are derived from Part 2 of the Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 4, dated: September 2012 and all international interpretations.

Requirement	Auditable Event
FCS_CKM_EXT.1	Cryptographic Key Generation Services
FCS_CKM.1	Cryptographic Key Generation
FCS_CKM.2	Cryptographic Key Establishment
FCS_COP.1(1)	Cryptographic Operation - Encryption/Decryption
FCS_COP.1(2)	Cryptographic Key Establishment
FCS_COP.1(3)	Cryptographic Operation - Encryption/Decryption
FCS_COP.1(4)	Cryptographic Operation - Signing
FCS_RBG_EXT.1	Cryptographic Operation - Keyed-Hash Message Authentication
FCS_RBG_EXT.2	Random Bit Generation from Application
FCS_STO_EXT.1	Storage of Secrets
FCS_TLSC_EXT.1	TLS Client Protocol
FDP_DEC_EXT.1	Access to Platform Resources
FDP_DAR_EXT.1	Encryption Of Sensitive Application Data
FIA_X509_EXT.1	X.509 Certificate Validation
FIA_X509_EXT.2	X.509 Certificate Authentication
FMT_MEC_EXT.1	Supported Configuration Mechanism
FMT_CFG_EXT.1	Secure by Default Configuration
FMT_SMF.1	Specification of Management Functions
FPT_API_EXT.1	Use of Supported Services and APIs
FPT_AEX_EXT.1	Anti-Exploitation Capabilities
FPT_TUD_EXT.1	Integrity for Installation and Update
FPT_LIB_EXT.1	Use of Third Party Libraries
FTP_DIT_EXT.1	Protection of Data in Transit

Table 8 SFRs

5.1 Conventions

The CC defines operations on Security Functional Requirements: assignments, selections, assignments within selections and refinements. This document uses the following font conventions to identify the operations defined by the CC:

- Assignment: Indicated with *italicized* text;
- Refinement: Indicated with **bold** text;
- Selection: Indicated with <u>underlined</u> text;
- Iteration: Indicated by appending the iteration number in parenthesis, e.g., (1), (2), (3).

- Where operations were completed in the PP itself, the formatting used in the PP has been retained.

Explicitly stated SFRs are identified by having a label 'EXT' after the requirement name for TOE SFRs. Formatting conventions outside of operations matches the formatting specified within the PP.

5.2 Security Functional requirements

5.2.1 Cryptographic Support (FCS)

FCS_CKM_EXT.1 Cryptographic Key Generation Services

FCS_CKM_EXT.1.1

The application shall [implement asymmetric key generation].

FCS_CKM.1 Cryptographic Key Generation

FCS_CKM.1.1

The application shall 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: [ANSI X9.311998, Section 4.1]

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: [RSA-based key establishment schemes] that meets the following: [NIST Special Publication 800-56B, "Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography"] and [no other schemes].

FCS_COP.1(1) Cryptographic Operation - Encryption/Decryption

FCS_COP.1.1(1)

The application shall perform encryption/decryption in accordance with a specified cryptographic algorithm

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

and [no other modes] and cryptographic key sizes 128-bit key sizes and [256-bit key sizes].

FCS_COP.1(2) Cryptographic Operation - Hashing

FCS_COP.1.1(2) The application shall perform cryptographic hashing services in accordance with a specified cryptographic algorithm [SHA-1, SHA-256] and message digest sizes [160, 256] bits that meet the following: FIPS Pub 180-4.

FCS_COP.1(3) Cryptographic Operation - Signing

FCS_COP.1.1(3)

The application shall perform cryptographic signature services (generation and verification) in

accordance with a specified cryptographic algorithm [RSA schemes using cryptographic key sizes of 2048-bit or greater that meet the following: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 4].

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

FCS_COP.1.1(4)

The application shall perform keyed-hash message authentication in accordance with a specified cryptographic algorithm HMAC-SHA-256 and [SHA-1] with key sizes [160 bits] and message digest sizes 256 and [160] bits that meet the following: FIPS Pub 198-1 The Keyed-Hash Message Authentication Code and FIPS Pub 180-4 Secure Hash Standard.

