NetinDS Security Target

Date: 2024-04-24

Created by

Change History

Version	Date	Author	Comment
0.1	2021/09/08	Netin Systems SL	First draft
0.2	2021/10/13	Netin Systems SL	Protection profile updates and Security Target amended
0.3	2021/11/11	Netin Systems SL	Minor changes due to the CPSTIC's feedback, minor typographic errors, and FPT_TUD_EXT.1.5 has been deleted.
0.4	2021/12/22	Netin Systems SL	Changes in FMT_SMF.1
0.5	2022/09/22	Netin Systems SL	The TOE version has been amended.
0.6	2022/09/25	Netin Systems SL	NCs have been fixed.
0.7	2022/09/29	Netin Systems SL	The TOE version has been amended, assumptions have been added, some SFRs have been modified and minor changes in section 7.
0.8	2022/10/18	Netin Systems SL	Minor changes.
0.9	2022/11/08	Netin Systems SL	NCs have been fixed. Replaced FCS_CKM.1/(3) by FCS_CKM_EXT.2. Updated FCS_CKM.4 dependency rationales
0.10	2022/12/22	Netin Systems SL	TOE version has been updated. NCs have been fixed.
0.11	2023/01/13	Netin Systems SL	Stated HMAC-SHA-1 as a "legacy" algorithm in accordance with SOGIS.
0.12	2023/04/11	Netin Systems SL	Third-party libraries updated. Some dependencies of requirements of section 5 updated. Dependencies of section 6.3.3 updated.
0.13	2023/05/31	Netin Systems SL	Modified FPT_TUD_EXT.2.2 due to non-compliance. Removed FPT_LIB_EXT.1.
0.14	2023/08/07	Netin Systems SL	Minor changes in instances of WebUI, removed Netin doc from physical scope and removed references to FPT_LIB_EXT.1
0.15	2024/04/24	Netin Systems SL	Assumptions and OE objectives clarified

Table of contents

1 ST INTRODUCTION

1.1 ST REFERENCE

Title: NetinDS Security Target

Version: v0.15

Author: Netin Systems SL

Evaluation Lab: jtsec Beyond IT Security

Date of publication: 2024-04-24

This is the Security Target for the Common Criteria Evaluation of the NetinDS Standalone software. The TOE defined in this ST strictly consists on a monitoring software running of industrial technology devices that meets minimum security requirements.

This Security Target includes in section 6 a set of Security Functional Requirements that are taken from **[PP_APP]**. Such functional requirements have been minimally adapted or some of them have not been included when the application does not directly implement the functionality to execute such SFR.

This table shows the SFRs that have been modified in relation with the **[PP_APP]** and the modification done:

Security Function Requirement (SFR)	Modification
FPT_AEX_EXT.1.1	This SFR has been modified because the application is made up of multiple binaries, most of them compiled with the /ASLR defense flag. The modification has been made in order to point out the list of binaries that meet this condition. The list of binaries that meet the condition can be found in the TOE Summary Specification section dedicated to this SFR.
FPT_AEX_EXT.1.2	This SFR has been modified because the application is made up of multiple binaries, most of them compiled with the

	/NXCOMPACT defense flag. The modification has been made in order to point out the list of binaries that meet this condition. The list of binaries that meet the condition can be found in the TOE Summary Specification section dedicated to this SFR.
FPT_AEX_EXT.1.5	This SFR has been modified because the application is made up of multiple binaries, most of them compiled with the /GS defense flag. The modification has been made in order to point out the list of binaries that meet this condition. The list of binaries that meet the condition can be found in the TOE Summary Specification section dedicated to this SFR.
FCS_CKM.1/(3)	This SFR has been replaced by FCS_CKM_EXT.2 in order to maintain consistency with the CC standard. The content of the SFR remains the same than its predecessor.
FCS_RBG_EXT.1.1 FCS_RBG_EXT.2.1 FCS_CKM_EXT.1.1 FCS_HTTPS_EXT.1.1 FCS_TLS_EXT.1.1 FCS_TLSS_EXT.1.1 FCS_TLSS_EXT.1.2 FCS_TLSS_EXT.1.3 FDP_DEC_EXT.1.1 FDP_DEC_EXT.1.2 FDP_NET_EXT.1.1	These SFRs have been modified by changing "The application" to "The TSF" in order to make their definition consistent.

FDP_DAR_EXT.1.1 FMT_MEC_EXT.1.1 FMT_CFG_EXT.1.1 FPR_ANO_EXT.1.1 FPT_API_EXT.1.1 FPT_AEX_EXT.1.1 FPT_AEX_EXT.1.2 FPT_AEX_EXT.1.3 FPT_AEX_EXT.1.4 FPT_AEX_EXT.1.5 FPT_IDV_EXT.1.1 FPT_TUD_EXT.1.1	
FCS_STO_EXT.1.1	This SFR has been modified by changing "The application" to "The TSF" and adding the FCS_COP.1 assignment in order to be consistent with its definition.
FCS_HTTPS_EXT.1.2	This SFR has been modified by changing "The application" to "The TSF" and rewritten to be consistent with FCS_TLS_EXT.1 and FCS_TLSS_EXT.1 dependencies.
FTP_DIT_EXT.1.1	This SFR has been modified by changing "The application" to "The TSF" and the selections corresponding to the DTLS, SSH protocols have been removed to be consistent with its definition.
FPT_TUD_EXT.2.2	The aforementioned SFR has undergone modifications wherein the section of the requirement that initially demanded the application to be packaged in a manner

	enabling removal through the operating system has been substituted. Instead, a new requirement has been introduced, which delegates the responsibility of uninstalling the TOE directly to the TSF.

1.2 TOE REFERENCE

TOE Name: NetinDS

TOE Developer: Netin Systems SL

TOE Version: 2.0.2

1.3 TOE OVERVIEW

1.3.1 INTRODUCTION

Netin Diagnostic System, better known as NetinDS, is a system for monitoring and diagnosing industrial installations and OT infrastructures, whose objective is to provide professionals with the necessary tools for the diagnosis of their installations. Designed and developed for the industry, NetinDS relies on the world's leading IT monitoring protocols as well as the most well-known and widespread OT standards. It is developed by Netin Systems SL.
Integration with the IIoT platform (Industrial Internet of Things) in the way of digitalization is one of its main bases, and together with the integration with IT systems, it allows the creation of the necessary ecosystem to have the information available when and where it is needed.
NetinDS helps with maintenance and operation tasks, making it possible to anticipate possible problematic situations and resolve them more efficiently.

1.3.2 TOE TYPE

NetinDS is a Network Management Software specially designed for monitoring industrial technology devices.

1.3.3 TOE USAGE & MAJOR SECURITY FEATURES

1.3.3.1 TOE USAGE

NetinDS is a distributed and agent-based system that monitors large OT infrastructures and modern industrial automation systems. As opposed to specific network monitoring devices, NetinDS is a

software product. It relies on a monitoring and diagnosing industrial installations and OT infrastructures, the aim of which is to provide industry professionals with the tools they need to diagnose their installations.
Through an agent structure, NetinDS is allowed to driver all the systems and to integrate them under one tool. The main components of the TOE are mentioned below:

- NetinDS Server: main axis of the NetinDS system, performs tasks of coordination of the agent, historization and storage of information, NetinDS configuration management and artifacts management and configuration. It is constituted by both backend and frontend of the application.

- Nginx: Nginx is an open-source web server software that functions as a reverse proxy, load balancer, and HTTP cache. Nginx acts as a web server for HTTPS connections. It uses the OpenSSLv1.1.1m library to implement the TLSv1.2 protocol, ensuring secure communication between the web browser and the TOE.

- NetinDS Agent: it is responsible for capturing and managing information from the external IT entities. In addition, the agent is responsible for generating analytics information that is sent to the NetinDS Server in order to be displayed to users.
    - Artifacts: Part of NetinDS Agent, an artifact is a modular unit that monitors and communicates with external IT entities. They are managed by administrators. They are blocks of code that allow NetinDS Agent connect with the external IT entities using a specific protocol.

This is a table of all the artifacts that can be used in the TOE:

Component	**Usage/Purpose description for TOE performance**
drv-SNMP	This is driver for SNMPv3
drv-ICMP	This is driver for ICMP
drv-DCP	This is driver for DCP
drv-MQTT	This is driver for MQTTv3.1.1

The TOE is deployed and runs on a general-purpose computer connected to the same local area network as the external IT entities that is able to monitor. Once the operating environment is properly prepared according to the evaluated configuration described in TOE Evaluated Configuration section 1.4.1.1, including the configuration of artifacts, the TSF can be deployed.

1.3.3.2 MAJOR SECURITY FEATURES

The TOE is comprised of several security features. Below are identified the security features and which of them are considered TSF:

- Cryptographic support. The objective is to help to satisfy several high-level security objectives. These include secure storage of passwords, and data integrity, trusted channel. This is done in order to establish secure channels for remote TOE's administration and communication between the TOE and external IT entities.

- Security Management. Specifies the management of several aspects of the TSF: security attributes and TSF data and functions. Via the web interface, it is possible for administrators the management of the agent, artifacts, users, configuration and maintenance.

- Trusted path. Provides protection of all the communications between the TOE and users and administrators and external IT entities. Trusted channels are provided by SNMPv3 and TLSv1.2. In parallel, TLS provides support for MQTTv3.1.1 and HTTPS protocols.

- User data protection and privacy. The TOE limits its access to necessary hardware resources and information repositories. Data at non-volatile memory is encrypted. The TOE does not share PII with third parties.

- Protection of the TSF. The application has been developed with attack prevention mechanisms in accordance to high quality standards. The TOE incorporates memory anti-exploitation capabilities. The TOE is versioned, signed and distributed as additional software. APIs used by the TOE are documented.

1.3.4 NON-TOE HARDWARE/SOFTWARE/FIRMWARE

The TOE needs a general-purpose computer with 64-bit Windows 10 Pro operating system running.

One of the following web browsers "Microsoft Edge 97.0.1072.62", "Google Chrome v97.0.4692.71" or "Mozilla Firefox v 92.0" or higher versions are needed as the endpoint of the communication used by the users and administrators for the establishment of secure communication with the TOE through HTTPS.

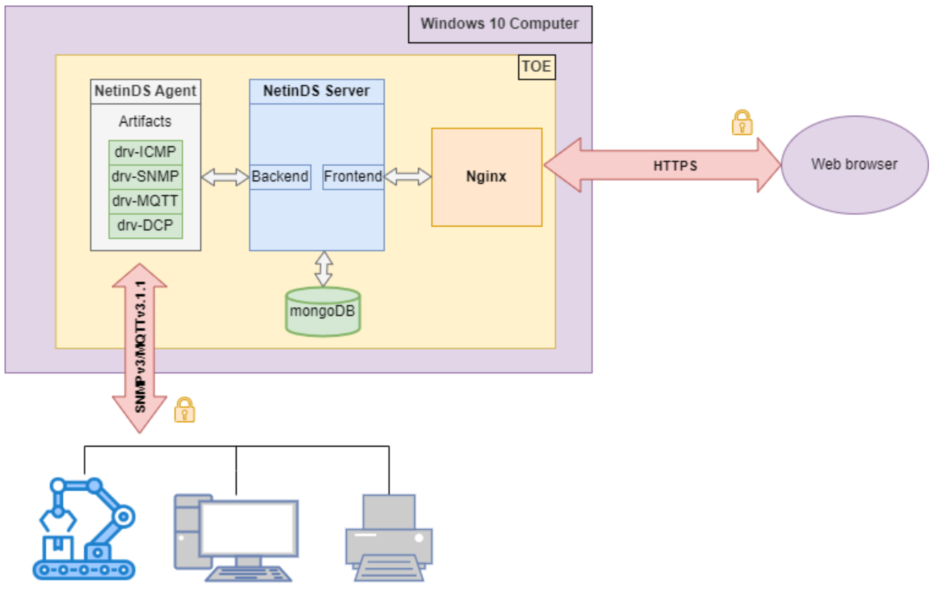
During the installation process, two dependencies must be installed if they are not already in the computer for the correct functioning of the TOE: Winpcap v4.1.3 and Microsoft Visual C++ v14.16

1.4 TOE DESCRIPTION

1.4.1 INTRODUCTION

NetinDS is a distributed and agent-based system that monitors large OT infrastructures and modern industrial automation systems.
The product doesn't provide a single deployment environment, since it allows a properly configuration accomplished with the specific operational environment and size of the organization. NetinDS, in its Standalone deployment mode, is the one covered from the scope of the Common Criteria Certification. In this version, NetinDS acts like agent and server as a single deployment.

The architecture of the standalone mode of NetinDS is shown in the diagram below:



The TOE is installed in a general-purpose computer with Windows 10 operating system. It is a software distributed as a .exe file that makes the functions of agent and server as needed. NetinDS Server is accessed through the web browser (via Nginx that serves the Server Frontend to the user). This is used to configure and to operate the TOE. The communication with the TOE is done via HTTPS that deploys a secure channel between Nginx and the web browser. When a user or an administrator wants to access the TOE, an authentication password-based process is done. Actions like successful login or authentication error are audited and saved to disk. The communication between Server and Agent is made via REDIS, which is an open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker. NetinDS Agent is in charge of the communication with the external IT entities. This communication takes place thanks to artifacts configured through NetinDS Server by the Configuration Administrator. The secure channel used for this communication is supported by the cryptographic functionality of the TOE and they are SNMPv3 and MQTTv3.1.1.

1.4.1.1 TOE EVALUATED CONFIGURATION

NetinDS supports three different installations: NetinDS Standalone, NetinDS Server and NetinDS mix. Only **NetinDS Standalone** mode is covered by the scope of this evaluation.
NetinDS Standalone installation mode acts in a single deployment as server and agent, installed in a single computer, while in other NetinDS operational modes, server and agent are in different distributed deployments.
The TOE evaluated configuration can be described as follows:

The external IT entities associated with NetinDS are integrated by the administrators through management operations. These external IT entities will be monitored using a secure channel through SNMPv3/MQTTv3.1.1 protocol connected to NetinDS Agent. Data generated by external IT entities will be sent from NetinDS Agent to NetinDS Server to be monitored by the administrators and users.
The TOE has to be configured to use HTTPS in the secure communication channel between remote administrators and Nginx.
The evaluated configuration requires the TOE operational environment to be equipped with the NON-TOE HW/FW/SW described in section 1.3.4.
In order to achieve the above-described configuration, the TOE preparative guides (**[AGD_PRE]**) must be thoroughly followed for the TOE installation and configuration.

A minimum deployment required to obtain functional system would be as follow:

NetinDS Standalone mode (Server and Agent included).

A number of external IT entities connected to the NetinDS through a proper protocol (only SNMPv3 and MQTTv3.1.1 are allowed under the scope of certification).

Apart from these considerations, the TOE does not require any other pre-configuration or particular usage to obtain the security features described in this document beyond the TOE guide's steps.

1.4.2 TOE LOGICAL SCOPE

The TOE includes several security features. Each of the security features identified above consists of several security functionalities and are considered TOE Security Functionalities, as identified below.

1.4.2.1 CRYPTOGRAPHIC SUPPORT

The TOE provides cryptographic support to secure connections that include remote administrative management through HTTPS, and remote connection with external IT entities, through SNMPv3 and MQTTv3.1.1. HTTPS and MQTTv3.1.1's security is provided using TLSv1.2, while SNMPv3 has its own security mechanisms. The cryptographic services provided by the TOE are described in Table below. The protection of stored passwords is also carried out by the cryptographic support.

Cryptographic function	Use in the TOE
DRBG according to the NIST Special Publication 800-90A	Used in session establishment of TLSv1.2
ECDHE	Used in key exchange and session establishment of TLSv1.2

RSA 3072 bits	Used in session establishment of TLSv1.2 and TOE's authentication.
SHA-384	Used to provide cryptographic hashing support to TLSv1.2
AES-256-GCM	Used to encrypt traffic transmitted through TLSv1.2
HMAC-SHA-256	Used to hash users' passwords

1.4.2.2 TRUSTED PATH

The TOE provides trusted connections using two protocols. SNMPv3 as well as TLSv1.2 protocols are used to protect the communication channels between the TOE and the final user/administrator and between the TOE and the external IT entities.

For communication channel between the TOE and web browser, the TOE implements through Nginx the TLS 1.2 protocol to support HTTPS. Administrators are authenticated in the TOE and they can manage it using the management interface, displayed in the remote web browser. The secure channel through which administrators connect the TOE is the only way of managing and controlling the product. In contrast, users do not access the TOE to manage or control the TOE administration functionalities, but rather to perform normal-analysis TOE operation actions.

For the communication channel between the TOE and the external IT entities, the TOE provides SNMPv3 as well as TLSv1.2 to support MQTTv3.1.1. These connections are managed through the SNMPv3 and MQTTv3.1.1 artifacts. Since the communication channel with the external IT entities implement cryptographic mechanisms to protect transmitted data, the connections establish trusted paths.

In this way, the TOE protects the communication channels between the TOE and the external IT entities as well as between the TOE and the user's endpoint.

Note: Authentication process (non-evaluated feature) might take place using HTTPS and SNMPv3 (to authenticate TOE administrators, and the TOE against external IT entities respectively). These communication channels provide the necessary cryptographic mechanisms to prevent an unauthorized attacker from sniffing sensitive data.

1.4.2.3 SECURITY MANAGEMENT

The following is a list of the security management capabilities that the TOE is equipped with:

- Artifacts management, these blocks of code can be started or interrupted, managing the communication with the external entities for which they are responsible.

- Users' management, users can be created and deleted by assigning them different privileges, passwords, user IDs and personal data.

- Secure communication channels management, in the case of the SNMPv3 artifact, it can be configured SNMPv3 artifact's security parameters such as keys and cryptographic algorithms.

1.4.2.4 USER DATA PROTECTION AND PRIVACY

The TOE restricts its functionalities to those strictly necessary to carry out its activity. This includes restricting access to platform hardware resources and sensitive information repositories, as well as restricting the use of communication channels that are necessary for the correct operation of the TOE.

The TOE manages PII such as user names and mail address but none of this information is sent over the network to third parties or items that do not correspond to parts of the TOE. All data stored by the TOE on disk is encrypted so that only a trusted computer administrator can access it.

1.4.2.5 PROTECTION OF THE TSF

The TOE relies on documented external APIs to assist it in developing its functionality and it is developed with methods to prevent exploitation attacks.

The application leverages the platform for verification of digital signature of TOE installer and updater packages, which are distributed as additional software packages to the OS. The TOE allows to check the current version of the application. Updates are signed by the manufacturer, which is responsible for notifying directly to the user when a new update is released. In addition, the application executables comply with minimum security measures according to the quality policies of the platform where it is installed.

1.4.2.6 NON-TOE SECURITY FEATURES

This section includes some of the TOE functionalities that could be related to security functionality. The following security functionalities are out of the scope of the evaluation.

- Monitoring: real-time monitoring and diagnosis of all the external IT entities and systems that conform the industrial installations and OT infrastructures discovered using the discovery mode (through DCP and ICMP).

- Generation of audit data: the TOE generates audit data related to management events (administrative actions), events related to monitoring and diagnosing, and normal user events (user actions).

- Integration: it integrates the whole industrial systems and external IT entities into a single monitoring and diagnostic tool, as well as the application of industrial and proprietary standards.
- Forensic analysis: tracing, analyzing and uncovering the reasons for incident in the industrial installations with the aim to solve the problem in the most efficient way.
- Assets management: getting the most out of investments by automatically controlling and creating an inventory of all industrial hardware assets.
- Identification and authentication: The TOE identify and authenticates all users accessing it through a username and password with minimum complexity requirements.
- Access control. The TOE restricts the ability of configuration to "Configuration/User Administrators". There are three defined roles: Basic User, Configuration Administrator, User Administrator.
  - The Basic User does not have any configuration ability, it is only able to supervise the network and all the artifact's collected data.
  - The User Administrator is the administrator in charge of the management of all users. In addition to having the supervision capabilities as the Basic User, the main configuration capabilities of this administrator are the following: To create new users and assign them a user role, to modify the credentials of the other users, to enable or disable users and to delete users.
  - The Configuration administrator is the administrator in charge of the management of the configuration of the product. In addition to having the supervision capabilities as the Basic User, the main capabilities of this administrator are the following: ability to start and stop artifacts, ability to configure artifacts templates.