FCS_RBG_EXT.1 Random Bit Generation Services

FCS_RBG_EXT.1.1

The application shall [implement DRBG functionality] for its cryptographic operations

FCS_RBG_EXT.2 Random Bit Generation from Application

FCS_RBG_EXT.2.1

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

FCS_RBG_EXT.2.2

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

FCS_STO_EXT.1 Storage of Secrets

FCS_STO_EXT.1.1

The application shall [implement functionality to securely store [digital certificates] to non-volatile memory.

FCS_TLSC_EXT.1 TLS Client Protocol

FCS_TLSC_EXT.1.1

The application shall [implement TLS 1.2 (RFC 5246)] supporting the following ciphersuites:

Mandatory Ciphersuites: TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246

Optional Ciphersuites: [

- No other cipher suites].

FCS_TLSC_EXT.1.2

The application shall verify that the presented identifier matches the reference identifier according to

RFC 6125.

FCS_TLSC_EXT.1.3

The application shall only establish a trusted channel if the peer certificate is valid.

5.2.2 User Data Protection (FDP)

FDP_DEC_EXT.1 Access to Platform Resources

FDP_DEC_EXT.1.1

The application shall provide user awareness of its intent to access [network connectivity].

FDP_DEC_EXT.1.2

The application shall provide user awareness of its intent to access [system logs].

FDP_DEC_EXT.1.3

The application shall only seek access to those resources for which it has provided a justification to access.

FDP_DEC_EXT.1.4

The application shall restrict network communication to [[respond to [*downloaded scanning policies (sending information to the associated FireEye HX appliance, as defined)*]], *polling and downloading new scanning policies to be used to identify potential intrusions on the host OS from the associated FireEye HX appliance*]].

FDP_DEC_EXT.1.5

The application shall [not transmit PII over a network].

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] in non-volatile memory.

5.2.3 Identification and Authentication (FIA)

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 and that the CA flag is set to TRUE for all CA certificates.
- The application shall validate the revocation status of the certificate using [a Certificate Revocation List (CRL) as specified in RFC 5759].

- The application shall validate the extendedKeyUsage 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 extendedKeyUsage 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 extendedKeyUsage 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 extendedKeyUsage field.
    - OCSP certificates presented for OCSP responses shall have the OCSP Signing purpose (id-kp 9 with OID 1.3.6.1.5.5.7.3.9) in the extendedKeyUsage 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 extendedKeyUsage field.

FIA_X509_EXT.1.2

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

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)

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.

FMT_CFG_EXT.1 Secure by Default Configuration

FMT_CFG_EXT.1.1

The application shall only provide 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 it and its data from unauthorized access.

FMT_SMF.1 Specification of Management Functions

FMT_SMF.1.1

The TSF shall be capable of performing the following management functions [no management functions].

5.2.5 Protection of TSF (FPT)

FPT_API_EXT.1 Use of Supported Services and APIs

FPT_API_EXT.1.1

The application shall only use supported platform APIs.

FPT_AEX_EXT.1 Anti-Exploitation Capabilities

FPT_AEX_EXT.1.1

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

FPT_AEX_EXT.1.2

The application shall [not allocate any memory region with both write and execute permissions].

FPT_AEX_EXT.1.3

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

FPT_AEX_EXT.1.4

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

FPT_AEX_EXT.1.5

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

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 be distributed using the format of the platform-supported package manager.

FPT_TUD_EXT.1.3

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.1.4

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

FPT_TUD_EXT.1.5

The application shall [provide the ability] to query the current version of the application software.

FPT_TUD_EXT.1.6

The application installation package and its updates shall be digitally signed such that its platform can

cryptographically verify them prior to installation.

FPT_LIB_EXT.1 Use of Third Party Libraries

FPT_LIB_EXT.1.1

The application shall be packaged with only [*Audits.dll, and on 32-bit systems: Libeay32.dll, SSLeay32.dll, Msvcr120.dll*].