1.4.3 TOE PHYSICAL SCOPE

The Target of Evaluation (TOE) is purely a software TOE and includes the following components:

Delivery Item	Type	Version	Delivery Method	Format
netin-ds-agent-installer-1.0.0	Software	2.0.2	Download from an FTP server with TLS encryption	.exe

Operational User Guidance	Guidance Documentation	0.12	Download from an FTP server with TLS encryption	PDF
Preparative Procedures	Preparative Documentation	0.12	Download from an FTP server with TLS encryption	PDF
Netin Documentation	User Documentation	0.1	Download from an FTP server with TLS encryption	PDF

2 CONFORMANCE CLAIMS

This Security Target and the TOE described are in accordance with the requirements of Common Criteria 3.1R5.

This Security Target claims conformance with the following parts of Common Criteria:

- Conformance with [CC31R5P2] extended.
- Conformance with [CC31R5P3] extended.

The methodology to be used for the evaluation is described in the "Common Evaluation Methodology" of the Common Criteria standard of April 2017, version 3.1 revision 5 with an evaluation assurance level of EAL2.

This Security Target does not claim conformance with any protection profile.

3 SECURITY PROBLEM DEFINITION

This section describes the security aspects of the operational environment and its expected use in said environment. It includes the declaration of the TOE operational environment that identifies and describes:

- The alleged known threats that will be countered by the TOE
- The organizational security policies that the TOE has to adhere to
- The TOE usage assumptions in the suggested operational environment.

We will begin defining Assets and Agents of threats.

3.1 ASSETS

TOE NETWORK TRAFFIC: Traffic incoming and outcoming from the TOE Network interfaces, exchanged with external IT entities and remote administrators in its operational environment. This data can include user information such as passwords or names, as well as information from external IT entities and TOE configuration parameters.

TOE DATA: Data stored by the TOE. It can be configuration data, collected data from external IT entities or user data such as usernames, passwords or PII.

Application Note

Artifact's templates are used to configure, among other things, secure channels, provided by the protocol to which the artifact is oriented, between external IT devices and the TOE. If these templates are corrupted, secure channels can be compromised.

3.2 THREAT AGENTS

REMOTE ATTACKER: Any individual with access to the services offered by the TOE through its remote interfaces and without physical access to the TOE, using the TOE services or listening to the communication channels between the TOE and the remote administrator or the TOE and external IT entities, in a way that intends to vulnerate or compromise the security of the TOE assets.

LOCAL ATTACKER: Any individual with physical access to the TOE, gaining access to it, reading or manipulating the TOE configuration files in any way that intends to vulnerate or compromise the security of the TOE assets.

3.3 THREATS TO SECURITY

This section identifies the threats to assets that require protection by the TOE. The threats are defined in terms of assets concerned, attackers and the adverse action that materializes the threat.

T.NETWORK_ATTACK: A **REMOTE 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. This threat attempts against **TOE NETWORK TRAFFIC**.

T.NETWORK_EAVESDROP: A **REMOTE 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. This threat attempts against **TOE NETWORK TRAFFIC**.

T.LOCAL_ATTACK: A **LOCAL 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. This threat attempts against **TOE DATA**.

T.PHYSICAL_ACCESS: A **LOCAL ATTACKER** may try to access sensitive data at rest. This threat attempts against **TOE DATA**.

3.4 ASSUMPTIONS

The assumptions when using the TOE are the following:

A.PLATFORM: The TOE relies upon a trustworthy computing platform with a reliable time clock 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 (*Basic User* role) is not willfully negligent or hostile, and uses the software in compliance with the applied enterprise security policy.

A.PROPER_ADMIN: The administrators of the application software (*Configuration Administrator* and *User Administrator* roles) are not careless, willfully negligent or hostile, and administer the software in compliance with the applied enterprise security policy.

A.INTEGRITY: During the security installation, the responsible administrator must change the default user passwords or delete the default users.

A.UDPATES: During the security installation, the guides shall encourage the responsible administrator to check for software updates on a regular basis.

4 SECURITY OBJECTIVES

The security objectives are high level declarations, concise and abstract of the solution to the problem exposed in the former section, which counteracts the threats and fulfills the security policies and the assumptions. These consist of:

- the security objectives for the operational environment.
- the security objectives for the TOE

4.1 SECURITY OBJECTIVES FOR THE TOE

The security objectives for the TOE must determine (to the desired extent) the responsibility of the TOE in countering the threats and in enforcing the OSPs. Each objective must be traced back to aspects of identified threats to be countered by the TOE and to aspects of OSPs to be met by the TOE.

O.INTEGRITY: The TOE ensures the integrity of their installation and update packages, and also leverage execution environment-based mitigations. Software is seldom, if ever, shipped without errors. 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. The most sensitive executables from the 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.

O.QUALITY: To ensure quality of implementation, the TOE leverages 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.

O.MANAGEMENT: To facilitate administrator management and the general tasks for the users, the TOE provides consistent and supported interfaces for their security-relevant configuration and maintenance. This includes the user management, external IT entities monitoring management, as well as providing mechanisms for TOE configuration.

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, the TOE will use data-at-rest protection. This involves encrypting data, hashed passwords and keys stored by the TOE in order to prevent unauthorized access to this data. This also includes unnecessary network communications whose consequence may be the loss of data.

O.PROTECTED_COMMS: To address both passive (eavesdropping) and active (packet modification) network attack threats, the TOE 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.

4.2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

The security objectives for the Operational Environment determine the responsibility of the environment in countering the threats, enforcing the OSPs and upholding the assumptions. Each objective must be traced back to aspects of identified threats to be countered by the environment, to aspects of OSPs to be enforced by the environment and to assumptions to be uphold by the 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 (*Basic User* role) is not willfully negligent or hostile, and uses the software within compliance of the applied enterprise security policy.

OE.PROPER_ADMIN: The administrators of the application software (*Configuration Administrator* and *User Administrator* roles) are not careless, willfully negligent or hostile, and administer the software within compliance of the applied enterprise security policy.

OE.INTEGRITY: During the security installation, the responsible administrator must change the default user passwords or delete the default users.

OE.UPDATES: During the security installation, the guides shall encourage the responsible administrator to check for software updates on a regular basis.

4.3 SECURITY OBJECTIVES RATIONALE

The following table provides a mapping of security objectives tracing each security objective for the TOE back to threats countered by that security objective and OSPs enforced by that security objective, and each security objective for the operational environment back to threats countered by that security objective, OSPs enforced by that security objective, and assumptions upheld by that security objective. This illustrates that the security objectives counter all threats, the security objectives enforce all OSPs and the security objectives for the operational environment uphold all assumptions.

	O.INTEGRITY	O.QUALITY	O.MANAGEMENT	O.PROTECTED_STORAGE	O.PROTECTED_COMMS	OE.PLATFORM	OE.PROPER_USER	OE.PROPER_ADMIN	OE.INTEGRITY	OE.UPDATES
T.NETWORK_ATTACK	X		X		X				X	X
T.NETWORK_EAVESDROP		X	X		X					
T.LOCAL_ATTACK		X							X	X
T.PHYSICAL_ACCESS				X						
A.PLATFORM						X				
A.PROPER_USER							X			
A.PROPER_ADMIN								X		
A.INTEGRITY									X	
A.UDPATES										X

Table 1 Security Objectives vs Security Problem Definition

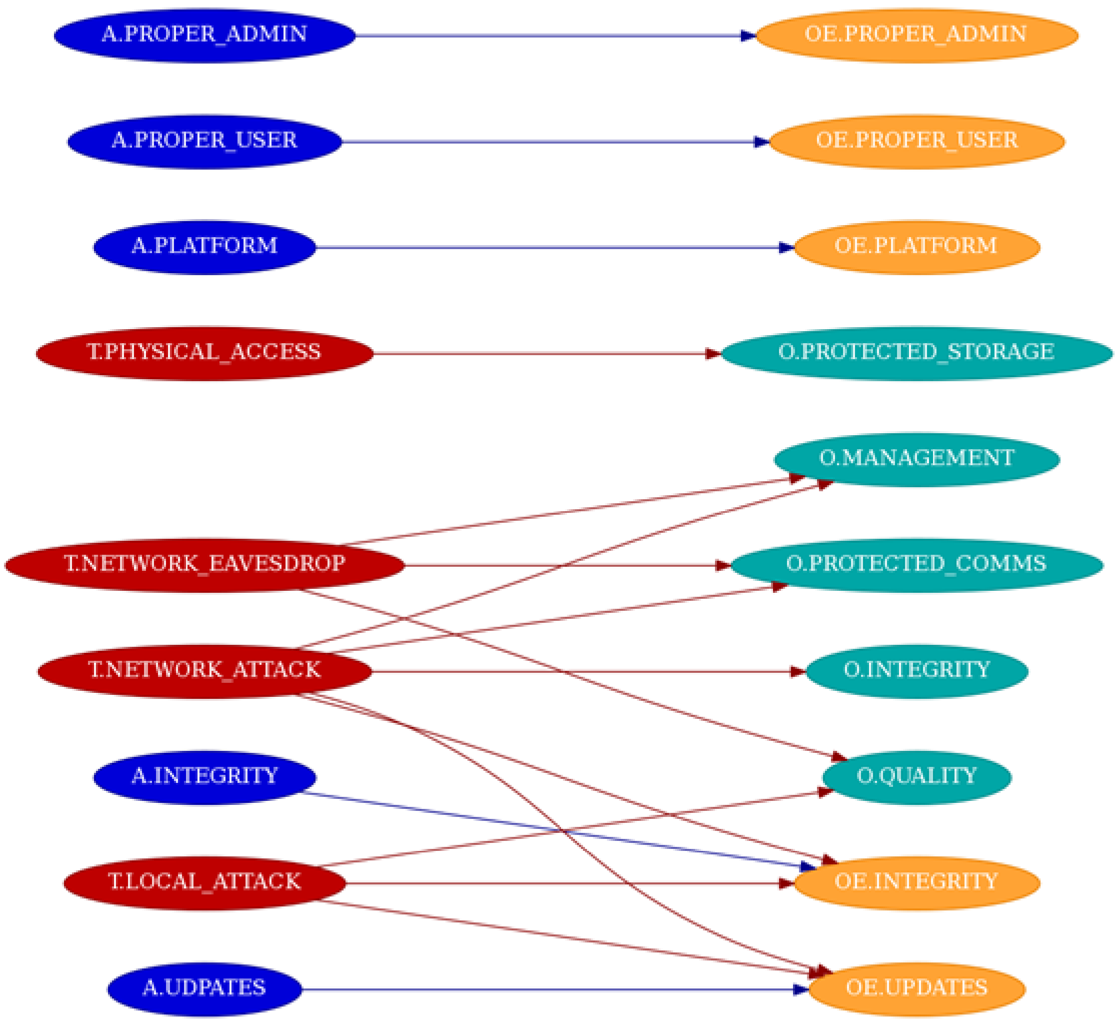


Figure 1 *Mapping of Security Problem Definition to Security Objectives*

4.3.1 THREATS

T.NETWORK_ATTACK: **O.PROTECTED_COMMS** provides for integrity of transmitted data.

O.INTEGRITY provides integrity of software that is installed onto the system from the network.

O.MANAGEMENT provides the ability to configure the application to defend against network attack using secure protocols for communication with IT entities and with the administrator.

OE.INTEGRITY ensures that an attacker with knowledge of the default credentials does not have access to the TOE to perform local attacks.

OE.UPDATES ensures that the TOE will be updated in the event that a patch addresses a vulnerability that could compromise assets.

T.NETWORK_EAVESDROP: The communication channels used by the TOE are secured as set in **O.PROTECTED_COMMS**. All channels used are strongly secured with strong cryptography supported by **O.QUALITY**, that ensures that the communication will provide protection against network-based attack. **O.MANAGEMENT** provides the ability to configure the application to protect the confidentiality of its transmitted data.

T.LOCAL_ATTACK: The objective **O.QUALITY** protects against the use of mechanisms that weaken the TOE with regard to attack by other software on the platform.

OE.INTEGRITY ensures that an attacker with knowledge of the default credentials does not have access to the TOE to perform local attacks.

OE.UPDATES ensures that the TOE will be updated in the event that a patch addresses a vulnerability that could compromise assets.

T.PHYSICAL_ACCESS: The objective **O.PROTECTED_STORAGE** protects against unauthorized attempts to access physical storage used by the TOE.

The TOE implements appropriate cryptographic mechanisms to keep sensitive data at rest in a secure and encrypted manner.

Therefore, even if the attacker accesses to the TOE directory on the same computing platform on which the product is installed, the sensitive data is protected.

The following table maps the threats of the security problem established to the security objectives of the TOE and the security objectives of the operational environment.

Threats	Security Objectives
T.NETWORK_ATTACK	O.PROTECTED_COMMS

Threats	Security Objectives
	O.INTEGRITY O.MANAGEMENT OE.INTEGRITY OE.UPDATES
T.NETWORK_EAVESDROP	O.PROTECTED_COMMS O.QUALITY O.MANAGEMENT
T.LOCAL_ATTACK	O.QUALITY OE.INTEGRITY OE.UPDATES
T.PHYSICAL_ACCESS	O.PROTECTED_STORAGE

Table 2 Threats vs Security Objectives

4.3.2 ASSUMPTIONS

A.PLATFORM: This assumption is directly upheld by **OE.PLATFORM**, which requires that the computer where the TOE is installed is trustworthy. Furthermore, the platform's administrators keep the platform maintained, updated and hardened.

A.PROPER_USER: This assumption is directly upheld by **OE.PROPER_USER**, which requires that TOE normal users (*Basic User* role) are trusted to follow and apply all guidance documentation and the administrator's guidelines in a trusted manner.

A.PROPER_ADMIN: This assumption is directly upheld by **OE.PROPER_ADMIN**, which requires that TOE administrators (*Configuration Administrator* and *User Administrator* roles) are trusted and follow and apply all security guidance documentation in a trusted manner.

A.INTEGRITY: This assumption is directly upheld by **OE.INTEGRITY**, which requires that the TOE administrators change the default passwords or remove the default users to create a new ones.

A.UPDATES: This assumption is directly upheld by **OE.UPDATES**, which requires that the guidance documentation describes how to check for TOE updates.

The following table maps the assumptions of the problem established to the security objectives of the TOE and the security objectives of the operational environment.

Assumptions	Security Objectives
A.PLATFORM	OE.PLATFORM
A.PROPER_USER	OE.PROPER_USER
A.PROPER_ADMIN	OE.PROPER_ADMIN
A.INTEGRITY	OE.INTEGRITY
A.UPDATES	OE.UPDATES

Table 3 *Assumptions vs Security Objectives for the Operational Environment*

5 EXTENDED COMPONENTS DEFINITION

5.1 EXTENDED FUNCTIONAL COMPONENTS

5.1.1 CLASS FCS: CRYPTOGRAPHIC SUPPORT

The TSF may employ cryptographic functionality to help satisfy several high-level security objectives. These include (but are not limited to): identification and authentication, non-repudiation, trusted path, trusted channel and data separation. This class is used when the TOE implements cryptographic functions, the implementation of which could be in hardware, firmware and/or software.

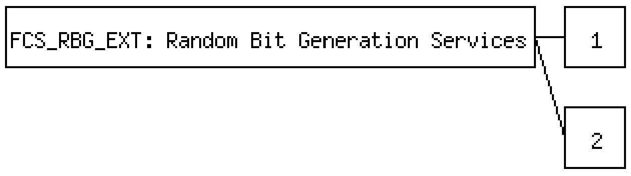
The FCS class is composed of two families: FCS_CKM and FCS_COP. The FCS_CKM family addresses the management aspects of cryptographic keys, while the FCS_COP family is concerned with the operational use of those cryptographic keys.

5.1.1.1 RANDOM BIT GENERATION SERVICES (FCS_RBG_EXT)

Family behavior

Components in this family address the requirements for random bit/number generation.

Component levelling



Management: FCS_RBG_EXT.1

There are no management activities foreseen.

Management: FCS_RBG_EXT.2

There are no management activities foreseen.

Audit: FCS_RBG_EXT.1

There are no auditable events foreseen.

Audit: FCS_RBG_EXT.2

There are no auditable events foreseen.

FCS_RBG_EXT.1: Random Bit Generation

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FCS_RBG_EXT.1.1: ***The TSF shall [selection: use no DRBG functionality, invoke platform-provided DRBG functionality, implement DRBG functionality] for its cryptographic operations***

Evaluation Activity

TSS

If use no DRBG functionality is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

If implement DRBG functionality is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.

If invoke platform-provided DRBG functionality is selected, the evaluator performs the following activities. The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers.

The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used correctly for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

Guidance

None.

Tests

If invoke platform-provided DRBG functionality is selected, the following tests shall be performed:

The evaluator shall decompile the application binary using a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

The following are the per-platform list of acceptable APIs:

The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

FCS_RBG_EXT.2: Random Bit Generation from Application

Hierarchical to:

No other components.

Dependencies:

FCS_RBG_EXT.1.

FCS_RBG_EXT.2.1: ***The TSF shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using [selection: Hash_DRBG (any), HMAC_DRBG (any), CTR_DRBG (AES)]***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall perform the following tests, depending on the standard to which the RBG conforms.

Implementations Conforming to FIPS 140-2 Annex C.

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

- Test 1: The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.

- Test 2: The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section E.3. The evaluators ensure that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

- Test 1: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality. If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. “Generate one block of random bits” means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).

If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.

The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.
Entropy input: the length of the entropy input value must equal the seed length.

Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does notuse a nonce), the nonce bit length is one-half the seed length.

Personalization string: The length of the personalization string must be less than or equal to seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.

Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

FCS_RBG_EXT.2.2: ***The deterministic RBG shall be seeded by an entropy source that accumulates entropy from a platform-based DRBG and [selection: a software-based noise source, a hardware-based noise source, no other noise source] with a minimum of [selection: 128 bits, 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***

TSS

Documentation shall be produced - and the evaluator shall perform the activities – in accordance with Appendix D of the Protection Profile - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

Guidance

None.

Tests

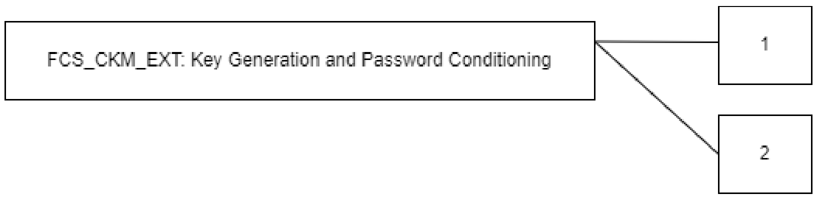
In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates

5.1.1.2 CRYPTOGRAPHIC KEY GENERATION SERVICES (FCS_CKM_EXT)

Family behavior

Components in this family define requirements for cryptographic key management beyond those which are specified in the Part 2 family FCS_CKM.

Component levelling



Management: FCS_CKM_EXT.1

There are no management activities foreseen.

Management: FCS_CKM_EXT.2

There are no management activities foreseen.

Audit: FCS_CKM_EXT.1

There are no auditable events foreseen.

Audit: FCS_CKM_EXT.2

There are no auditable events foreseen.

FCS_CKM_EXT.1: Cryptographic Key Generation Services

Hierarchical to:

No other components.

Dependencies:

See Application Note.

FCS_CKM_EXT.1.1: ***The TSF shall [selection: generate no asymmetric cryptographic keys, invoke platform-provided functionality for asymmetric key generation, implement asymmetric key generation]***

Application Note

If the selection "implement asymmetric key generation" is chosen, there will be a dependency with FCS_CKM.1.

Evaluation Activity

TSS

The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the generate no asymmetric cryptographic keys selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.

Guidance

None.

Tests

None.

FCS_CKM_EXT.2: Password Conditioning

Hierarchical to:

No other components.

Dependencies:

FCS_COP.1 and FCS_RBG_EXT.1

FCS_CKM_EXT.2.1: ***A password/passphrase shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm as specified in [assignment: FCS_COP.1 cryptographic algorithm], with [assignment: 1000 iterations or more], and output cryptographic key sizes [assignment: cryptographic key size] that meet [assignment: Recommendation for PBKDF].***

FCS_CKM_EXT.2.2: ***The TSF shall generate salts using a RBG that meets FCS_RBG_EXT.1 and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.2.***

Evaluation Activity

TSS

Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements

(FCS_COP.1.1(4)). No explicit testing of the formation of the submask from the input password is required. FCS_CKM.1.1(3): The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1.

Guidance

None.

Tests

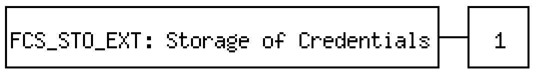
None.

5.1.1.3 STORAGE OF CREDENTIALS (FCS_STO_EXT)

Family behavior

Family created due to the need to introduce a requirement to ensure that persistent credentials (secret keys, PKI private keys or passwords) are stored securely.

Component levelling



Management: FCS_STO_EXT.1

There are no management activities foreseen.

Audit: FCS_STO_EXT.1

There are no auditable events foreseen.

FCS_STO_EXT.1: Storage of Credentials

Hierarchical to:

No other components.

Dependencies:

[FCS_COP.1 or FCS_CKM_EXT.2].

FCS_STO_EXT.1.1: ***The TSF shall [selection: not store any credentials, invoke the functionality provided by the platform to securely store [assignment: list of credentials], implement functionality to securely store [assignment: list of credentials] according to [selection: [assignment: FCS_COP.1 cryptographic algorithm], FCS_CKM_EXT.2]] to non-volatile memory***

Evalaution Activity

TSS

The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

Guidance

None.

Tests

For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/(1) or conditioned according to FCS_CKM.1.1/(1) and FCS_CKM_EXT.2. For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). The evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

5.1.1.4 HTTPS PROTOCOL (FCS_HTTPS_EXT)

Family behavior

Family created due to the need to introduce a requirement to protect remote administration sessions between the TOE and users.

Component levelling

FCS_HTTPS_EXT: HTTPS Protocol
1

Management: FCS_HTTPS_EXT.1

There are no management activities foreseen.

Audit: FCS_HTTPS_EXT.1

There are no auditable events foreseen.

FCS_HTTPS_EXT.1: HTTPS Protocol

Hierarchical to:

No other components.

Dependencies:

FCS_TLS_EXT.1 and FCS_TLSS_EXT.1

FCS_HTTPS_EXT.1.1: ***The TSF shall implement the HTTPS protocol that complies with RFC 2818.***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

FCS_HTTPS_EXT.1.2: ***The TSF shall implement HTTPS using TLS as defined in FCS_TLS_EXT.1 and FCS_TLSS_EXT.1 .***

TSS

None.

Guidance

None.

Tests

Other tests are performed in conjunction with FCS_TLS_EXT.1 and FCS_TLSS_EXT.1.

5.1.1.5 TLS PROTOCOL (FCS_TLS_EXT)

Family behavior

This family defines the requirements for explicit TLS usage

Component levelling

FCS_TLS_EXT: TLS Protocol
1

Management: FCS_TLS_EXT.1

There are no management activities foreseen.

Audit: FCS_TLS_EXT.1

There are no auditable events foreseen.

FCS_TLS_EXT.1: TLS Protocol

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FCS_TLS_EXT.1.1: ***The TSF shall implement [selection: TLS as a client, TLS as a server]***

Evaluation Activity

TSS

None.

Guidance

The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

Tests

None.

5.1.1.6 TLS SERVER PROTOCOL (FCS_TLSS_EXT)

Family behavior

The components in this family addresses the ability for a server to use TLS to protect data between a client and the server using the TLS protocol.

Component levelling

Management: FCS_TLSS_EXT.1

There are no management activities foreseen.

Audit: FCS_TLSS_EXT.1

There are no auditable events foreseen.

FCS_TLSS_EXT.1: TLS Server Protocol

Hierarchical to:

No other components.

Dependencies:

FCS_TLS_EXT.1.

FCS_TLSS_EXT.1.1: ***The TSF shall implement TLS 1.2 (RFC 5246) and [selection: TLS 1.1 (RFC 4346), no earlier TLS versions] as a server that supports the ciphersuites [selection:***
TLS_RSA_WITH_AES_128_CBC_SHA as defined in RFC 5246,
TLS_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,
TLS_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,
TLS_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5246,
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256 as defined in RFC 5246,
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5288,

TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 as defined in RFC 5289, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 as defined in RFC 5289, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 as defined in RFC 5289, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289] and no other ciphersuites, and also supports functionality for [selection: mutual authentication, session renegotiation, none]

Evaluation Activity

TSS

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

Guidance

The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

Tests

The evaluator shall also perform the following tests:

- Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
- Test 2: The evaluator shall send a Client Hello to the server with a list of cipher suites that does not contain any of the cipher suites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the server denies the connection.
- Test 3: If RSA key exchange is used in one of the selected ciphersuites, the evaluator shall use a client to send a properly constructed Key Exchange message with a modified EncryptedPreMasterSecret field during the TLS handshake. The evaluator shall verify that the handshake is not completed successfully and no application data flows.
- Test 4: The evaluator shall perform the following modifications to the traffic:

- Test 4.1: Change the TLS version proposed by the client in the Client Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the server rejects the connection.
- Test 4.2: Modify a byte in the data of the client's Finished handshake message, and verify that the server rejects the connection and does not send any application data.
- Test 4.3: Demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption): Generate a Fatal Alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, and then send a Client Hello with the session identifier from the previous incomplete session, and verify that the server does not resume the session.
- Test 4.4: Send a message consisting of random bytes from the client after the client has issued the ChangeCipherSpec message and verify that the server denies the connection.

FCS_TLSS_EXT.1.2: ***The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and [selection: TLS 1.1, none]***

Evaluation Activity

TSS

The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions consistent relative to selections in FCS_TLSS_EXT.1.2.

Guidance

The evaluator shall verify that the AGD guidance includes any configuration necessary to meet this requirement.

Tests

Test 1: The evaluator shall send a Client Hello requesting a connection with version SSL 2.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 3.0 and TLS 1.0, and TLS 1.1 if it is selected.

FCS_TLSS_EXT.1.3: ***The TSF shall perform key establishment for TLS using [selection: RSA with size [selection: 2048 bits, 3072 bits, 4096 bits, no other sizes], Diffie-Hellman parameters with size [selection: 2048 bits, 3072 bits, 4096 bits, 6144 bits, 8192 bits, no other sizes], Diffie-Hellman groups [selection: ffdhe2048, ffdhe3072, ffdhe4096, ffdhe6144, ffdhe8192, no other groups],***

ECDHE parameters using elliptic curves [selection: secp256r1, secp384r1, secp521r1] and no other curves, no other key establishment methods]

Evaluation Activity

TSS

The evaluator shall verify that the TSS describes the key agreement parameters of the server's Key Exchange message.

Guidance

The evaluator shall verify that any configuration guidance necessary to meet the requirement must be contained in the AGD guidance.

Tests

The evaluator shall conduct the following tests. The testing can be carried out manually with a packet analyzer or with an automated framework that similarly captures such empirical evidence. Note that this testing can be accomplished in conjunction with other testing activities. For each of the following tests, determining that the size matches the expected size is sufficient.

- Test 1: [conditional] If RSA-based key establishment is selected, the evaluator shall attempt a connection using RSA-based key establishment with a supported size. The evaluator shall verify that the size used matches that which is configured.
- The evaluator shall repeat this test for each supported size of RSA-based key establishment.
- Test 2: [conditional] If finite-field (i.e. non-EC) Diffie-Hellman ciphers are selected, the evaluator shall attempt a connection using a Diffie-Hellman key exchange with a supported parameter size or supported group. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported parameter size or group.
- Test 3: [conditional] If ECDHE ciphers are selected, the evaluator shall attempt a connection using an ECDHE ciphersuite with a supported curve. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported elliptic curve.

5.1.2 CLASS FDP: USER DATA PROTECTION

This class contains families specifying requirements related to protecting user data. FDP is split into four groups of families (listed below) that address user data within a TOE, during import, export, and storage as well as security attributes directly related to user data.

The families in this class are organised into four groups:

- User data protection security function policies
- Forms of user data protection
- Off-line storage, import and export
- Inter-TSF communication

5.1.2.1 ACCESS TO PLATFORM RESOURCES (FDP_DEC_EXT)

Family behavior

Family created to protect access to the platform's resources.

Component levelling

Management: FDP_DEC_EXT.1

There are no management activities foreseen.

Audit: FDP_DEC_EXT.1

There are no auditable events foreseen.

FDP_DEC_EXT.1: Access to Platform Resources

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FDP_DEC_EXT.1.1: ***The TSF shall restrict its access to [selection: no hardware resources, network connectivity, camera, microphone, location services, NFC, USB, Bluetooth, [assignment: list of additional hardware resources]]***

Evaluation Activity

TSS
None.

Guidance

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.

Tests

The evaluator shall check the WMAppManifest.xml file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as ID_CAP_ISV_CAMERA, ID_CAP_LOCATION, ID_CAP_NETWORKING, ID_CAP_MICROPHONE, ID_CAP_PROXIMITY and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of the required hardware resources.

FDP_DEC_EXT.1.2: ***The TSF shall restrict its access to [selection: no sensitive information repositories, address book, calendar, call lists, system logs, [assignment: list of additional sensitive information repositories]]***

Evaluation Activity

TSS

None.

Guidance

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information repository which it accesses, identify the justification as to why access is required.

Tests

The evaluator shall check the WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS, ID_CAP_APPOINTMENTS, ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

The evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.

5.1.2.2 NETWORK COMMUNICATIONS (FDP_NET_EXT)

Family behavior

Components in this family address restrictions to network communications.

Component levelling

FDP_NET_EXT: Network Communications
1

Management: FDP_NET_EXT.1

There are no management activities foreseen.

Audit: FDP_NET_EXT.1

There are no auditable events foreseen.

FDP_NET_EXT.1: Network Communications

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FDP_NET_EXT.1.1: ***The TSF shall restrict network communication to [selection: no network communication, user-initiated communication for [assignment: list of functions for which the user can initiate network communication], respond to [assignment: list of remotely initiated communication], [assignment: list of application-initiated network communication]]***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.
- Test 2: The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).

5.1.2.3 ENCRYPTION OF SENSITIVE APPLICATION DATA (FDP_DAR_EXT)

Family behavior

Family created to describe the functional requirements for data at rest protection purposes. .

Component levelling

FDP_DAR_EXT: Encryption Of Sensitive Application Data
1

Management: FDP_DAR_EXT.1

There are no management activities foreseen.

Audit: FDP_DAR_EXT.1

There are no auditable events foreseen.

FDP_DAR_EXT.1: Encryption Of Sensitive Application Data

Hierarchical to:

No other components.

Dependencies:

See Application Note

FDP_DAR_EXT.1.1: ***The TSF shall [selection: leverage platform-provided functionality to encrypt sensitive data, implement functionality to encrypt sensitive data as defined in the EP for File Encryption, protect sensitive data in accordance with FCS_STO_EXT.1, not store any sensitive dat a] in non-volatile memory***

Application Note

If the selection "protect sensitive data in accordance with FCS_STO_EXT.1" is chosen, there will be a dependency with FCS_STO_EXT.1.

Evaluation Activity

TSS

The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application. The evaluator shall then ensure that the following activities cover all of the sensitive data identified in the TSS.

If not store any sensitive data is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory.

The evaluator shall also ensure that this is consistent with the filesystem test below.

Guidance

None.

Tests

Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1.

The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted. If leverage platform-provided functionality is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis.

The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User

Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.

5.1.3 CLASS FMT: SECURITY MANAGEMENT

This class is intended to specify the management of several aspects of the TSF: security attributes, TSF data and functions. The different management roles and their interaction, such as separation of capability, can be specified.

This class has several objectives:

- management of TSF data, which include, for example, banners;
- management of security attributes, which include, for example, the Access Control Lists, and Capability Lists;
- management of functions of the TSF, which includes, for example, the selection of functions, and rules or conditions influencing the behaviour of the TSF;
- definition of security roles.

5.1.3.1 SUPPORTED CONFIGURATION MECHANISM (FMT_MEC_EXT)

Family behavior

Family created to support the configuration mechanisms.

Component levelling

FMT_MEC_EXT: Supported Configuration Mechanism
1

Management: FMT_MEC_EXT.1

There are no management activities foreseen.

Audit: FMT_MEC_EXT.1

There are no auditable events foreseen.

FMT_MEC_EXT.1: Supported Configuration Mechanism

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FMT_MEC_EXT.1.1: ***The TSF shall invoke the mechanisms recommended by the platform vendor for storing and setting configuration options.***

Evaluation Activity

TSS

The evaluator shall review the TSS to identify the application's configuration options (e.g., settings) and determine whether these are stored and set using the mechanisms supported by the platform. At a minimum the TSS shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an SFR.

Guidance

None.

Tests

The method of testing varies per platform.

The evaluator shall determine the Windows.UI.ApplicationSettings namespace or the IsolatedStorageSettings namespace for storing application specific settings. The evaluator shall run the application while monitoring it with the SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify that ProcMon logs show corresponding changes to the the Windows Registry or C:\ProgramData\ directory.

5.1.3.2 SECURE BY DEFAULT CONFIGURATION (FMT_CFG_EXT)

Family behavior

Components in this family address requirements for secure default configuration.

Component levelling

FMT_CFG_EXT: Secure by Default Configuration
1

Management: FMT_CFG_EXT.1

There are no management activities foreseen.

Audit: FMT_CFG_EXT.1

There are no auditable events foreseen.

FMT_CFG_EXT.1: Secure by Default Configuration

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FMT_CFG_EXT.1.1: ***The TSF shall be configured by default with file permissions which protect the application binaries and data files from modification by normal unprivileged users.***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

The evaluator shall run the SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. The evaluator shall consider the requirement met because of the AppContainer sandbox.

5.1.4 CLASS FTP: TRUSTED PATH/CHANNELS

Families in this class provide requirements for a trusted communication path between users and the TSF, and for a trusted communication channel between the TSF and other trusted IT products. Trusted paths and channels have the following general characteristics:

- The communications path is constructed using internal and external communications channels (as appropriate for the component) that isolate an identified subset of TSF data and commands from the remainder of the TSF and user data.
- Use of the communications path may be initiated by the user and/or the TSF (as appropriate for the component).
- The communications path is capable of providing assurance that the user is communicating with the correct TSF, and that the TSF is communicating with the correct user (as appropriate for the component).

In this paradigm, a trusted channel is a communication channel that may be initiated by either side of the channel, and provides non-repudiation characteristics with respect to the identity of the sides of the channel.

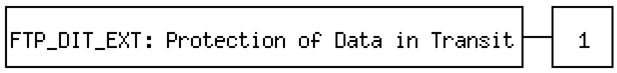
A trusted path provides a means for users to perform functions through an assured direct interaction with the TSF. Trusted path is usually desired for user actions such as initial identification and/or authentication, but may also be desired at other times during a user's session. Trusted path exchanges may be initiated by a user or the TSF. User responses via the trusted path are guaranteed to be protected from modification by or disclosure to untrusted applications.

5.1.4.1 PROTECTION OF DATA IN TRANSIT (FTP_DIT_EXT)

Family behavior

Components in this family address requirements to ensure the TOE either doesn't transmit data or if it does transmit sensitive data such data is transmitted in a secure tunnel.

Component levelling



Management: FTP_DIT_EXT.1

There are no management activities foreseen.

Audit: FTP_DIT_EXT.1

There are no audit activities foreseen.

FTP_DIT_EXT.1: Protection of Data in Transit

Hierarchical to:

No other components.

Dependencies:

See Application Note.

FTP_DIT_EXT.1.1: ***The TSF shall [selection: not transmit any [selection: data, sensitive data], encrypt all transmitted [selection: sensitive data, data] with [selection: HTTPS in accordance with FCS_HTTPS_EXT.1, TLS as defined in FCS_TLS_EXT.1 and FCS_TLSS_EXT.1], invoke platform-provided functionality to encrypt all transmitted sensitive data with [selection: HTTPS, TLS], invoke platform-provided functionality to encrypt all transmitted data with [selection: HTTPS, TLS]] between itself and another trusted IT product.***

Application Note

If the selection "encrypt all transmitted [selection: sensitive data, data] with [selection: HTTPS in accordance with FCS_HTTPS_EXT.1, TLS as defined in FCS_TLS_EXT.1 and FCS_TLSS_EXT.1]" is chosen, there will be a dependency with [FCS_HTTPS_EXT.1 or [FCS_TLS_EXT.1 and FCS_TLSS_EXT.1]].

Evaluation Activity

TSS

For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.

Guidance

None.

Tests

The evaluator shall perform the following tests.

- Test 1: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS or TLS in accordance with the selection in the ST.
- Test 2: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the

application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.

- Test 3: The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

5.1.5 CLASS FPR: PRIVACY

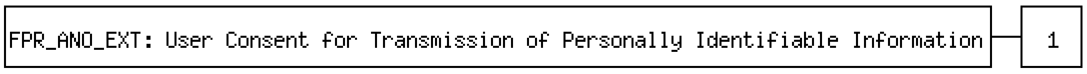
This class contains privacy requirements. These requirements provide a user protection against discovery and misuse of identity by other users.

5.1.5.1 USER CONSENT FOR TRANSMISSION OF PERSONALLY IDENTIFIABLE INFORMATION (FPR_ANO_EXT)

Family behavior

Family created to protect the transmission of personal identifiable information.

Component levelling


FPR_ANO_EXT: User Consent for Transmission of Personally Identifiable Information
1

Management: FPR_ANO_EXT.1

There are no management activities foreseen.

Audit: FPR_ANO_EXT.1

There are no auditable events foreseen.

FPR_ANO_EXT.1: User Consent for Transmission of Personally Identifiable Information

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FPR_ANO_EXT.1.1: ***The TSF shall [selection: not transmit PII over a network, require user approval before executing [assignment: list of functions that transmit PII over a network]]***

Evaluation Activity

TSS

The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted.

Guidance

None.

Tests

If require user approval before executing is selected, the evaluator shall run the application and exercise the functionality responsibly for transmitting PII and verify that user approval is required before transmission of the PII.

5.1.6 CLASS FPT: PROTECTION OF THE TSF

This class contains families of functional requirements that relate to the integrity and management of the mechanisms that constitute the TSF and to the integrity of TSF data. In some sense, families in this class may appear to duplicate components in the FDP class; they may even be implemented using the same mechanisms. However, FDP focuses on user data protection, while FPT focuses on TSF data protection. In fact, components from the FPT class are necessary to provide requirements that the SFPs In the TOE cannot be tampered with or bypassed.

From the point of view of this class, regarding to the TSF there are three significant elements:

- The TSF's implementation, which executes and implements the mechanisms that enforce the SFRs.
- The TSF's data, which are the administrative databases that guide the enforcement of the SFRs.
- The external entities that the TSF may interact with in order to enforce the SFRs.