5.2.6 Trusted Path/Channel (FTP)

FTP_DIT_EXT.1.1

The application shall [encrypt all transmitted data with [TLS]] between itself and another trusted IT product.

5.3 TOE SFR Dependencies Rationale for SFRs

The Protection Profile for Application Software contains all the requirements claimed in this Security Target. As such, the dependencies are not applicable since the PP has been approved.

5.4 Security Assurance Requirements

The TOE assurance requirements for this ST are taken directly from the Protection Profile for Application Software which are derived from Common Criteria Version 3.1, Revision 4. The assurance requirements are summarized in the table below.

Assurance Class	Components	Components Description
Development	ADV_FSP.1	Basic Functional Specification
Guidance Documents	AGD_OPE.1	Operational User Guidance
	AGD_PRE.1	Preparative User Guidance
Life Cycle Support	ALC_CMC.1	Labeling of the TOE
	ALC_CMS.1	TOE CM Coverage
	ALC_TSU_EXT.1	Timely Security Updates
Tests	ATE_IND.1	Independent Testing – Conformance
Vulnerability Assessment	AVA_VAN.1	Vulnerability Analysis

Table 9 Security Assurance Requirements

5.5 Rationale for Security Assurance Requirements

The functional specification describes the external interfaces of the TOE; such as the means for a user to invoke a service and the corresponding response of those services. The description includes the interface(s) that enforces a security functional requirement, the interface(s) that supports the enforcement of a security functional requirement, and the interface(s) that does not enforce any security functional requirements. The interfaces are described in terms of their purpose (general goal of the interface), method of use (how the interface is to be used), parameters (explicit inputs to and outputs from an interface that control the behavior of that interface), parameter descriptions (tells what the parameter is in some meaningful way), and error messages (identifies the condition that generated it, what the message is, and the meaning of any error codes). The development evidence also contains a tracing of the interfaces to the SFRs described in this ST.

5.6 Assurance Measures

The TOE satisfies the identified assurance requirements. This section identifies the Assurance Measures applied by FEYE to satisfy the assurance requirements. The table below lists the details.

SAR Component	How the SAR will be met
ADV_FSP.1	The functional specification describes the external interfaces of the TOE; such as the means for a user to invoke a service and the corresponding response of those services. The description includes the interface(s) that enforces a security functional requirement, the interface(s) that supports the enforcement of a security functional requirement, and the interface(s) that does not enforce any security functional requirements. The interfaces are described in terms of their purpose (general goal of the interface), method of use (how the interface is to be used), parameters (explicit inputs to and outputs from an interface that control the behavior of that interface), parameter descriptions (tells what the parameter is in some meaningful way), and error messages (identifies the condition that generated it, what the message is, and the meaning of any error codes).
AGD_OPE.1	The Administrative Guide provides the descriptions of the processes and procedures of how the administrative users of the TOE can securely administer the TOE using the interfaces that provide the features and functions detailed in the guidance.
AGD_PRE.1	The Installation Guide describes the installation, generation, and startup procedures so that the users of the TOE can put the components of the TOE in the evaluated configuration.
ALC_CMC.1	The Configuration Management (CM) documents describe how the consumer identifies the evaluated TOE. The CM documents identify the configuration items, how those configuration items are uniquely identified, and the adequacy of the procedures that are used to control and track changes that are made to the TOE. This includes details on what changes are tracked and how potential changes are incorporated.
ALC_CMS.1	
ALC_TSU_EXT.1	FEYE uses a systematic method for identifying and providing security relevant updates to the TOEs users via its support infrastructure.
ATE_IND.1	FEYE will provide the TOE for testing.
AVA_VAN.1	FEYE will provide the TOE for testing.

Table 10 TOE Security Assurance Measures

6 TOE Summary Specification

This chapter identifies and describes how the Security Functional Requirements identified above are met by the TOE.

TOE SFR	Rationale
FCS_CKM_EXT.1/ FCS_CKM.1	The TOE uses its own cryptographic implementation to generate asymmetric RSA key pairs in accordance with ANSI X9.31 1998, Section 4.1, in support of TLS sessions. Details regarding the TOE's conformance to ANSI X9.31 can be found in section 6.1 below. The TOE leverages an RSA key-length of 2048-bits. Of note, the TOE does not use the RNG specific in A.2.4 of the ANSI X9.31. Instead, the TOE uses the SP 800-90A DRBG specified throughout this document.
FCS_CKM.2	The TOE implements a random number generator for RSA key establishment schemes (conformant to NIST SP 800-56B). See Table 3 for validation details. For RSA Key Establishment, the TOE implements the all sections of SP 800-56B. The TOE does not perform any operation marked as "Shall Not" or "Should not" in SP 800-56B. Additionally, the TOE does not omit any operation marked as "Shall." For key establishment, the TOE is only a session initiator. The TOE does not support any listening ports to establish TLS connections. Because of this, the TOE always acts as a RSA key establishment sender.
FCS_COP.1(1)	The TOE provides symmetric encryption and decryption capabilities using AES in CBC mode (128 and 256 bits) as described in NIST SP 800-38A. See Table 3 for validation details.
FCS_COP.1(2)	The TOE provides cryptographic hashing services using SHA-1 and SHA-256 with message digest sizes 160 and 256 bits respectively, as specified in FIPS Pub 180-4 "Secure Hash Standard." These hashes are used as part of TLS session negotiation and with HMACs used to verify the integrity of TLS traffic. The hash functions are also used in conjunction with RSA as part of the HX server certificate verification. See Table 3 for validation details.
FCS_COP.1(3)	The TOE provides cryptographic signature services using RSA Digital Signature Algorithm with key size of 2048 and greater as specified in FIPS PUB 186-4, "Digital Signature Standard". See Table 3 for validation details.
FCS_COP.1(4)	The TOE provides keyed-hashing message authentication services using HMAC-SHA-1 and SHA-256 with 160-bit key size and message digests sizes 160 and 256 bits respectively, as specified in FIPS Pub 198-1, "The Keyed-Hash Message Authentication Code," and FIPS 180-4, "Secure Hash Standard."
FCS_RBG_EXT.1	The Random Bit Generation used as part of xAgent TLS connections and certificate generation is provided by a TOE implementation of the SP 800-90a CTR_DRBG_AES Deterministic Random Bit Generator. The DRBG implementation has been CAVP test (Cert #1103 and #1104). The TOE uses OpenSSL (which is integrated into the TOE binary) to provide cryptographic services including approved SP 800-90B DRBG (256-bit AES CTR). This DRBG is seeded using RAND_add() API. The information on the entropy source has been provided to NIAP as part of this evaluation.
FCS_RBG_EXT.2	
FCS_STO_EXT.1	The TOE stores digital certificates in a TOE JSON structure stored in non-volatile memory.