5.1.6.1 USE OF SUPPORTED SERVICES AND APIS (FPT_API_EXT)

Family behavior

Components in this family address requirements to ensure the TOE uses platform services and APIs that are supported by the platform vendor.

Component levelling

FPT_API_EXT: Use of Supported Services and APIs
1

Management: FPT_API_EXT.1

There are no management activities foreseen.

Audit: FPT_API_EXT.1

There are no auditable events foreseen.

FPT_API_EXT.1: Use of Supported Services and APIs

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FPT_API_EXT.1.1: ***The TSF shall use only documented platform APIs***

Evaluation Activity

TSS

The evaluator shall verify that the TSS lists the platformAPIs used in the application.

Guidance

None.

Tests

The evaluator shall then compare the list with the supported APIs (available through e.g. developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

5.1.6.2 ANTI-EXPLOITATION CAPABILITIES (FPT_AEX_EXT)

Family behavior

Components in this family address requirements to ensure the TOE is not susceptible to commonly used exploitation methods. Additionally, it ensures that the application doesn’t circumvent security functionality provided by the platform.

Component levelling

Management: FPT_AEX_EXT.1

There are no management activities foreseen.

Audit: FPT_AEX_EXT.1

There are no auditable events foreseen.

FPT_AEX_EXT.1: Anti-Exploitation Capabilities

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FPT_AEX_EXT.1.1: ***The TSF shall not request to map memory at an explicit address for [assignment: list of executables performing ASLR protection]***

Evaluation Activity

TSS

The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

Guidance

None.

Tests

The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform.

The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

FPT_AEX_EXT.1.2: ***The TSF shall [selection: not allocate any memory region with both write and execute permissions for only [assignment: list of functions performing just-in-time compilation], not allocate any memory region with both write and execute permissions, not allocate any memory region with both write and execute permissions for [assignment: list of executables performing DEP protection]]***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the /NXCOMPAT flag was used during compilation to verify that DEP protections are enabled for the application.

FPT_AEX_EXT.1.3: ***The TSF shall be compatible with security features provided by the platform vendor.***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-s/windows/security/threatprotection/windows-defender-exploit-guard/customize-exploit-protection. If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

FPT_AEX_EXT.1.4: ***The TSF shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:

The shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all usermodifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files and no data files in the application's install directory.

FPT_AEX_EXT.1.5: ***The TSF shall be built with stack-based buffer overflow protection enabled for the following executables: [assignment: list of executables for which stack-based buffer overflow protection is enabled]***

Evaluation Activity

The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS and verify that the /GS flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of /GS.

5.1.6.3 SOFTWARE IDENTIFICATION AND VERSIONS (FPT_IDV_EXT)

Family behavior

Family created to address software identification and versions.

Component levelling

FPT_IDV_EXT: Software Identification and Versions
1

Management: FPT_IDV_EXT.1

There are not management activities foreseen.

Audit: FPT_IDV_EXT.1

There are not audit activities foreseen.

FPT_IDV_EXT.1: Software Identification and Versions

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FPT_IDV_EXT.1.1: ***The application shall be versioned with [selection: SWID tags that comply with minimum requirements from ISO/IEC 19770-2:2015, [assignment: other version information]]***

Evaluation Activity

TSS

If "other version information" is selected the evaluator shall verify that the TSS contains an explaination of the versioning methodology.

Guidance

None.

Tests

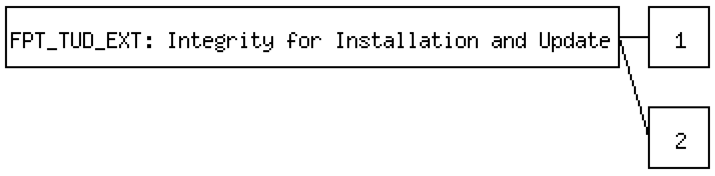
The evaluator shall install the application, then check for the / existence of version information. If SWID tags is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

5.1.6.4 INTEGRITY FOR INSTALLATION AND UPDATE (FPT_TUD_EXT)

Family behavior

Components in this family address the requirements for updating the TOE software.

Component levelling



Management: FPT_TUD_EXT.1

There are not management activities foreseen.

Management: FPT_TUD_EXT.2

There are no management activities foreseen.

Audit: FPT_TUD_EXT.1

There are not audit activities foreseen.

Audit: FPT_TUD_EXT.2

There are no auditable events foreseen.

FPT_TUD_EXT.1: Integrity for Installation and Update

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FPT_TUD_EXT.1.1: ***The TSF shall [selection: provide the ability, leverage the platform] to query the current version of the application software***

Evaluation Activity

TSS

None.

Guidance

The evaluator shall verify guidance includes a description of how to query the current version of the application.

Tests

The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

FPT_TUD_EXT.1.2: ***The application shall not download, modify, replace or update its own binary code.***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall verify that the application's executable files are not changed by the application. The evaluator shall complete the following test:

- Test 1: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

FPT_TUD_EXT.1.3: ***Application updates shall be digitally signed such that the application platform can cryptographically verify them prior to installation.***

Evaluation Activity

TSS

The evaluator shall verify that the TSS identifies how the application installation package and updates to it are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.

Guidance

None.

Tests

None.

FPT_TUD_EXT.1.4: ***The application is distributed [selection: with the platform OS, as an additional software package to the platform OS]***

Evaluation Activity

TSS

The evaluator shall verify that the TSS identifies how the application is distributed. If "with the platform" is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If "as an additional package" is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

Guidance

None.

Tests

None.

FPT_TUD_EXT.2: Integrity for Installation and Update

Hierarchical to:

No other components.

Dependencies:

FDP_TUD_EXT.1.

FPT_TUD_EXT.2.1: ***The application shall be distributed using the format of the platform-supported package manager.***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:

The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the package (.APPX) format. See https://msdn.microsoft.com/enus/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.

FPT_TUD_EXT.2.2: ***The application shall implement the mechanisms 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.***

Evaluation Activity

TSS

None.

Guidance

None.

Tests

The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.

5.2 EXTENDED ASSURANCE COMPONENTS

The extended assurance components have been extracted from [PP_APP].

5.2.1 CLASS AGD: GUIDANCE DOCUMENTS

The guidance documents class provides the requirements for guidance documentation for all user roles. For the secure preparation and operation of the TOE it is necessary to describe all relevant aspects for the secure handling of the TOE. The class also addresses the possibility of unintended incorrect configuration or handling of the TOE.

In many cases it may be appropriate that guidance is provided in separate documents for preparation and operation of the TOE, or even separate for different user roles as end-users, administrators, application programmers using software or hardware interfaces, etc.

The guidance documents class is subdivided into two families which are concerned with the preparative user guidance (what has to be done to transform the delivered TOE into its evaluated configuration in the operational environment as described in the ST) and with the operational user guidance (what has to be done during the operation of the TOE in its evaluated configuration).

5.2.1.1 AGD_OPE.1 OPERATIONAL USER GUIDANCE (AGD_OPE.1)

Objectives

Operational user guidance refers to written material that is intended to be used by all types of users of the TOE in its evaluated configuration: end-users, persons responsible for maintaining and administering the TOE in a correct manner for maximum security, and by others (e.g. programmers) using the TOE's external interfaces. Operational user guidance describes the security functionality provided by the TSF, provides instructions and guidelines (including warnings), helps to understand the TSF and includes the security-critical information, and the security-critical actions required, for its secure use. Misleading and unreasonable guidance should be absent from the guidance documentation, and secure procedures for all modes of operation should be addressed. Insecure states should be easy to detect.

The operational user guidance provides a measure of confidence that non-malicious users, administrators, application providers and others exercising the external interfaces of the TOE will understand the secure operation of the TOE and will use it as intended. The evaluation of the user guidance includes investigating whether the TOE can be used in a manner that is insecure but that the user of the TOE would reasonably believe to be secure. The objective is to minimise the risk of human or other errors in operation that may deactivate, disable, or fail to activate security functionality, resulting in an undetected insecure state.

Component levelling

This family contains only one component.

Application notes

The operational user guidance does not have to be contained in a single document. Guidance to users, administrators and application developers can be spread among documents or web pages. Where appropriate, the guidance documentation is expressed in the eXtensible Configuration Checklist Description Format (XCCDF) to support security automation. Rather than repeat information here, the developer should review the evaluation activities for this component to ascertain the specifics of the guidance that the evaluator will be checking for. This will provide the necessary information for the preparation of acceptable guidance.

AGD_OPE.1-PP13-GUIDANCE.1 Operational User Guidance

Dependencies:

ADV_FSP.1 Basic functional specification

Developer action elements:

AGD_OPE.1-PP13-GUIDANCE.1.1D

The developer shall provide operational user guidance.

Content and presentation elements:

AGD_OPE.1-PP13-GUIDANCE.1.1C

Some of the contents of the operational guidance will be verified by the evaluation activities in 6.1 Security Functional Requirements and evaluation of the TOE according to the [CEM31R5]. The following additional information is also required.

If cryptographic functions are provided by the TOE, the operational guidance shall contain instructions for configuring the cryptographic engine associated with the evaluated configuration of the TOE. It shall provide a warning to the administrator that use of other cryptographic engines was not evaluated nor tested during the CC evaluation of the TOE.

The documentation must describe the process for verifying updates to the TOE by verifying a digital signature – this may be done by the TOE or the underlying platform.

The evaluator shall verify that this process includes the following steps:

Instructions for obtaining the update itself. This should include instructions for making the update accessible to the TOE (e.g., placement in a specific directory).

Instructions for initiating the update process, as well as discerning whether the process was successful or unsuccessful. This includes generation of the digital signature. The TOE will likely contain security functionality that does not fall in the scope of evaluation under this PP. The operational guidance shall make it clear to an administrator which security functionality is covered by the evaluation activities.

Evaluator action elements:

AGD_OPE.1-PP13-GUIDANCE.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.1.2 AGD_PRE.1 PREPARATIVE PROCEDURES (AGD_PRE.1)

Objectives

Preparative procedures are useful for ensuring that the TOE has been received and installed in a secure manner as intended by the developer. The requirements for preparation call for a secure transition from the delivered TOE to its initial operational environment. This includes investigating whether the TOE can be configured or installed in a manner that is insecure but that the user of the TOE would reasonably believe to be secure.

Component levelling

This family contains only one component.

Application notes

None.

AGD_PRE.1-PP13-GUIDANCE.1 Preparative Procedures

Dependencies:

No dependencies

Developer action elements:

AGD_PRE.1-PP13-GUIDANCE.1.1D

The developer shall provide the TOE, including its preparative procedures.

Content and presentation elements:

AGD_PRE.1-PP13-GUIDANCE.1.1C

The evaluator shall check to ensure that the guidance provided for the TOE adequately addresses all platforms claimed for the TOE in the ST.

Evaluator action elements:

AGD_PRE.1-PP13-GUIDANCE.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2 CLASS ALC: LIFE CYCLE SUPPORT

Life-cycle support is an aspect of establishing discipline and control in the processes of refinement of the TOE during its development and maintenance. Confidence in the correspondence between the TOE security requirements and the TOE is greater if security analysis and the production of the evidence are done on a regular basis as an integral part of the development and maintenance activities.

In the product life-cycle it is distinguished whether the TOE is under the responsibility of the developer or the user rather than whether it is located in the development or user environment. The point of transition is the moment where the TOE is handed over to the user. This is also the point of transition from the ALC to the AGD class.

5.2.2.1 ALC_CMC.2 USE OF A CM SYSTEM (ALC_CMC.2)

Objectives

A unique reference is required to ensure that there is no ambiguity in terms of which instance of the TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be aware of which instance of the TOE they are using.

Unique identification of the configuration items leads to a clearer understanding of the composition of the TOE, which in turn helps to determine those items which are subject to the evaluation requirements for the TOE.

The use of a CM system increases assurance that the configuration items are maintained in a controlled manner.

Component levelling

This family contains only one component.

Application notes

None

ALC_CMC.2-PP13-GUIDANCE.1 Use of a CM system

Dependencies:

ALC_CMS.1 TOE CM coverage

Developer action elements:

ALC_CMC.2-PP13-GUIDANCE.1.1D

The developer shall provide the TOE and a reference for the TOE.

Content and presentation elements:

ALC_CMC.2-PP13-GUIDANCE.1.1C

The evaluator shall check the ST to ensure that it contains an identifier (such as a product name/version number) that specifically identifies the version that meets the requirements of the ST. Further, the evaluator shall check the AGD guidance and TOE samples received for testing to ensure that the version number is consistent with that in the ST. If the vendor maintains a web site advertising the TOE, the evaluator shall examine the information on the web site to ensure that the information in the ST is sufficient to distinguish the product.

Evaluator action elements:

ALC_CMC.2-PP13-GUIDANCE.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.2.2 ALC_CMS.2 PARTS OF THE TOE CM COVERAGE (ALC_CMS.2)

Objectives

A CM system can control changes only to those items that have been placed under CM (i.e., the configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise the TOE, and the evaluation evidence required by the other SARs under CM provides assurance that they have been modified in a controlled manner with proper authorisations.

Component levelling

This family contains only one component.

Application notes

None

ALC_CMS.2-PP13-GUIDANCE.1 Parts of the TOE CM coverage

Dependencies:

No dependencies

Developer action elements:

ALC_CMS.2-PP13-GUIDANCE.1.1D

The developer shall provide a configuration list for the TOE.

Content and presentation elements:

ALC_CMS.2-PP13-GUIDANCE.1.1C

The "evaluation evidence required by the SARs" in this ST is limited to the information in the ST coupled with the guidance provided to administrators and users under the AGD requirements. By ensuring that the TOE is specifically identified and that this identification is consistent in the ST and in the AGD guidance (as done in the evaluation activity for ALC_CMC.1), the evaluator implicitly confirms the information required by this component. Life-cycle support is targeted aspects of the developer's life-cycle and instructions to providers of applications for the developer's devices, rather than an indepth examination of the TSF manufacturer's development and configuration management process. This is not meant to diminish the critical role that a developer's practices play in contributing to the overall trustworthiness of a product; rather, it's a reflection on the information to be made available for evaluation.

Evaluator action elements:

ALC_CMS.2-PP13-GUIDANCE.1.1E

The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.

5.2.3 CLASS ATE: TEST

Testing is specified for functional aspects of the system as well as aspects that take advantage of design or implementation weaknesses. The former is done through the ATE_IND family, while the latter is through the AVA_VAN family. At the assurance level specified in this ST, testing is based on advertised functionality and interfaces with dependency on the availability of design information. One of the primary outputs of the evaluation process is the test report as specified in the following requirements.

5.2.3.1 ATE_IND.2 INDEPENDENT TESTING – SAMPLE (ATE_IND.2)

Objectives

Testing is performed to confirm the functionality described in the TSS as well as the administrative (including configuration and operational) documentation provided. The focus of the testing is to confirm that the requirements specified in 6.1 Security Functional Requirements being met.

Component levelling

This family contains only one component.

Application notes

The developer must provide at least one product instance of the TOE for complete testing on at least platform regardless of equivalency.

ATE_IND.2-PP13-GUIDANCE.1 Independent testing – Sample

Dependencies:

ADV_FSP.2 Security-enforcing functional specification

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

ATE_COV.1 Evidence of coverage

ATE_FUN.1 Functional testing

Developer action elements:

ATE_IND.2-PP13-GUIDANCE.1.1D

The developer shall provide the TOE for testing

Content and presentation elements:

ATE_IND.2-PP13-GUIDANCE.1.1C

The evaluator shall prepare a test plan and report documenting the testing aspects of the system, including any application crashes during testing. The evaluator shall determine the root cause of any application crashes and include that information in the report. The test plan covers all of the testing actions contained in the [CEM31R5] and the body of this PP's evaluation activities.

While it is not necessary to have one test case per test listed in an evaluation activity, the evaluator must document in the test plan that each applicable testing requirement in the ST is covered. The test plan identifies the platforms to be tested, and for those platforms not included in the test plan but included in the ST, the test plan provides a justification for not testing the platforms. This justification must address the differences between the tested platforms and the untested platforms, and make an argument that the differences do not affect the testing to be performed. It is not sufficient to merely assert that the differences have no effect; rationale must be provided. If all platforms claimed in the ST are tested, then no rationale is necessary. The test plan describes the composition of each platform to be tested, and any setup that is necessary beyond what is contained in the AGD documentation. It should be noted that the evaluator is expected to follow the AGD documentation for installation and setup of each platform either as part of a test or as a standard pre-test condition. This may include special test drivers or tools. For each driver or tool, an argument (not just an assertion) should be provided that the driver or tool will not adversely affect the performance of the

functionality by the TOE and its platform. This also includes the configuration of the cryptographic engine to be used. The cryptographic algorithms implemented by this engine are those specified by this ST and used by the cryptographic protocols being evaluated (e.g SSH). The test plan identifies high-level test objectives as well as the test procedures to be followed to achieve those objectives. These procedures include expected results.

The test report (which could just be an annotated version of the test plan) details the activities that took place when the test procedures were executed, and includes the actual results of the tests. This shall be a cumulative account, so if there was a test run that resulted in a failure; a fix installed; and then a successful re-run of the test, the report would show a "fail" and "pass" result (and the supporting details), and not just the "pass" result.

Evaluator action elements:

ATE_IND.2-PP13-GUIDANCE.1.1E

The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5.2.4 CLASS AVA: VULNERABILITY ASSESSMENT

For the current generation of this ST, the evaluation lab is expected to survey open sources to discover what vulnerabilities have been discovered in these types of products. In most cases, these vulnerabilities will require sophistication beyond that of a basic attacker. Until penetration tools are created and uniformly distributed to the evaluation labs, the evaluator will not be expected to test for these vulnerabilities in the TOE. The labs will be expected to comment on the likelihood of these vulnerabilities given the documentation provided by the vendor.

5.2.4.1 AVA_VAN.2 VULNERABILITY ANALYSIS (AVA_VAN.2)

Objectives

A vulnerability analysis is performed by the evaluator to ascertain the presence of potential vulnerabilities.

Component levelling

This family contains only one component.

Application notes

Suitability for testing means not being obfuscated or packaged in such a way as to disrupt either static or dynamic analysis by the evaluator.

AVA_VAN.2-PP13-GUIDANCE.1 Vulnerability Analysis

Dependencies:

ADV_ARC.1 Security architecture description

ADV_FSP.2 Security-enforcing functional specification

ADV_TDS.1 Basic design

AGD_OPE.1 Operational user guidance

AGD_PRE.1 Preparative procedures

Developer action elements:

AVA_VAN.2-PP13-GUIDANCE.1.1D

The developer shall provide the TOE for testing.

Content and presentation elements:

AVA_VAN.2-PP13-GUIDANCE.1.1C

The evaluator shall generate a report to document their findings with respect to this requirement. This report could physically be part of the overall test report mentioned in ATE_IND, or a separate document. The evaluator performs a search of public information to find vulnerabilities that have been found in similar applications with a particular focus on network protocols the application uses and document formats it parses. The evaluator shall also run a virus scanner with the most current virus definitions against the application files and verify that no files are flagged as malicious.

The evaluator documents the sources consulted and the vulnerabilities found in the report.

For each vulnerability found, the evaluator either provides a rationale with respect to its non-applicability, or the evaluator formulates a test (using the guidelines provided in ATE_IND) to confirm the vulnerability, if suitable. Suitability is determined by assessing the attack vector needed to take advantage of the vulnerability. If exploiting the vulnerability requires expert skills and an electron microscope, for instance, then a test would not be suitable and an appropriate justification would be formulated.

Evaluator action elements:

AVA_VAN.2-PP13-GUIDANCE.1.1E

The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack potential.