TOE SFR	Rationale
	The database is encrypted using the TOE provided AES algorithm and is not accessible to any external entity.
FCS_TLSC_EXT.1	In support of secure communication with external entities, the TOE supports the TLS protocol. TLS is used to facilitate communication with the following entities, • HX Series Appliances • The TOE only communicates with the HX appliance using TLS_RSA_WITH_AES_128_CBC_SHA X.509 certificates used for this connection are validated using the certificate path validation algorithm defined in RFC 5280. This includes performing a bit-by-bit verification of the reference identifier. For this TOE, the reference identifier is a CN that contains the uuid of the peer HX. Additionally, the TOE supports certificate pinning. This means, if there are any changes to the certificate associated with its peer HX (including the reference identifier), the TOE will reject the certificate.
FDP_DEC_EXT.1	The TOE never processes or sends PII data outside the boundary of the host platform. The only external communication that is supported by the TOE is with the associated HX appliance. The communication consists of a fast-polling channel on port 80 which is used to receive a Boolean value about whether there are further instructions to receive. If there are, a TLS-protected channel on Port 443 is initiated with the HX and instructions or updates are transferred via the TLS session. This channel is used to download new scanning policies. The TOE then acts on these policies (e.g., performing scans on the platform). These downloaded policies may also include instructions to send the results of the scanning to the associated HX. In these cases, the TOE again initiates a TLS-protected channel on Port 443 as before. The TOE never accesses any other host platform hardware functionality besides network connectivity. Depending upon the contents of the policies the TOE receives from the associated HX appliance, the TOE may access the host OS syslog. The contents of memory are scanned as well, leveraging the functionality provided by a kernel driver (fekern.sys) which is installed with the TOE.
FDP_DAR_EXT.1	The only information that is stored by the TOE are the policies, the TOE identity, and associated HX identity which are downloaded from the associated HX series appliance. This data is protected by the Windows platform using the OS provided services. The provided guidance documentation provides instructions to ensure that BitLocker is enabled in the evaluated configuration. No other data is stored by the xAgent.
FIA_X509_EXT.1 FIA_X509_EXT.2	The TOE uses X.509v3 certificates as defined by RFC 5280 to support authentication for TLS connections. The X.509 certificates are validated using the certificate path validation algorithm defined in RFC 5280, which can be summarized as follows: • the public key algorithm and parameters are checked • the current date/time is checked against the validity period • revocation status is checked • issuer name of X matches the subject name of X+1 • name constraints are checked • policy OIDs are checked • policy constraints are checked; issuers are ensured to have CA signing bits

TOE SFR	Rationale
	• path length is checked • critical extensions are processed The TOE accepts only CRL files managed by the PKI service on the HX Management Console to determine whether HX certificates have been revoked. If the PKI service is down and the CRL is unavailable, the last known state of the HX certificate is accepted.
FMT_MEC_EXT.1	The xAgent software does not provide any Security Relevant configuration options for the software. The software is an agent that installs on the host systems OS. Once installed, the product only allows very limited interaction with the host OS user.
FMT_CFG_EXT.1	The TOE does not use authentication for the users of the host OS. The certificate used to communicate with the HX appliance is generated by the TOE at initial installation. No other functionality is available until after the TOE is installed on the host platform. No modifications may be made to the xAgent or its associated data by any user of the host platform.
FMT_SMF.1	The TOE is pushed to the host platform by the HX appliance completely configured. At no time does the TOE user perform any management of the software.
FPT_API_EXT.1	The TOE leverages the following platform provided Application Programing Interfaces (APIs), • KERNEL32.dll • WS2_32.dll • PSAPI.dll • IPHLPAPI.dll • RPCRT4.dll • CRYPT32.dll • USERENV.dll • NETAPI32.dll • pdh.dll • SETUPAPI.dll • WTSAPI32.dll • USER32.dll • SHELL32.dll • ADVAPI32.dll
FPT_AEX_EXT.1	The TOE never allocates memory with both write and execute permission. Write execution is always separate from execute. The TOE is designed to operate in an environment in which the following security techniques are in effect, Data execution prevention, Mandatory address space layout randomization (no memory map to an explicit address), Structured exception handler overwrite protection, Export address table access filtering, Anti-Return Oriented Programming, and SSL/TLS certificate trust pinning. This allows the TOE to operate in an environment in which the Enhanced Mitigation Experience Toolkit is also running. TOE executables are written to “C:\Program Files (x86)\FireEye\xagt”, in which no other files are written. In particular, no executable files are co-located in the directory in which the software is installed. During compilation the TOE is built with several flags enabled that check for engineering flaws. The TOE is built with the /GS flag enabled. This reduces the possibilities of stack-based buffer