6 SECURITY REQUIREMENTS

This section defines the Security functional requirements (SFRs) and the Security assurance requirements (SARs) that fulfill the TOE. Assignment, selection, iteration and refinement operations have been made, adhering to the following conventions:

- Assignments. They appear between square brackets. The word “assignment” is maintained and the resolution is presented in ***boldface, italic and blue color.***
- Selections. They appear between square brackets. The word “selection” is maintained and the resolution is presented in ***boldface, italic and blue color.***
- Iterations. It includes “/” and an “identifier” following requirement identifier that allows to distinguish the iterations of the requirement. Example: FCS_COP.1/XXX.
- Refinements: the text where the refinement has been done is shown ***bold, italic, and light red color.*** Where part of the content of a SFR component has been removed, the removed text is shown in ~~***bold, italic, light red color and crossed out.***~~

6.1 SECURITY FUNCTIONAL REQUIREMENTS

6.1.1 FCS: CRYPTOGRAPHIC SUPPORT

6.1.1.1 FCS_CKM_EXT.2: PASWORD CONDITIONING

FCS_CKM_EXT.2.1 A password/passphrase shall perform Password-based Key Derivation Functions in accordance with a specified cryptographic algorithm as specified in ***[assignment: FCS_COP.1/(4)]***, with ***[assignment: 1000 iterations]***, and output cryptographic key sizes ***[assignment: 256]*** that meet ***[assignment: NIST SP 800-132]***.

FCS_CKM_EXT.2.2 The TSF shall generate salts using a RBG that meets FCS_RBG_EXT.1 and with entropy corresponding to the security strength selected for PBKDF in FCS_CKM_EXT.2.

6.1.1.2 FCS_CKM.1: CRYPTOGRAPHIC ASYMMETRIC KEY GENERATION

FCS_CKM.1.1 The TSF shall ***implement functionality to*** generate ***asymmetric*** cryptographic keys in accordance with a specified cryptographic key generation algorithm ***[assignment: ECC schemes using“"NIST curve”" P-256, P-384 and no other curves]*** and specified cryptographic key sizes ***[assignment: none]*** that meet the following: ***[assignment: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Appendix B.4]*** .

Application Note

This SFR refers to the asymmetric ephemeral keys generated between the TOE and endpoints in TLSv1.2-based communications.

6.1.1.3 FCS_CKM.2: CRYPTOGRAPHIC KEY ESTABLISHMENT

FCS_CKM.2.1 The TSF shall ***~~distribute cryptographic keys~~ implement functionality to perform cryptographic key establishment*** in accordance with a specified cryptographic key ***~~distribution~~ establishment*** method ***[assignment: Elliptic curve-based key establishment schemes]*** that meets the following: ***[assignment: NIST Special Publication 800-56A, “"Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography”"]*** .

6.1.1.4 FCS_COP.1/(4): CRYPTOGRAPHIC OPERATION–- PASSWORDS

FCS_COP.1.1/(4) The TSF shall perform ***[assignment: keyed-hash message]*** in accordance with a specified cryptographic algorithm ***[assignment: HMAC-SHA-256]*** and cryptographic key sizes ***[assignment: 256]*** that meet the following: ***[assignment: FIPS Pub 198-1 The Keyed-Hash Message Authentication Code and FIPS Pub 180-4 Secure Hash Standard]*** .

6.1.1.5 FCS_COP.1/(1): CRYPTOGRAPHIC OPERATION–- ENCRYPTION/DECRYPTION

FCS_COP.1.1/(1) The TSF shall perform ***[assignment: encryption/decryption]*** in accordance with a specified cryptographic algorithm ***[assignment: AES-GCM (as defined in NIST SP 800-38D) mode]*** and cryptographic key sizes ***[assignment: 256-bit]*** that meet the following: ***[assignment: none]*** .

6.1.1.6 FCS_COP.1/(3): CRYPTOGRAPHIC OPERATION–- SIGNING

FCS_COP.1.1/(3) The TSF shall perform ***[assignment: cryptographic signature services (generation and verification)]*** in accordance with a specified cryptographic algorithm ***[assignment: RSA schemes]*** and cryptographic key sizes ***[assignment: of 3072-bit]*** that meet the following: ***[assignment: FIPS PUB 186-4, “Digital Signature Standard (DSS)”, Section 4]***

6.1.1.7 FCS_COP.1/(2): CRYPTOGRAPHIC OPERATION–- HASHING

FCS_COP.1.1/(2) The TSF shall perform ***[assignment: cryptographic hashing services]*** in accordance with a specified cryptographic algorithm ***[assignment: SHA-384]*** and ***~~cryptographic key~~ message digest*** sizes ***[assignment: 384]*** that meet the following: ***[assignment: FIPS Pub 180-4]*** .

6.1.1.8 FCS_RBG_EXT.1: RANDOM BIT GENERATION

FCS_RBG_EXT.1.1 The TSF shall ***[selection: implement DRBG functionality]*** for its cryptographic operations

6.1.1.9 FCS_RBG_EXT.2: RANDOM BIT GENERATION FROM APPLICATION

FCS_RBG_EXT.2.1 The TSF shall perform all deterministic random bit generation (DRBG) services in accordance with NIST Special Publication 800-90A using ***[selection: 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 ***[selection: a software-based noise source]*** with a minimum of ***[selection: 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.

6.1.1.10FCS_CKM_EXT.1: CRYPTOGRAPHIC KEY GENERATION SERVICES

FCS_CKM_EXT.1.1 The TSF shall ***[selection: implement asymmetric key generation]***

Application Note

There are TLS-based secure communication channels which need asymmetric cryptographic keys. There are two different asymmetric cryptographic keys, first are generated following the security guidelines and second ones are generated by the TOE, based on the previously generated keys, for TLS session. This SFR refers to the ephemeral asymmetric keys generated in for the TLS session.

6.1.1.11FCS_STO_EXT.1: STORAGE OF CREDENTIALS

FCS_STO_EXT.1.1 The TSF shall ***[selection: implement functionality to securely store [assignment: user's passwords and SNMPv3 credentials] according to [selection: FCS_CKM_EXT.2]]*** to non-volatile memory

6.1.1.12FCS_HTTPS_EXT.1: HTTPS PROTOCOL

FCS_HTTPS_EXT.1.1 The TSF shall implement the HTTPS protocol that complies with RFC 2818.

Application Note

The application acts as an HTTPS server, so it does not have to check the client's certificate

FCS_HTTPS_EXT.1.2 The TSF shall implement HTTPS using TLS as defined in FCS_TLS_EXT.1 and FCS_TLSS_EXT.1.

6.1.1.13FCS_TLS_EXT.1: TLS PROTOCOL

FCS_TLS_EXT.1.1 The TSF shall implement ***[selection: TLS as a server]***

6.1.1.14FCS_TLSS_EXT.1: TLS SERVER PROTOCOL

FCS_TLSS_EXT.1.1 The TSF shall implement TLS 1.2 (RFC 5246) and ***[selection: no earlier TLS versions]*** as a server that supports the ciphersuites ***[selection: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 as defined in RFC 5289]*** and no other ciphersuites, and also supports functionality for ***[selection: none]***

Application Note

The TLS channel described by this SFR is used both in the communication implemented by the MQTTv3.1.1 device and in the HTTPS channel with the remote administrator or user.

FCS_TLSS_EXT.1.2 The TSF shall deny connections from clients requesting SSL 2.0, SSL 3.0, TLS 1.0 and ***[selection: TLS 1.1]***

FCS_TLSS_EXT.1.3 The TSF shall perform key establishment for TLS using ***[selection: ECDHE parameters using elliptic curves [selection: secp256r1, secp384r1] and no other curves]***

6.1.2 FMT: SECURITY MANAGEMENT

6.1.2.1 FMT_SMF.1: SPECIFICATION OF MANAGEMENT FUNCTIONS

FMT_SMF.1.1 The TSF shall be capable of performing the following management functions:
[assignment: - Ability to manage artifacts
- Ability to manage communications channels configuration
- Ability to perform user management].

6.1.2.2 FMT_MEC_EXT.1: SUPPORTED CONFIGURATION MECHANISM

FMT_MEC_EXT.1.1 The TSF shall invoke the mechanisms recommended by the platform vendor for storing and setting configuration options.

6.1.2.3 FMT_CFG_EXT.1: SECURE BY DEFAULT CONFIGURATION

FMT_CFG_EXT.1.1 The TSF shall be configured by default with file permissions which protect the application binaries and data files from modification by normal unprivileged users.

6.1.3 FTP: TRUSTED PATH/CHANNELS

6.1.3.1 FTP_ITC.1/SNMPV3: INTER-TSF TRUSTED CHANNEL

FTP_ITC.1.1/SNMPV3 The TSF shall provide a communication channel between itself and another trusted IT product that is logically distinct from other communication channels and provides assured identification of its end points and protection of the channel data from modification or disclosure.

FTP_ITC.1.2/SNMPV3 The TSF shall permit ***[selection: the TSF]*** to initiate communication via the trusted channel.

FTP_ITC.1.3/SNMPV3 The TSF shall initiate communication via the trusted channel for ***[assignment: communicating with external IT entities through SNMPv3's-based artifact]*** .

6.1.3.2 FTP_DIT_EXT.1: PROTECTION OF DATA IN TRANSIT

FTP_DIT_EXT.1.1 The TSF shall ***[selection: encrypt all transmitted [selection: data] with [selection: TLS as defined in FCS_TLS_EXT.1 and FCS_TLSS_EXT.1]*** between itself and another trusted IT product.

Application Note

The security of the MQTTv3.1.1 artifact channel connecting external IT entities to the TOE is based on TLS, in accordance with FCS_TLSS_EXT.

Application Note

DCP and ICMP are protocols used by artifacts for the discovery of external IT entities and therefore do not use any protection mechanism for this purpose. Once these entities are discovered, the MQTTv3.1.1 and SNMPv3 artifacts are responsible for secure communication.

6.1.4 FDP: USER DATA PROTECTION

6.1.4.1 FDP_DEC_EXT.1: ACCESS TO PLATFORM RESOURCES

FDP_DEC_EXT.1.1 The TSF shall restrict its access to ***[selection: network connectivity]***

FDP_DEC_EXT.1.2 The TSF shall restrict its access to ***[selection: no sensitive information repositories]***

6.1.4.2 FDP_DAR_EXT.1: ENCRYPTION OF SENSITIVE APPLICATION DATA

FDP_DAR_EXT.1.1 The TSF shall ***[selection: leverage platform-provided functionality to encrypt sensitive data, protect sensitive data in accordance with FCS_STO_EXT.1]*** in non-volatile memory

Application Note

Users and SNMPv3 passwords are securely stored in non-volatile memory according to FCS_STO_EXT. In addition, all sensitive data is encrypted at full drive encryption level via BitLocker. The disk encryption process to be followed by the user is described at the security guidelines.

6.1.4.3 FDP_NET_EXT.1: NETWORK COMMUNICATIONS

FDP_NET_EXT.1.1 The TSF shall restrict network communication to ***[selection: user-initiated communication for [assignment: HTTPS secure channel between the TOE and the web browser, MQTTv3.1.1 secure channel between the TOE and external IT entities, and for the discovery of external IT entities through DCP and ICMP], [assignment: TOE-initiated communication for SNMPv3 secure channel between the TOE and external IT entities]]***

6.1.5 FPR: PRIVACY

6.1.5.1 FPR_ANO_EXT.1: USER CONSENT FOR TRANSMISSION OF PERSONALLY IDENTIFIABLE INFORMATION

FPR_ANO_EXT.1.1 The TSF shall ***[selection: not transmit PII over a network]***

6.1.6 FPT: PROTECTION OF THE TSF

6.1.6.1 FPT_API_EXT.1: USE OF SUPPORTED SERVICES AND APIS

FPT_API_EXT.1.1 The TSF shall use only documented platform APIs

6.1.6.2 FPT_AEX_EXT.1: ANTI-EXPLOITATION CAPABILITIES

FPT_AEX_EXT.1.1 The TSF shall not request to map memory at an explicit address for ***[assignment: -***
node.exe
- ***mongod.exe***
- ***netinservicecontroller.exe]***

Application Note

The TSS section of this SFR lists all .exe and .dll files contained in the installation directory. All those with the ASLR protection flag are included in this requirement.

FPT_AEX_EXT.1.2 The TSF shall ***[selection: not allocate any memory region with both write and execute permissions for [assignment: -memurai.exe***

- ***nginx.exe***
- ***compat.exe***
- ***uninstall_Netin.exe***
- ***node.exe***
- ***mongod.exe***
- ***netinservicecontroller.exe]]***

Application Note

The TSS section of this SFR lists all .exe and .dll files contained in the installation directory. All those with DEP protection are included in this requirement.

FPT_AEX_EXT.1.3 The TSF shall be compatible with security features provided by the platform vendor.

FPT_AEX_EXT.1.4 The TSF shall not write user-modifiable files to directories that contain executable files unless explicitly directed by the user to do so.

Application Note

The TOE allows the installation of its binary files in a different directory from its data files.

FPT_AEX_EXT.1.5 The TSF shall be built with stack-based buffer overflow protection enabled for the following executables: ***[assignment: - memurai.exe***

- ***compat.exe***
- ***nginx.exe***
- ***node.exe***
- ***uninstall_Netin.exe***
- ***mongod.exe***
- ***netinservicecontroller.exe]***

Application Note

The TSS section of this SFR lists all .exe and .dll files contained in the installation directory. All those with the GS protection flag are included in this requirement.

6.1.6.3 FPT_IDV_EXT.1: SOFTWARE IDENTIFICATION AND VERSIONS

FPT_IDV_EXT.1.1 The application shall be versioned with ***[selection: [assignment: SEMVER2.0]]***

6.1.6.4 FPT_TUD_EXT.1: INTEGRITY FOR INSTALLATION AND UPDATE

FPT_TUD_EXT.1.1 The TSF shall ***[selection: provide the ability]*** to query the current version of the application software

FPT_TUD_EXT.1.2 The application shall not download, modify, replace or update its own binary code.

FPT_TUD_EXT.1.3 Application updates shall be digitally signed such that the application platform can cryptographically verify them prior to installation.

FPT_TUD_EXT.1.4 The application is distributed ***[selection: as an additional software package to the platform OS]***

6.1.6.5 FPT_TUD_EXT.2: INTEGRITY FOR INSTALLATION AND UPDATE

FPT_TUD_EXT.2.1 The application shall be distributed using the format of the platform-supported package manager.

FPT_TUD_EXT.2.2 The application shall implement the mechanisms 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.

6.2 SECURITY ASSURANCE REQUIREMENTS

The development and the evaluation of the TOE shall be done in accordance to the following security assurance requirements: **EAL2**.

The evaluation lab also performs the extended assurance evaluation activities defined within section 5.2 Extended Assurance Components and within 6 Security Requirements, which are intended to be an interpretation of the other CEM assurance requirements as they apply to the specific technology instantiated in the TOE. The evaluation activities that are captured in 5.2 Extended Assurance Components and 6 Security Requirements also provide clarification as to what the developer needs to provide to demonstrate the TOE is compliant with the ST.

The following table shows the assurance requirements by reference the individual components in [CC31R5P3]

Assurance Class	Assurance Components
ASE: Security Target evaluation	ASE_CCL.1: Conformance claims ASE_ECD.1: Extended components definition ASE_INT.1: ST introduction ASE_TSS.1: TOE summary specification ASE_OBJ.2: Security objectives ASE_REQ.2: Derived security requirements ASE_SPD.1: Security problem definition

Assurance Class	Assurance Components
ALC: Life-cycle support	ALC_CMC.2: Use of a CM system ALC_CMS.2: Parts of the TOE CM coverage ALC_DEL.1: Delivery procedures
ADV: Development	ADV_TDS.1: Basic design ADV_ARC.1: Security architecture description ADV_FSP.2: Security-enforcing functional specification
AGD: Guidance documents	AGD_OPE.1: Operational user guidance AGD_PRE.1: Preparative procedures
ATE: Tests	ATE_COV.1: Evidence of coverage ATE_FUN.1: Functional testing ATE_IND.2: Independent testing–- sample
AVA: Vulnerability assessment	AVA_VAN.2: Vulnerability analysis

Table 4 Security Assurance Requirements

The following additional evaluation activities are defined for this ST:

Assurance Class	Assurance Components
AGD: Guidance documents	AGD_OPE.1-PP13-GUIDANCE.1 Operational User Guidance AGD_PRE.1-PP13-GUIDANCE.1 Preparative Procedures
ALC: Life-cycle support	ALC_CMC.2-PP13-GUIDANCE.1 Use of a CM system ALC_CMS.2-PP13-GUIDANCE.1 Parts of the TOE CM coverage
ATE: Tests	ATE_IND.2-PP13-GUIDANCE.1 Independent testing – Sample
AVA: Vulnerability assessment	AVA_VAN.2-PP13-GUIDANCE.1 Vulnerability Analysis

FCS_RBG_EXT.1 Random Bit Generation Services (FCS_RBG_EXT.1)

FCS_RBG_EXT.1.1

TSS

If use no DRBG functionality is selected, the evaluator shall inspect the application and its developer documentation and verify that the application needs no random bit generation services.

If implement DRBG functionality is selected, the evaluator shall ensure that additional FCS_RBG_EXT.2 elements are included in the ST.

If invoke platform-provided DRBG functionality is selected, the evaluator performs the following activities. The evaluator shall examine the TSS to confirm that it identifies all functions (as described by the SFRs included in the ST) that obtain random numbers from the platform RBG. The evaluator shall determine that for each of these functions, the TSS states which platform interface (API) is used to obtain the random numbers.

The evaluator shall confirm that each of these interfaces corresponds to the acceptable interfaces listed for each platform below.

It should be noted that there is no expectation that the evaluators attempt to confirm that the APIs are being used correctly for the functions identified in the TSS; the activity is to list the used APIs and then do an existence check via decompilation.

Guidance

None.

Tests

If invoke platform-provided DRBG functionality is selected, the following tests shall be performed:

The evaluator shall decompile the application binary using a decompiler suitable for the application (TOE). The evaluator shall search the output of the decompiler to determine that, for each API listed in the TSS, that API appears in the output. If the representation of the API does not correspond directly to the strings in the following list, the evaluator shall provide a mapping from the decompiled text to its corresponding API, with a description of why the API text does not directly correspond to the decompiled text and justification that the decompiled text corresponds to the associated API.

The following are the per-platform list of acceptable APIs:

The evaluator shall verify that rand_s, RtlGenRandom, BCryptGenRandom, or CryptGenRandom API is used for classic desktop applications. The evaluator shall verify the application uses the RNGCryptoServiceProvider class or derives a class from System.Security.Cryptography.RandomNumberGenerator. It is only required that the API is called/invoked, there is no requirement that the API be used directly. In future versions of this document, CryptGenRandom may be removed as an option as it is no longer the preferred API per vendor documentation.

If invocation of platform-provided functionality is achieved in another way, the evaluator shall ensure the TSS describes how this is carried out, and how it is equivalent to the methods listed here (e.g. higher-level API invokes identical low-level API).

FCS_RBG_EXT.2 Random Bit Generation from Application (FCS_RBG_EXT.2)

FCS_RBG_EXT.2.1

TSS

None.

Guidance

None.

Tests

The evaluator shall perform the following tests, depending on the standard to which the RBG conforms.

Implementations Conforming to FIPS 140-2 Annex C.

The reference for the tests contained in this section is The Random Number Generator Validation System (RNGVS). The evaluators shall conduct the following two tests. Note that the "expected values" are produced by a reference implementation of the algorithm that is known to be correct. Proof of correctness is left to each Scheme.

- Test 1: The evaluators shall perform a Variable Seed Test. The evaluators shall provide a set of 128 (Seed, DT) pairs to the TSF RBG function, each 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant for all 128 (Seed, DT) pairs. The DT value is incremented by 1 for each set. The seed values shall have no repeats within the set. The evaluators ensure that the values returned by the TSF match the expected values.
- Test 2: The evaluators shall perform a Monte Carlo Test. For this test, they supply an initial Seed and DT value to the TSF RBG function; each of these is 128 bits. The evaluators shall also provide a key (of the length appropriate to the AES algorithm) that is constant throughout the test. The evaluators then invoke the TSF RBG 10,000 times, with the DT value being incremented by 1 on each iteration, and the new seed for the subsequent iteration produced as specified in NIST Recommended Random Number Generator Based on ANSI X9.31 Appendix A.2.4 Using the 3-Key Triple DES and AES Algorithms, Section E.3. The evaluators ensure that the 10,000th value produced matches the expected value.

Implementations Conforming to NIST Special Publication 800-90A

- Test 1: The evaluator shall perform 15 trials for the RNG implementation. If the RNG is configurable, the evaluator shall perform 15 trials for each configuration. The evaluator shall also confirm that the operational guidance contains appropriate instructions for configuring the RNG functionality. If the RNG has prediction resistance enabled, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) generate a second block of random bits (4) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The next two are additional input and entropy input for the first call to generate. The final two are additional input and entropy input for the second call to generate. These values are randomly generated. "Generate one block of random bits" means to generate random bits with number of returned bits equal to the Output Block Length (as defined in NIST SP 800-90A).
  If the RNG does not have prediction resistance, each trial consists of (1) instantiate DRBG, (2) generate the first block of random bits (3) reseed, (4) generate a second block of random bits (5) uninstantiate. The evaluator verifies that the second block of random bits is the expected value. The evaluator shall generate eight input values for each trial. The first is a count (0 – 14). The next three are entropy input, nonce, and personalization string for the instantiate operation. The fifth value is additional input to the first call to generate. The sixth and seventh are additional input and entropy input to the call to reseed. The final value is additional input to the second generate call.
  The following paragraphs contain more information on some of the input values to be generated/selected by the evaluator.
  Entropy input: the length of the entropy input value must equal the seed length.
  Nonce: If a nonce is supported (CTR_DRBG with no Derivation Function does notuse a nonce), the nonce bit length is one-half the seed length.
  Personalization string: The length of the personalization string must be less than or equal to seed length. If the implementation only supports one personalization string length, then the same length can be used for both values. If more than one string length is support, the evaluator shall use personalization strings of two different lengths. If the implementation does not use a personalization string, no value needs to be supplied.
  Additional input: the additional input bit lengths have the same defaults and restrictions as the personalization string lengths.

FCS_RBG_EXT.2.2

TSS

Documentation shall be produced - and the evaluator shall perform the activities – in accordance with Appendix D of the Protection Profile - Entropy Documentation and Assessment and the Clarification to the Entropy Documentation and Assessment Annex.

Guidance

None.

Tests

In the future, specific statistical testing (in line with NIST SP 800-90B) will be required to verify the entropy estimates

FCS_CKM_EXT.1 Cryptographic Key Generation Services (FCS_CKM_EXT.1)

FCS_CKM_EXT.1.1

TSS

The evaluator shall inspect the application and its developer documentation to determine if the application needs asymmetric key generation services. If not, the evaluator shall verify the generate no asymmetric cryptographic keys selection is present in the ST. Otherwise, the evaluation activities shall be performed as stated in the selection-based requirements.

Guidance

None.

Tests

None.

FCS_CKM.1(1) Cryptographic Asymmetric Key Generation (FCS_CKM.1(1))

FCS_CKM.1.1(1)

TSS

The evaluator shall ensure that the TSS identifies the key sizes supported by the TOE. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

If the application invokes platform-provided functionality for asymmetric key generation, then the evaluator shall examine the TSS to verify that it describes how the key generation functionality is invoked.

Guidance

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key generation scheme(s) and key size(s) for all uses defined in this ST.

Tests

If the application implements asymmetric key generation, then the following test activities shall be carried out.

Evaluation Activity Note: The following tests may require the developer to provide access to a developer environment that provides the evaluator with tools that are typically available to end-users of the application.

Key Generation for FIPS PUB 186-4 RSA Schemes

The evaluator shall verify the implementation of RSA Key Generation by the TOE using the Key Generation test. This test verifies the ability of the TSF to correctly produce values for the key components including the public verification exponent e, the private prime factors p and q, the public modulus n and the calculation of the private signature exponent d. Key Pair generation specifies 5 ways (or methods) to generate the primes p and q. These include:

1. Random Primes:

- Provable primes
- Probable primes

2. Primes with Conditions:

- Primes p1, p2, q1,q2, p and q shall all be provable primes
- Primes p1, p2, q1, and q2 shall be provable primes and p and q shall be probable primes
- Primes p1, p2, q1,q2, p and q shall all be probable primes

To test the key generation method for the Random Provable primes method and for all the Primes with Conditions methods, the evaluator must seed the TSF key generation routine with sufficient data to deterministically generate the RSA key pair. This includes the random seed(s), the public exponent of the RSA key, and the desired key length. For each key length supported, the evaluator shall have the TSF generate 25 key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation.

If possible, the Random Probable primes method should also be verified against a known good implementation as described above. Otherwise, the evaluator shall have the TSF generate 10 keys pairs for each supported key length nlen and verify:

- $n = p \cdot q$,
- p and q are probably prime according to Miller-Rabin tests,
- $GCD(p-1,e) = 1$,
- $GCD(q-1,e) = 1$,
- $2^{16} \le e \le 2^{256}$ and e is an odd integer,
- $|p-q| > 2^{nlen/2} - 100$,
- $p \ge 2^{nlen/2 -1/2}$,
- $q \ge 2^{nlen/2 -1/2}$,

- $2^{(nlen/2)} < d < LCM(p-1,q-1)$,
- $e \cdot d = 1 \mod LCM(p-1,q-1)$.

Key Generation for Elliptic Curve Cryptography (ECC)

FIPS 186-4 ECC Key Generation Test For each supported NIST curve, i.e., P-256, P384 and P-521, the evaluator shall require the implementation under test (IUT) to generate 10 private/public key pairs. The private key shall be generated using an approved random bit generator (RBG). To determine correctness, the evaluator shall submit the generated key pairs to the public key verification (PKV) function of a known good implementation.

FIPS 186-4 Public Key Verification (PKV) Test For each supported NIST curve, i.e., P256, P-384 and P-521, the evaluator shall generate 10 private/public key pairs using the key generation function of a known good implementation and modify five of the public key values so that they are incorrect, leaving five values unchanged (i.e., correct). The evaluator shall obtain in response a set of 10 PASS/FAIL values.

Key Generation for Finite-Field Cryptography (FFC)

The evaluator shall verify the implementation of the Parameters Generation and the Key Generation for FFC by the TOE using the Parameter Generation and Key Generation test. This test verifies the ability of the TSF to correctly produce values for the field prime p, the cryptographic prime q (dividing p-1), the cryptographic group generator g, and the calculation of the private key x and public key y. The Parameter generation specifies 2 ways (or methods) to generate the cryptographic prime q and the field prime p:

Cryptographic and Field Primes:

- Primes q and p shall both be provable primes
- Primes q and field prime p shall both be probable primes

and two ways to generate the cryptographic group generator g:

Cryptographic Group Generator:

- Generator g constructed through a verifiable process
- Generator g constructed through an unverifiable process.

The Key generation specifies 2 ways to generate the private key x: Private Key:

- len(q) bit output of RBG where $1 \leq x \leq q-1$
- len(q) + 64 bit output of RBG, followed by a mod q-1 operation where $1 \leq x \leq q-1$.

The security strength of the RBG must be at least that of the security offered by the FFC parameter set. To test the cryptographic and field prime generation method for the provable primes method and/or the group generator g for a verifiable process, the evaluator must seed the TSF parameter generation routine with sufficient data to deterministically generate the parameter set. For each key length supported, the evaluator shall have the TSF generate 25 parameter sets and key pairs. The evaluator shall verify the correctness of the TSF's implementation by comparing values generated by the TSF with those generated from a known good implementation. Verification must also confirm

- $g \neq 0,1$
- q divides p-1
- $g^q \bmod p = 1$
- $g^x \bmod p = y$

for each FFC parameter set and key pair.

Diffie-Hellman Group 14 and FFC Schemes using "safe-prime" groups

Testing for FFC Schemes using Diffie-Hellman group 14 and/or safe-prime groups is done as part of testing in CKM.2.1.

FCS_CKM.1(3) Password Conditioning (FCS_CKM.1(3))

FCS_CKM.1.2(3)

TSS

Support for PBKDF: The evaluator shall examine the password hierarchy TSS to ensure that the formation of all password based derived keys is described and that the key sizes match that described by the ST author. The evaluator shall check that the TSS describes the method by which the password/passphrase is first encoded and then fed to the SHA algorithm. The settings for the algorithm (padding, blocking, etc.) shall be described, and the evaluator shall verify that these are supported by the selections in this component as well as the selections concerning the hash function itself. The evaluator shall verify that the TSS contains a description of how the output of the hash function is used to form the submask that will be input into the function. For the NIST SP 800-132-based conditioning of the password/passphrase, the required evaluation activities will be performed when doing the evaluation activities for the appropriate requirements (FCS_COP.1.1(4)). No explicit testing of the formation of the submask from the input password is required. FCS_CKM.1.1(3): The ST author shall provide a description in the TSS regarding the salt generation. The evaluator shall confirm that the salt is generated using an RBG described in FCS_RBG_EXT.1.

Guidance

None.

Tests

None.

FCS_CKM.2 Cryptographic Key Establishment (FCS_CKM.2)

TSS

The evaluator shall ensure that the supported key establishment schemes correspond to the key generation schemes identified in FCS_CKM.1.1. If the ST specifies more than one scheme, the evaluator shall examine the TSS to verify that it identifies the usage for each scheme.

Guidance

The evaluator shall verify that the AGD guidance instructs the administrator how to configure the TOE to use the selected key establishment scheme(s).

Tests

Evaluation Activity Note: The following tests require the developer to provide access to a test platform that provides the evaluator with tools that are typically not found on factory products.

Key Establishment Schemes

The evaluator shall verify the implementation of the key establishment schemes supported by the TOE using the applicable tests below.

SP800-56A Key Establishment Schemes

The evaluator shall verify a TOE's implementation of SP800-56A key agreement schemes using the following Function and Validity tests. These validation tests for each key agreement scheme verify that a TOE has implemented the components of the key agreement scheme according to the specifications in the Recommendation. These components include the calculation of the DLC primitives (the shared secret value Z) and the calculation of the derived keying material (DKM) via the Key Derivation Function (KDF). If key confirmation is supported, the evaluator shall also verify that the components of key confirmation have been implemented correctly, using the test procedures described below. This includes the parsing of the DKM, the generation of MACdata and the calculation of MACtag.

Function Test

The Function test verifies the ability of the TOE to implement the key agreement schemes correctly. To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each supported key agreement scheme-key agreement role combination, KDF type, and, if supported, key confirmation role- key confirmation type combination, the tester shall generate 10 sets of test vectors. The data set consists of one set of domain parameter values (FFC) or the NIST approved curve (ECC) per 10 sets of public keys. These keys are static, ephemeral or both depending on the scheme being tested.

The evaluator shall obtain the DKM, the corresponding TOE's public keys (static and/or ephemeral), the MAC tag(s), and any inputs used in the KDF, such as the Other Information (OtherInfo) and TOE id fields.

If the TOE does not use a KDF defined in SP 800-56A, the evaluator shall obtain only the public keys and the hashed value of the shared secret.

The evaluator shall verify the correctness of the TSF's implementation of a given scheme by using a known good implementation to calculate the shared secret value, derive the keying material DKM, and compare hashes or MAC tags generated from these values.

If key confirmation is supported, the TSF shall perform the above for each implemented approved MAC algorithm.

Validity Test

The Validity test verifies the ability of the TOE to recognize another party's valid and invalid key agreement results with or without key confirmation. To conduct this test, the evaluator shall obtain a list of the supporting cryptographic functions included in the SP800-56A key agreement implementation to determine which errors the TOE should be able to recognize. The evaluator generates a set of 24 (FFC) or 30 (ECC) test vectors consisting of data sets including domain parameter values or NIST approved curves, the evaluator's public keys, the TOE's public/private key pairs, MACTag, and any inputs used in the KDF, such as the OtherInfo and TOE id fields.

The evaluator shall inject an error in some of the test vectors to test that the TOE recognizes invalid key agreement results caused by the following fields being incorrect: the shared secret value Z, the DKM, the OtherInfo field, the data to be MACed, or the generated MACTag. If the TOE contains the full or partial (only ECC) public key validation, the evaluator will also individually inject errors in both parties' static public keys, both parties' ephemeral public keys and the TOE's static private key to assure the TOE detects errors in the public key validation function and/or the partial key validation function (in ECC only). At least two of the test vectors shall remain unmodified and therefore should result in valid key agreement results (they should pass).

The TOE shall use these modified test vectors to emulate the key agreement scheme using the corresponding parameters. The evaluator shall compare the TOE's results with the results using a known good implementation verifying that the TOE detects these errors.

SP800-56B Key Establishment Schemes

The evaluator shall verify that the TSS describes whether the TOE acts as a sender, a recipient, or both for RSA-based key establishment schemes.

If the TOE acts as a sender, the following evaluation activity shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with or without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTSOAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA public key, the plaintext keying material, any additional input parameters if applicable, the MacKey and MacTag if key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform a key establishment encryption operation on the TOE with the same inputs (in cases where key confirmation is incorporated, the test shall use the MacKey from the test vector instead of the randomly generated MacKey used in normal operation) and ensure that the outputted ciphertext is equivalent to the ciphertext in the test vector.

If the TOE acts as a receiver, the following evaluation activities shall be performed to ensure the proper operation of every TOE supported combination of RSA-based key establishment scheme:

To conduct this test the evaluator shall generate or obtain test vectors from a known good implementation of the TOE supported schemes. For each combination of supported key establishment scheme and its options (with our without key confirmation if supported, for each supported key confirmation MAC function if key confirmation is supported, and for each supported mask generation function if KTSOAEP is supported), the tester shall generate 10 sets of test vectors. Each test vector shall include the RSA private key, the plaintext keying material (KeyData), any additional input parameters if applicable, the MacTag in cases where key confirmation is incorporated, and the outputted ciphertext. For each test vector, the evaluator shall perform the key establishment decryption operation on the TOE and ensure that the outputted plaintext keying material (KeyData) is equivalent to the plaintext keying material in the test vector. In cases where key confirmation is incorporated, the evaluator shall perform the key confirmation steps and ensure that the outputted MacTag is equivalent to the MacTag in the test vector.

The evaluator shall ensure that the TSS describes how the TOE handles decryption errors. In accordance with NIST Special Publication 800-56B, the TOE must not reveal the particular error that occurred, either through the contents of any outputted or logged error message or through timing variations. If KTS-OAEP is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.2.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each. If KTS-KEM-KWS is supported, the evaluator shall create separate contrived ciphertext values that trigger each of the three decryption error checks described in NIST Special Publication 800-56B section 7.2.3.3, ensure that each decryption attempt results in an error, and ensure that any outputted or logged error message is identical for each.

RSA-based key establishment

The evaluator shall verify the correctness of the TSF's implementation of RSAESPKCS1-v1_5 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses RSAES-PKCS1-v1_5.

Diffie-Hellman Group 14

The evaluator shall verify the correctness of the TSF's implementation of Diffie-Hellman group 14 by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses Diffie-Hellman group 14.

FFC Schemes using "safe-prime" groups

The evaluator shall verify the correctness of the TSF's implementation of safe-prime groups by using a known good implementation for each protocol selected in FTP_DIT_EXT.1 that uses safe-prime groups. This test must be performed for each safe-prime group that each protocol uses.

FCS_STO_EXT.1 Storage of Credentials (FCS_STO_EXT.1)

FCS_STO_EXT.1.1

TSS

The evaluator shall check the TSS to ensure that it lists all persistent credentials (secret keys, PKI private keys, or passwords) needed to meet the requirements in the ST. For each of these items, the evaluator shall confirm that the TSS lists for what purpose it is used, and how it is stored.

Guidance

None.

Tests

For all credentials for which the application implements functionality, the evaluator shall verify credentials are encrypted according to FCS_COP.1/(1) or conditioned according to FCS_CKM.1.1/(1) and FCS_CKM.1/(3). For all credentials for which the application invokes platform-provided functionality, the evaluator shall perform the following actions which vary per platform.

The evaluator shall verify that all certificates are stored in the Windows Certificate Store. The evaluator shall verify that other credentials, like passwords, are stored in the Windows Credential Manager or stored using the Data Protection API (DPAPI). The evaluator shall verify that the application is using the ProtectData class and storing credentials in IsolatedStorage.

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

FCS_COP.1.1(1)

TSS

None.

Guidance

The evaluator checks the AGD documents to determine that any configuration that is required to be done to configure the functionality for the required modes and key sizes is present.

Tests

The evaluator shall perform all of the following tests for each algorithm implemented by the TSF and used to satisfy the requirements of this PP:

AES-CBC Known Answer Tests: There are four Known Answer Tests (KATs), described below. In all KATs, the plaintext, ciphertext, and IV values shall be 128-bit blocks. The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

- KAT-1. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 plaintext values and obtain the ciphertext value that results from AESCBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros. Five plaintext values shall be encrypted with a 128-bit all-zeros key, and the other five shall be encrypted with a 256-bit all- zeros key. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using 10 ciphertext values as input and AES-CBC decryption.
- KAT-2. To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of 10 key values and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros. Five of the keys shall be 128-bit keys, and the other five shall be 256-bit keys. To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using an all-zero ciphertext value as input and AES-CBC decryption.
- KAT-3. To test the encrypt functionality of AES-CBC, the evaluator shall supply the two sets of key values described below and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using the given key value and an IV of all zeros. The first set of keys shall have 128 128-bit keys, and the second set shall have 256 256-bit keys. Key i in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. To test the decrypt functionality of AES-CBC, the evaluator shall supply the two sets of key and ciphertext value pairs described below and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using the given key and an IV of all zeros. The first set of key/ciphertext pairs shall have 128 128-bit key/ciphertext pairs, and the second set of key/ciphertext pairs shall have 256 256-bit key/ciphertext pairs. Key I in each set shall have the leftmost i bits be ones and the rightmost N-i bits be zeros, for i in [1,N]. The ciphertext value in each pair shall be the value that results in an all-zeros plaintext when decrypted with its corresponding key.

- KAT-4. To test the encrypt functionality of AES-CBC, the evaluator shall supply the set of 128 plaintext values described below and obtain the two ciphertext values that result from AES-CBC encryption of the given plaintext using a 128-bit key value of all zeros with an IV of all zeros and using a 256-bit key value of all zeros with an IV of all zeros, respectively. Plaintext value i in each set shall have the leftmost i bits be ones and the rightmost 128-i bits be zeros, for i in [1,128].

To test the decrypt functionality of AES-CBC, the evaluator shall perform the same test as for encrypt, using ciphertext values of the same form as the plaintext in the encrypt test as input and AES-CBC decryption.