TOE SFR	Rationale
	overflows in the product.
FPT_TUD_EXT.1	The TOE is distributed as a Microsoft .MSI file. After initial installation, all updates to the xAgent are distributed as .MSI files. The TOE software version can be queried via the Microsoft command prompt. TOE updates are signed using digital certificates. The MSI packages are signed using certificates with a public trust chain which leads to Verisign. Some components of the installation package (for instance, containment driver), are signed using the Mandiant/FireEye internal CA. Updates are distributed by being pushed to the TOE via the associated HX appliances. Only the configured HX appliance (which cannot be changed after installation) may push updates to the TOE. And updates will only be performed when pushed by the associated HX appliance. The TOE provides the ability to completely remove all application files when uninstalled.
FPT_LIB_EXT.1	On 64-bit platforms, the "Program Files (x86)" directory contains only the TOE binary (xagt.exe) and one supporting, vendor-authored dll (audits.dll). On 32-bit binaries, the following supporting DLLs are brought with the TOE into the installation directory: • Libeay32.dll (OpenSSL) • Ssleay32.dll (OpenSSL) • Msvcr120.dll (Microsoft Visual C Redistributable package)
FTP_DIT_EXT.1	The TOE communicates externally with one trust IT entity, the FireEye HX Series appliances. The xAgent periodically polls the HX appliance for policy updates. To do this the TOE initiates a TLS 1.2 secured tunnel using the TOE cryptographic implementation. Updates to the scanning policies are sent through this TLS 1.2 tunnel. No additional information is sent from the TOE.

Table 11 TOE Summary Specification SFR Description

6.1 ANSI X9.31 1998 Conformance

The following table describes the TOEs conformance to ANSI X9.31 1998, Section 4.1.

Section	Statement	Conformant?
4.1.1	Each signatory shall select a positive integer e as its public exponent, where 2 £ e < 2k-160, and k is the length of the modulus n in bits.	Yes
	If e is randomly generated, it shall be odd (both the high order bit and low order bit of e is a binary 1)	Yes
4.1.2	Each signing entity shall secretly and randomly select two distinct positive primes, p and q	Yes
	Large prime factors, p1, p2, q1, and q2, are randomly selected from the set of prime numbers between 2100 and 2^120, and each shall pass at least 27 iterations of Miller-Rabin	Yes
	The private prime factor p shall be the first discovered prime greater than a random number Xp	Yes
	the private prime factor q shall be the first discovered prime greater than a random number Xq	Yes

	The random numbers, Xp and Xq, shall be chosen using a random or pseudo-random number generator algorithm specified in an ANSI X9 standard.	No, the TOE uses the implemented SP 800-90a DRBG rather than one specified in an ANSI X9 standard.
	The private prime factors, p and q, shall pass at least 8 rounds of the Miller-Rabin probabilistic primality test followed by a single round of the Lucas test	Yes
	The private prime factors, p and q, shall be different by at least one of the first 100 bits	Yes
4.1.2.1	This shall be done by generating four random numbers Xp1, Xp2, Xq1, and Xq2.	Yes
	The size of these random numbers shall be chosen from an interval, see Figure 1 Random Number Interval at least [2100+a, 2101+a-1], such that 2100 £ 2100+a £ 2121-1	Yes
	The random numbers, Xp1, Xp2, Xq1, and Xq2, shall be chosen using a random or pseudo-random number generator algorithm specified in an ANSI X9 standard	No, the TOE uses the implemented SP 800-90a DRBG rather than one specified in an ANSI X9 standard.
	They shall be validated with at least 27 iterations of the Miller-Rabin (or equivalent) algorithm	Yes
	If not, the generation of Xq for finding q shall be repeated until this constraint is satisfied.	Yes
4.1.3	The private signature exponent, d, shall be a positive integer value such that d > 2512+128s, where s is the integer s ³ 0	Yes
	In the extremely rare event that d £ 2512+128s, then the key generation process shall be repeated with new seeds for Xq1, Xq2, and Xq	Yes
B.2	The TOE fully implementation section B.2. No shall/should functionality is unsupported.	
B.3	The TOE fully implementation section B.2. No shall/should functionality is unsupported.	
B.4	The TOE fully implementation section B.4. No shall/should functionality is unsupported.	

Table 12 ANSI X9.31 Conformance