AES-CBC Multi-Block Message Test: The evaluator shall test the encrypt functionality by encrypting an i-block message where 1 < i <= 10. The evaluator shall choose a key, an IV and plaintext message of length i blocks and encrypt the message, using the mode to be tested, with the chosen key and IV. The ciphertext shall be compared to the result of encrypting the same plaintext message with the same key and IV using a known good implementation. The evaluator shall also test the decrypt functionality for each mode by decrypting an i-block message where 1 < i <=10. The evaluator shall choose a key, an IV and a ciphertext message of length i blocks and decrypt the message, using the mode to be tested, with the chosen key and IV. The plaintext shall be compared to the result of decrypting the same ciphertext message with the same key and IV using a known good implementation. AES-CBC Monte Carlo Tests The evaluator shall test the encrypt functionality using a set of 200 plaintext, IV, and key 3- tuples. 100 of these shall use 128 bit keys, and 100 shall use 256 bit keys. The plaintext and IV values shall be 128-bit blocks. For each 3-tuple, 1000 iterations shall be run as follows:

```
# Input: PT, IV, Key
for i = 1 to 1000:
    if i == 1:
        CT[1] = AES-CBC-Encrypt(Key, IV, PT)
        PT = IV
    else:
        CT[i] = AES-CBC-Encrypt(Key, PT)
        PT = CT[i-1]
```

The ciphertext computed in the 1000th iteration (i.e., CT[1000]) is the result for that trial.

This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as for encrypt, exchanging CT and PT and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

AES-GCM Monte Carlo Tests: The evaluator shall test the authenticated encrypt functionality of AES-GCM for each combination of the following input parameter lengths:

128 bit and 256 bit keys

Two plaintext lengths. One of the plaintext lengths shall be a non-zero integer multiple of 128 bits, if supported. The other plaintext length shall not be an integer multiple of 128 bits, if supported.

Three AAD lengths. One AAD length shall be 0, if supported. One AAD length shall be a non-zero integer multiple of 128 bits, if supported. One AAD length shall not be an integer multiple of 128 bits, if supported.

Two IV lengths. If 96 bit IV is supported, 96 bits shall be one of the two IV lengths tested.

The evaluator shall test the encrypt functionality using a set of 10 key, plaintext, AAD, and IV tuples for each combination of parameter lengths above and obtain the ciphertext value and tag that results from AES-GCM authenticated encrypt. Each supported tag length shall be tested at least once per set of 10. The IV value may be supplied by the evaluator or the implementation being tested, as long as it is known.

The evaluator shall test the decrypt functionality using a set of 10 key, ciphertext, tag, AAD, and IV 5-tuples for each combination of parameter lengths above and obtain a Pass/Fail result on authentication and the decrypted plaintext if Pass. The set shall include five tuples that Pass and five that Fail.

The results from each test may either be obtained by the evaluator directly or by supplying the inputs to the implementer and receiving the results in response. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known good implementation.

AES-XTS Tests: The evaluator shall test the encrypt functionality of XTS-AES for each combination of the following input parameter lengths: 256 bit (for AES-128) and 512 bit (for AES-256) keys.

Three data unit (i.e., plaintext) lengths. One of the data unit lengths shall be a non-zero integer multiple of 128 bits, if supported. One of the data unit lengths shall be an integer multiple of 128 bits, if supported. The third data unit length shall be either the longest supported data unit length or 216 bits, whichever is smaller.

Using a set of 100 (key, plaintext and 128-bit random tweak value) 3-tuples and obtain the ciphertext that results from XTS-AES encrypt.

The evaluator may supply a data unit sequence number instead of the tweak value if the implementation supports it. The data unit sequence number is a base-10 number ranging between 0 and 255 that implementations convert to a tweak value internally.

The evaluator shall test the decrypt functionality of XTS-AES using the same test as for encrypt, replacing plaintext values with ciphertext values and XTS-AES encrypt with XTS-AES decrypt.

FCS_COP.1(2) Cryptographic Operation – Hashing (FCS_COP.1(2))

FCS_COP.1.1(2)

TSS

The evaluator shall check that the association of the hash function with other application cryptographic functions (for example, the digital signature verification function) is documented in the TSS.

Guidance

None.

Tests

The TSF hashing functions can be implemented in one of two modes. The first mode is the byte-oriented mode. In this mode the TSF hashes only messages that are an integral number of bytes in length; i.e., the length (in bits) of the message to be hashed is divisible by 8. The second mode is the bit-oriented mode. In this mode the TSF hashes messages of arbitrary length. As there are different tests for each mode, an indication is given in the following sections for the bit-oriented vs. the byte-oriented testmacs. The evaluator shall perform all of the following tests for each hash algorithm implemented by the TSF and used to satisfy the requirements of this PP.

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

- Test 1: Short Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m bits. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.
- Test 2: Short Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8+1 messages, where m is the block length of the hash algorithm. The length of the messages range sequentially from 0 to m/8 bytes, with each message being an integral number of bytes. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.
- Test 3: Selected Long Messages Test - Bit oriented Mode The evaluators devise an input set consisting of m messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 99*i, where 1 ≤ i ≤ m. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.

- Test 4: Selected Long Messages Test - Byte oriented Mode The evaluators devise an input set consisting of m/8 messages, where m is the block length of the hash algorithm. The length of the ith message is 512 + 8*99*i, where 1 ≤ i ≤ m/8. The message text shall be pseudorandomly generated. The evaluators compute the message digest for each of the messages and ensure that the correct result is produced when the messages are provided to the TSF.
- Test 5: Pseudorandomly Generated Messages Test This test is for byte-oriented implementations only. The evaluators randomly generate a seed that is n bits long, where n is the length of the message digest produced by the hash function to be tested. The evaluators then formulate a set of 100 messages and associated digests by following the algorithm provided in Figure 1 of [SHAVS]. The evaluators then ensure that the correct result is produced when the messages are provided to the TSF.

FCS_COP.1(3) Cryptographic Operation – Signing (FCS_COP.1(3))

FCS_COP.1.1(3)

The evaluator shall perform the following activities based on the selections in the ST.

TSS

Guidance

Tests

The following tests require the developer to provide access to a test application that provides the evaluator with tools that are typically not found in the production application.

ECDSA Algorithm Tests

- Test 1: ECDSA FIPS 186-4 Signature Generation Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate 10 1024-bit long messages and obtain for each message a public key and the resulting signature values R and S. To determine correctness, the evaluator shall use the signature verification function of a known good implementation.
- Test 2: ECDSA FIPS 186-4 Signature Verification Test. For each supported NIST curve (i.e., P-256, P-384 and P-521) and SHA function pair, the evaluator shall generate a set of 10 1024-bit message, public key and signature tuples and modify one of the values (message, public key or signature) in five of the 10 tuples. The evaluator shall obtain in response a set of 10 PASS/FAIL values.

RSA Signature Algorithm Tests

- Test 1: Signature Generation Test. The evaluator shall verify the implementation of RSA Signature Generation by the TOE using the Signature Generation Test. To conduct this test the evaluator must generate or obtain 10 messages from a trusted reference implementation for each modulus size/SHA combination supported by the TSF. The evaluator shall have the TOE use their private key and modulus value to sign these messages. The evaluator shall verify the correctness of the TSF's signature using a known good implementation and the associated public keys to verify the signatures.

- Test 2: Signature Verification Test. The evaluator shall perform the Signature Verification test to verify the ability of the TOE to recognize another party's valid and invalid signatures. The evaluator shall inject errors into the test vectors produced during the Signature Verification Test by introducing errors in some of the public keys, e, messages, IR format, and/or signatures. The TOE attempts to verify the signatures and returns success or failure.

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

FCS_COP.1.1(4)

The evaluator shall perform the following activities based on the selections in the ST.

TSS

Guidance

None.

Tests

For each of the supported parameter sets, the evaluator shall compose 15 sets of test data. Each set shall consist of a key and message data. The evaluator shall have the TSF generate HMAC tags for these sets of test data. The resulting MAC tags shall be compared to the result of generating HMAC tags with the same key and IV using a known-good implementation.

FCS_HTTPS_EXT.1 HTTPS Protocol (FCS_HTTPS_EXT.1)

FCS_HTTPS_EXT.1.1

TSS

None.

Guidance

None.

Tests

The evaluator shall attempt to establish an HTTPS connection with a webserver, observe the traffic with a packet analyzer, and verify that the connection succeeds and that the traffic is identified as TLS or HTTPS.

FCS_HTTPS_EXT.1.2

TSS

None.

Guidance

None.

Tests

Other tests are performed in conjunction with FCS_TLS_EXT.1 and FCS_TLSS_EXT.1.

FCS_TLS_EXT.1 TLS Protocol (FCS_TLS_EXT.1)

FCS_TLS_EXT.1.1

Guidance

The evaluator shall ensure that the selections indicated in the ST are consistent with selections in the dependent components.

FCS_TLSS_EXT.1 TLS Server Protocol (FCS_TLSS_EXT.1)

FCS_TLSS_EXT.1.1

TSS

The evaluator shall check the description of the implementation of this protocol in the TSS to ensure that the cipher suites supported are specified. The evaluator shall check the TSS to ensure that the cipher suites specified include those listed for this component.

Guidance

The evaluator shall also check the operational guidance to ensure that it contains instructions on configuring the TOE so that TLS conforms to the description in the TSS.

Tests

The evaluator shall also perform the following tests:

- Test 1: The evaluator shall establish a TLS connection using each of the cipher suites specified by the requirement. This connection may be established as part of the establishment of a higher-level protocol, e.g., as part of an EAP session. It is sufficient to observe the successful negotiation of a cipher suite to satisfy the intent of the test; it is not necessary to examine the characteristics of the encrypted traffic in an attempt to discern the cipher suite being used (for example, that the cryptographic algorithm is 128-bit AES and not 256-bit AES).
- Test 2: The evaluator shall send a Client Hello to the server with a list of cipher suites that does not contain any of the cipher suites in the server's ST and verify that the server denies the connection. Additionally, the evaluator shall send a Client Hello to the server containing only the TLS_NULL_WITH_NULL_NULL cipher suite and verify that the server denies the connection.
- Test 3: If RSA key exchange is used in one of the selected ciphersuites, the evaluator shall use a client to send a properly constructed Key Exchange message with a modified EncryptedPreMasterSecret field during the TLS handshake. The evaluator shall verify that the handshake is not completed successfully and no application data flows.
- Test 4: The evaluator shall perform the following modifications to the traffic:
    - Test 4.1: Change the TLS version proposed by the client in the Client Hello to a non-supported TLS version (for example 1.3 represented by the two bytes 03 04) and verify that the server rejects the connection.
    - Test 4.2: Modify a byte in the data of the client's Finished handshake message, and verify that the server rejects the connection and does not send any application data.
    - Test 4.3: Demonstrate that the TOE will not resume a session for which the client failed to complete the handshake (independent of TOE support for session resumption): Generate a Fatal Alert by sending a Finished message from the client before the client sends a ChangeCipherSpec message, and then send a Client Hello with the session identifier from the previous incomplete session, and verify that the server does not resume the session.
    - Test 4.4: Send a message consisting of random bytes from the client after the client has issued the ChangeCipherSpec message and verify that the server denies the connection.

FCS_TLSS_EXT.1.2

TSS

The evaluator shall verify that the TSS contains a description of the denial of old SSL and TLS versions consistent relative to selections in FCS_TLSS_EXT.1.2.

Guidance

The evaluator shall verify that the AGD guidance includes any configuration necessary to meet this requirement.

Tests

Test 1: The evaluator shall send a Client Hello requesting a connection with version SSL 2.0 and verify that the server denies the connection. The evaluator shall repeat this test with SSL 3.0 and TLS 1.0, and TLS 1.1 if it is selected.

FCS_TLSS_EXT.1.3

TSS

The evaluator shall verify that the TSS describes the key agreement parameters of the server's Key Exchange message.

Guidance

The evaluator shall verify that any configuration guidance necessary to meet the requirement must be contained in the AGD guidance.

Tests

The evaluator shall conduct the following tests. The testing can be carried out manually with a packet analyzer or with an automated framework that similarly captures such empirical evidence. Note that this testing can be accomplished in conjunction with other testing activities. For each of the following tests, determining that the size matches the expected size is sufficient.

- Test 1: [conditional] If RSA-based key establishment is selected, the evaluator shall attempt a connection using RSA-based key establishment with a supported size. The evaluator shall verify that the size used matches that which is configured.
- The evaluator shall repeat this test for each supported size of RSA-based key establishment.
- Test 2: [conditional] If finite-field (i.e. non-EC) Diffie-Hellman ciphers are selected, the evaluator shall attempt a connection using a Diffie-Hellman key exchange with a supported parameter size or supported group. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported parameter size or group.
- Test 3: [conditional] If ECDHE ciphers are selected, the evaluator shall attempt a connection using an ECDHE ciphersuite with a supported curve. The evaluator shall verify that the key agreement parameters in the Key Exchange message are the ones configured. The evaluator shall repeat this test for each supported elliptic curve.

FMT_SMF.1 Specification of Management Functions (FMT_SMF.1)

FMT_SMF.1.1

TSS

None.

Guidance

The evaluator shall verify that every management function mandated by the ST is described in the operational guidance and that the description contains the information required to perform the management duties associated with the management function.

Tests

The evaluator shall test the application's ability to provide the management functions by configuring the application and testing each option selected from above. The evaluator is expected to test these functions in all the ways in which the ST and guidance documentation state the configuration can be managed.

FMT_MEC_EXT.1 Supported Configuration Mechanism (FMT_MEC_EXT.1)

FMT_MEC_EXT.1.1

TSS

The evaluator shall review the TSS to identify the application's configuration options (e.g. settings) and determine whether these are stored and set using the mechanisms supported by the platform. At a minimum the TSS shall list settings related to any SFRs and any settings that are mandated in the operational guidance in response to an SFR.

Guidance

None.

Tests

The method of testing varies per platform.

The evaluator shall determine the Windows.UI.ApplicationSettings namespace or the IsolatedStorageSettings namespace for storing application specific settings. For Classic Desktop applications, the evaluator shall run the application while monitoring it with the SysInternals tool ProcMon and make changes to its configuration. The evaluator shall verify that ProcMon logs show corresponding changes to the the Windows Registry or C:\ProgramData\ directory.

FMT_CFG_EXT.1 Secure by Default Configuration (FMT_CFG_EXT.1)

FMT_CFG_EXT.1.2

TSS

None.

Guidance

None.

Tests

The evaluator shall install and run the application. The evaluator shall inspect the filesystem of the platform (to the extent possible) for any files created by the application and ensure that their permissions are adequate to protect them. The method of doing so varies per platform.

The evaluator shall run the SysInternals tools, Process Monitor and Access Check (or tools of equivalent capability, like icacls.exe) for Classic Desktop applications to verify that files written to disk during an application's installation have the correct file permissions, such that a standard user cannot modify the application or its data files. The evaluator shall consider the requirement met because of the AppContainer sandbox.

FTP_DIT_EXT.1 Protection of Data in Transit (FTP_DIT_EXT.1)

FTP_DIT_EXT.1.1

TSS

For platform-provided functionality, the evaluator shall verify the TSS contains the calls to the platform that TOE is leveraging to invoke the functionality.

Guidance

None.

Tests

The evaluator shall perform the following tests.

- Test 1: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall verify from the packet capture that the traffic is encrypted with HTTPS or TLS in accordance with the selection in the ST.

- Test 2: The evaluator shall exercise the application (attempting to transmit data; for example by connecting to remote systems or websites) while capturing packets from the application. The evaluator shall review the packet capture and verify that no sensitive data is transmitted in the clear.
- Test 3: The evaluator shall inspect the TSS to determine if user credentials are transmitted. If credentials are transmitted the evaluator shall set the credential to a known value. The evaluator shall capture packets from the application while causing credentials to be transmitted as described in the TSS. The evaluator shall perform a string search of the captured network packets and verify that the plaintext credential previously set by the evaluator is not found.

FDP_DEC_EXT.1 Access to Platform Resources (FDP_DEC_EXT.1)

FDP_DEC_EXT.1.1

TSS

None.

Guidance

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to hardware resources. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each resource which it accesses, identify the justification as to why access is required.

Tests

The evaluator shall check the WMAppManifest.xml file for a list of required hardware capabilities. The evaluator shall verify that the user is made aware of the required hardware capabilities when the application is first installed. This includes permissions such as ID_CAP_ISV_CAMERA, ID_CAP_LOCATION, ID_CAP_NETWORKING, ID_CAP_MICROPHONE, ID_CAP_PROXIMITY and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

The evaluator shall identify in either the application software or its documentation the list of the required hardware resources.

FDP_DEC_EXT.1.2

TSS

None.

Guidance

The evaluator shall perform the platform-specific actions below and inspect user documentation to determine the application's access to sensitive information repositories. The evaluator shall ensure that this is consistent with the selections indicated. The evaluator shall review documentation provided by the application developer and for each sensitive information repository which it accesses, identify the justification as to why access is required.

Tests

The evaluator shall check the WMAppManifest.xml file for a list of required capabilities. The evaluator shall identify the required information repositories when the application is first installed. This includes permissions such as ID_CAP_CONTACTS,ID_CAP_APPOINTMENTS,ID_CAP_MEDIALIB and so on. A complete list of Windows App permissions can be found at:

http://msdn.microsoft.com/en-US/library/windows/apps/jj206936.aspx

For Windows Desktop Applications the evaluator shall identify in either the application software or its documentation the list of sensitive information repositories it accesses.

FDP_NET_EXT.1 Network Communications (FDP_NET_EXT.1)

FDP_NET_EXT.1.1

TSS

None.

Guidance

None.

Tests

The evaluator shall perform the following tests:

- Test 1: The evaluator shall run the application. While the application is running, the evaluator shall sniff network traffic ignoring all non-application associated traffic and verify that any network communications witnessed are documented in the TSS or are user-initiated.
- Test 2: The evaluator shall run the application. After the application initializes, the evaluator shall run network port scans to verify that any ports opened by the application have been captured in the ST for the third selection and its assignment. This includes

connection-based protocols (e.g. TCP, DCCP) as well as connectionless protocols (e.g. UDP).

FDP_DAR_EXT.1 Encryption Of Sensitive Application Data (FDP_DAR_EXT.1)

FDP_DAR_EXT.1.1

TSS

The evaluator shall examine the TSS to ensure that it describes the sensitive data processed by the application. The evaluator shall then ensure that the following activities cover all of the sensitive data identified in the TSS.

If not store any sensitive data is selected, the evaluator shall inspect the TSS to ensure that it describes how sensitive data cannot be written to non-volatile memory.

The evaluator shall also ensure that this is consistent with the filesystem test below.

Guidance

None.

Tests

Evaluation activities (after the identification of the sensitive data) are to be performed on all sensitive data listed that are not covered by FCS_STO_EXT.1.

The evaluator shall inventory the filesystem locations where the application may write data. The evaluator shall run the application and attempt to store sensitive data. The evaluator shall then inspect those areas of the filesystem to note where data was stored (if any), and determine whether it has been encrypted. If leverage platform-provided functionality is selected, the evaluation activities will be performed as stated in the following requirements, which vary on a per-platform basis.

The Windows platform currently does not provide data-at-rest encryption services which depend upon invocation by application developers. The evaluator shall verify that the Operational User Guidance makes the need to activate platform encryption, such as BitLocker or Encrypting File System (EFS), clear to the end user.

FPR_ANO_EXT.1 User Consent for Transmission of Personally Identifiable Information (FPR_ANO_EXT.1)

FPR_ANO_EXT.1.1

TSS

The evaluator shall inspect the TSS documentation to identify functionality in the application where PII can be transmitted.

Guidance

None.

Tests

If require user approval before executing is selected, the evaluator shall run the application and exercise the functionality responsibly for transmitting PII and verify that user approval is required before transmission of the PII.

FPT_API_EXT.1 Use of Supported Services and APIs (FPT_API_EXT.1)

FPT_API_EXT.1.1

TSS

The evaluator shall verify that the TSS lists the platformAPIs used in the application.

Guidance

None.

Tests

The evaluator shall then compare the list with the supported APIs (available through e.g., developer accounts, platform developer groups) and ensure that all APIs listed in the TSS are supported.

FPT_AEX_EXT.1 Anti-Exploitation Capabilities (FPT_AEX_EXT.1)

FPT_AEX_EXT.1.1

TSS

The evaluator shall ensure that the TSS describes the compiler flags used to enable ASLR when the application is compiled.

Guidance

None.

Tests

The evaluator shall perform either a static or dynamic analysis to determine that no memory mappings are placed at an explicit and consistent address. The method of doing so varies per platform.

The evaluator shall run the same application on two different Windows systems and run a tool that will list all memory mapped addresses for the application. The evaluator shall then verify the two different instances share no mapping locations. The Microsoft SysInternals tool, VMMap, could be used to view memory addresses of a running application. The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application has ASLR enabled.

FPT_AEX_EXT.1.2

TSS

None.

Guidance

None.

Tests

The evaluator shall verify that no memory mapping requests are made with write and execute permissions. The method of doing so varies per platform.

The evaluator shall use a tool such as Microsoft's BinScope Binary Analyzer to confirm that the application passes the NXCheck. The evaluator may also ensure that the /NXCOMPAT flag was used during compilation to verify that DEP protections are enabled for the application.

FPT_AEX_EXT.1.3

TSS

None.

Guidance

None.

Tests

The evaluator shall configure the platform in the ascribed manner and carry out one of the prescribed tests:

If the OS platform supports Windows Defender Exploit Guard (Windows 10 version 1709 or later), then the evaluator shall ensure that the application can run successfully with Windows Defender Exploit Guard Exploit Protection configured with the following minimum mitigations enabled; Control Flow Guard (CFG), Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), Import address filtering (IAF), and Data Execution Prevention (DEP). The following link describes how to enable Exploit Protection, https://docs.microsoft.com/en-s/windows/security/threatprotection/windows-defender-exploit-guard/customize-exploit-protection. If the OS platform supports the Enhanced Mitigation Experience Toolkit (EMET) which can be installed on Windows 10 version 1703 and earlier, then the evaluator shall ensure that the application can run successfully with EMET configured with the following minimum mitigations enabled; Memory Protection Check, Randomize memory allocations (Bottom-Up ASLR), Export address filtering (EAF), and Data Execution Prevention (DEP).

FPT_AEX_EXT.1.4

TSS

None.

Guidance

None.

Tests

The evaluator shall run the application and determine where it writes its files. For files where the user does not choose the destination, the evaluator shall check whether the destination directory contains executable files. This varies per platform:

The evaluator shall consider the requirement met because the platform forces applications to write all data within the application working directory (sandbox). For Windows Desktop Applications the evaluator shall run the program, mimicking normal usage, and note where all usermodifiable files are written. The evaluator shall ensure that there are no executable files stored in the same directories to which the application wrote user-modifiable files and no data files in the application's install directory.

FPT_AEX_EXT.1.5

The evaluator will inspect every native executable included in the TOE to ensure that stack-based buffer overflow protection is present.

Applications that run as Managed Code in the .NET Framework do not require these stack protections. Applications developed in Object Pascal using the Delphi IDE compiled with

RangeChecking enabled comply with this element. For other code, the evaluator shall review the TSS and verify that the /GS flag was used during compilation. The evaluator shall run a tool like, BinScope, that can verify the correct usage of /GS.

FPT_IDV_EXT.1 Software Identification and Versions (FPT_IDV_EXT.1)

FPT_IDV_EXT.1.1

TSS

If "other version information" is selected the evaluator shall verify that the TSS contains an explaination of the versioning methodology.

Guidance

None.

Tests

The evaluator shall install the application, then check for the / existence of version information. If SWID tags is selected the evaluator shall check for a .swidtag file. The evaluator shall open the file and verify that is contains at least a SoftwareIdentity element and an Entity element.

FPT_TUD_EXT.1 Integrity for Installation and Update (FPT_TUD_EXT.1)

FPT_TUD_EXT.1.2

TSS

None.

Guidance

The evaluator shall verify guidance includes a description of how to query the current version of the application.

Tests

The evaluator shall query the application for the current version of the software according to the operational user guidance. The evaluator shall then verify that the current version matches that of the documented and installed version.

FPT_TUD_EXT.1.3

TSS

None.

Guidance

None.

Tests

The evaluator shall verify that the application's executable files are not changed by the application. The evaluator shall complete the following test:

- Test 1: The evaluator shall install the application and then locate all of its executable files. The evaluator shall then, for each file, save off either a hash of the file or a copy of the file itself. The evaluator shall then run the application and exercise all features of the application as described in the ST. The evaluator shall then compare each executable file with the either the saved hash or the saved copy of the files. The evaluator shall verify that these are identical.

FPT_TUD_EXT.1.4

TSS

The evaluator shall verify that the TSS identifies how the application installation package and updates to it are signed by an authorized source. The definition of an authorized source must be contained in the TSS. The evaluator shall also ensure that the TSS (or the operational guidance) describes how candidate updates are obtained.

Guidance

None.

Tests

None.

FPT_TUD_EXT.1.5

TSS

The evaluator shall verify that the TSS identifies how the application is distributed. If "with the platform" is selected the evaluated shall perform a clean installation or factory reset to confirm that TOE software is included as part of the platform OS. If "as an additional package" is selected the evaluator shall perform the tests in FPT_TUD_EXT.2.

Guidance

None.

Tests

None.

FPT_TUD_EXT.2 Integrity for Installation and Update (FPT_TUD_EXT.2)

FPT_TUD_EXT.2.1

TSS

None.

Guidance

None.

Tests

The evaluator shall verify that application updates are distributed in the format supported by the platform. This varies per platform:

The evaluator shall ensure that the application is packaged in the standard Windows Installer (.MSI) format, the Windows Application Software (.EXE) format signed using the Microsoft Authenticode process, or the package (.APPX) format. See https://msdn.microsoft.com/enus/library/ms537364(v=vs.85).aspx for details regarding Authenticode signing.

FPT_TUD_EXT.2.2

TSS

None.

Guidance

None.

Tests

The evaluator shall record the path of every file on the entire filesystem prior to installation of the application, and then install and run the application. Afterwards, the evaluator shall then uninstall the application, and compare the resulting filesystem to the initial record to verify that no files, other than configuration, output, and audit/log files, have been added to the filesystem.

6.3 SECURITY REQUIREMENTS RATIONALE

6.3.1 NECESSITY AND SUFFICIENCY ANALYSIS

SFR / TOE Security Objective	O.INTEGRITY	O.QUALITY	O.MANAGEMENT	O.PROTECTED_STORAGE	O.PROTECTED_COMMS
FCS_CKM.2		X			X
FCS_RBG_EXT.1		X		X	X
FCS_CKM_EXT.2				X	
FCS_RBG_EXT.2				X	X
FCS_CKM_EXT.1		X			X
FCS_STO_EXT.1		X		X	
FDP_DEC_EXT.1	X				
FDP_NET_EXT.1					X
FDP_DAR_EXT.1		X		X	
FMT_MEC_EXT.1		X			

SFR / TOE Security Objective	O.INTEGRITY	O.QUALITY	O.MANAGEMENT	O.PROTECTED_STORAGE	O.PROTECTED_COMMS
FMT_CFG_EXT.1	X				
FPR_ANO_EXT.1			X		
FPT_API_EXT.1		X			
FPT_AEX_EXT.1	X				
FPT_TUD_EXT.1	X		X		
FPT_IDV_EXT.1			X		
FTP_DIT_EXT.1		X			X
FTP_ITC.1/SNMPv3					X
FCS_TLSS_EXT.1					X
FCS_COP.1/(4)				X	X
FMT_SMF.1			X		
FCS_HTTPS_EXT.1					X
FCS_CKM.1		X			X
FCS_COP.1/(1)				X	X

SFR / TOE Security Objective	O.INTEGRITY	O.QUALITY	O.MANAGEMENT	O.PROTECTED_STORAGE	O.PROTECTED_COMMS
FCS_COP.1/(3)					X
FCS_COP.1/(2)				X	X
FPT_TUD_EXT.2		X			
FCS_TLS_EXT.1					X

Table 5 *SFRs / TOE Security Objectives coverage*

6.3.2 SECURITY REQUIREMENT SUFFICIENCY

O.INTEGRITY: This objective is fulfilled by the following SFRs:

FDP_DEC_EXT.1 The ST includes FDP_DEC_EXT.1 to limit access to platform hardware resources, which limits the methods by which an attacker can attempt to compromise the integrity of the TOE.

FMT_CFG_EXT.1 The ST includes FMT_CFG_EXT.1 for the TSF to limit unauthorized access to itself by ensuring that the TOE uses appropriately restrictive platform permissions on its binaries and data.

FPT_AEX_EXT.1 The ST includes FPT_AEX_EXT.1 to add complexity to the task of compromising systems by ensuring that application is compatible with security features provided by the platform vendor and that the most important executables form the application implement platform-provided anti-exploitations such as ASLR and stack overflow protection.

FPT_TUD_EXT.1 The ST includes FPT_TUD_EXT.1 to ensure that the TOE can be patched and that any updates to the TOE have appropriate integrity protection.

O.QUALITY: This objective is fulfilled by the following SFRs:

FCS_CKM_EXT.1 The ST supports this objective by allowing FCS_CKM_EXT.1 to specify that the TSF may rely on platform-provided key generation services.

FCS_RBG_EXT.1 The ST supports this objective by allowing FCS_RBG_EXT.1 to specify that the TSF may rely on platform-provided random bit generation services.

FCS_STO_EXT.1 The ST supports this objective by allowing FCS_STO_EXT.1 to specify that the TSF may rely on platform-provided credential storage services.

FDP_DAR_EXT.1 The ST supports this objective by allowing FDP_DAR_EXT.1 to specify that the TSF may rely on platform-provided data-at-rest protection services.

FMT_MEC_EXT.1 The ST includes FMT_MEC_EXT.1 to ensure that the TOE can use platform services to store and set configuration options.

FPT_API_EXT.1 The ST includes FPT_API_EXT.1 to require the TOE to leverage platform functionality by using only documented and supported APIs.

FTP_DIT_EXT.1 The ST supports this objective by allowing FTP_DIT_EXT.1 to specify that the TSF may rely on platform-provided services to implement trusted communications.

FCS_CKM.1 The ST supports this objective by allowing FCS_CKM.1 to specify that the TSF may rely on platform-provided asymmetric key generation services.

FCS_CKM.2 The ST supports this objective by allowing FCS_CKM.2 to specify that the TSF may rely on platform-provided key establishment services.

FPT_TUD_EXT.2 The TSF includes FPT_TUD_EXT.2 to specify that the TOE may leverage the platform-supported package manager for application distribution and the TSF to remove all traces of itself when removed from the platform system.

O.MANAGEMENT: This objective is fulfilled by the following SFRs:

FMT_SMF.1 The ST includes FMT_SMF.1 to define the security-relevant management functions that are supported by the TOE.

FPR_ANO_EXT.1 The ST includes FPR_ANO_EXT.1 to define how the TSF provides control to the user regarding the disclosure of any PII.

FPT_IDV_EXT.1 The ST includes FPT_IDV_EXT.1 to provide a methodology for identifying the TOE versioning.

FPT_TUD_EXT.1 The ST includes FPT_TUD_EXT.1 to define how updates to the TOE are deployed and verified.

O.PROTECTED_STORAGE: This objective is fulfilled by the following SFRs:

FCS_RBG_EXT.1 The ST includes FCS_RBG_EXT.1 to define whether random bit generation services are implemented by the TSF or the platform. Depending on how data at rest is protected, the TOE may rely on the use of a random bit generator to create keys that are subsequently used for data protection.

FCS_STO_EXT.1 The ST includes FCS_STO_EXT.1 to define the mechanism that the TSF uses or relies upon to protect stored credential data.

FDP_DAR_EXT.1 The ST includes FDP_DAR_EXT.1 to define the mechanism that the TSF uses or relies upon to protect sensitive data at rest.

FCS_CKM_EXT.2 The ST includes FCS_CKM_EXT.2 to define the password-based key derivation function that may be used to encrypt stored credential data based on the claims made in FCS_STO_EXT.1.

FCS_COP.1/(1) The ST includes FCS_COP.1/(1) to define the AES cryptographic algorithm that may be used to encrypt stored credential data based on the claims made in FCS_STO_EXT.1.

FCS_COP.1/(2) The ST includes FCS_COP.1/(2) to define integrity mechanisms that may be used by the TOE as part of ensuring that data at rest is protected.

FCS_COP.1/(4) The ST includes FCS_COP.1/(4) to define HMAC mechanisms that may be used by the TOE as part of ensuring that data at rest is protected.

FCS_RBG_EXT.2 The ST includes FCS_RBG_EXT.2 to define the TOE's implementation of random bit generation functionality in the event that the TOE provides this function in support of generating keys that are used for data protection.

O.PROTECTED_COMMS: This objective is fulfilled by the following SFRs:

FCS_RBG_EXT.1 The ST includes FCS_RBG_EXT.1 to define whether the random bit generation services used in establishing trusted communications are implemented by the TSF or by the platform.

FCS_CKM_EXT.1 The ST includes FCS_CKM_EXT.1 to specify whether the TOE or the platform is responsible for generation of any asymmetric keys that may be used for establishing trusted communications.

FTP_DIT_EXT.1 The ST includes FTP_DIT_EXT.1 to define the trusted channels used to protect data in transit, the data that is protected, and whether the trusted channels are implemented by the TSF or the platform.

FCS_CKM.1 The ST includes FCS_CKM.1 to define whether the TSF or the platform generates asymmetric keys that are used in support of trusted communications.

FCS_CKM.2 The ST includes FCS_CKM.2 to define whether the TSF or the platform performs key establishment for trusted communications.

FCS_COP.1/(1) The ST includes FCS_COP.1/(1) to define the symmetric encryption algorithms used in support of trusted communications.

FCS_COP.1/(2) The ST includes FCS_COP.1/(2) to define the hash algorithms used in support of trusted communications.

FCS_COP.1/(3) The ST includes FCS_COP.1/(3) to define the digital signature algorithms used in support of trusted communications.

FCS_COP.1/(4) The ST includes FCS_COP.1/(4) to define the HMAC algorithms used in support of trusted communications.

FCS_RBG_EXT.2 The ST includes FCS_RBG_EXT.2 to define the DRBG algorithms used in support of trusted communications.

FCS_HTTPS_EXT.1 The ST includes FCS_HTTPS_EXT.1 to define the TOE's support for the HTTPS trusted communications protocol.

FDP_NET_EXT.1 The ST includes FDP_NET_EXT.1 to define the TOE's usage of network communications, which may include the transmission or receipt of data over a trusted channel.

FTP_ITC.1/SNMPv3 ensures that there is a separate SNMPv3 channel for each SNMP-based communication that the TOE makes with external IT entities.

FCS_TLSS_EXT.1 describes the TLSv1.2 channel ciphersuites.

FCS_TLS_EXT.1 ensures that the TOE implements TLS.

6.3.3 SFR DEPENDENCY RATIONALE

6.3.3.1 TABLE OF SFR DEPENDENCIES

The following table lists the dependencies for each requirement, indicating how they have been satisfied. The abbreviation "h.a" indicates that the dependency has been satisfied by a SFR that is hierarchically above the required dependency.

SFR	Required	Fulfilled	Missing
FCS_CKM.2	FCS_CKM.4, [FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1	FCS_CKM.4
FCS_RBG_EXT.1	None	None	None
FCS_CKM_EXT.2	FCS_COP.1, FCS_RBG_EXT.1	FCS_COP.1/(4) FCS_RBG_EXT.1	None
FCS_RBG_EXT.2	FCS_RBG_EXT.1	FCS_RBG_EXT.1	None
FCS_CKM_EXT.1	FCS_CKM.1	FCS_CKM.1	None
FCS_STO_EXT.1	[FCS_COP.1 or FCS_CKM_EXT.2]	FCS_CKM_EXT.2	None
FDP_DEC_EXT.1	None	None	None
FDP_NET_EXT.1	None	None	None
FDP_DAR_EXT.1	FCS_STO_EXT.1	FCS_STO_EXT.1	None
FMT_MEC_EXT.1	None	None	None
FMT_CFG_EXT.1	None	None	None
FPR_ANO_EXT.1	None	None	None
FPT_API_EXT.1	None	None	None
FPT_AEX_EXT.1	None	None	None
FPT_TUD_EXT.1	None	None	None
FPT_IDV_EXT.1	None	None	None

SFR	Required	Fulfilled	Missing
FTP_DIT_EXT.1	[FCS_HTTPS_EXT.1 or [FCS_TLS_EXT.1 and FCS_TLSS_EXT.1]]	FCS_HTTPS_EXT.1 FCS_TLS_EXT.1 FCS_TLSS_EXT.1	None
FTP_ITC.1/SNMPv3	None	None	None
FCS_TLSS_EXT.1	FCS_TLS_EXT.1	FCS_TLS_EXT.1	None
FCS_COP.1/(4)	FCS_CKM.4, [FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	None	FCS_CKM.4 FCS_CKM.1
FMT_SMF.1	None	None	None
FCS_HTTPS_EXT.1	FCS_TLS_EXT.1 and FCS_TLSS_EXT.1	FCS_TLS_EXT.1 FCS_TLSS_EXT.1	None
FCS_CKM.1	FCS_CKM.4, [FCS_CKM.2 or FCS_COP.1]	FCS_CKM.2	FCS_CKM.4
FCS_COP.1/(1)	FCS_CKM.4, [FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1	FCS_CKM.4
FCS_COP.1/(3)	FCS_CKM.4, [FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	FCS_CKM.1	FCS_CKM.4
FCS_COP.1/(2)	FCS_CKM.4, [FDP_ITC.1 or FDP_ITC.2 or FCS_CKM.1]	None	FCS_CKM.4 FCS_CKM.1
FPT_TUD_EXT.2	FDP_TUD_EXT.1	FDP_TUD_EXT.1	None
FCS_TLS_EXT.1	None	None	None

Table 6 SFR Dependencies

6.3.3.2 JUSTIFICATION FOR MISSING DEPENDENCIES

FCS_CKM.2 dependency on FCS_CKM.4

FCS_CKM.2 is declared in the ST in order to model the Elliptic-Curve Diffie Hellman used by the TOE's TLS-based communication channels, according to its chiphersuite.

Given that the supported ciphersuite is TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, the DH-keys used by the communication channel are ephemeral.

With DHE, the server private key (the permanent one, the one which is stored in a file, and whose public key is in the server certificate) is of type RSA (DHE_RSA ciphersuites), and is used only for signatures. The server generates a new random DH key pair (the private key will not be stored, which is how perfect forward secrecy is achieved: a private key cannot be stolen afterwards if it has never been stored), and sends the public key to the client, in a message which the server signs with its RSA private key.

With DH ciphersuites, the permanent server private key is a DH private key. The server certificate contains the DH public key. The server cannot see its RSA key be stolen because the server does not have an RSA key. The server only has a DH key. When a ciphersuites is called "DH_RSA", it means "the server key is a DH key, and the server certificate was issued (i.e. signed) by a Certification Authority who uses a RSA key".

Stealing the DH private key of one party involved in a DH key exchange allows ulterior reconstruction of the shared secret, just like RSA. In "ephemeral DH", the PFS is obtained through "ephemeral", not through "DH". Technically, it would be possible to have "ephemeral RSA" but it is not done in practice because generating a new RSA key pair is kind of expensive, whereas producing a new DH key pair is cheap.

Moreover, session-ephemeral keys are removed whenever the session gets closed. OpenSSL, which is the one responsible for this communication channel, is configured by default to provide a session timeout after 300 seconds. Another way of closing a session is by the exchange of the message *close_notify* between the client and the server, mandatory on TLS sessions as specified at [RFC 5246].

FCS_COP.1/(4) dependency on FCS_CKM.4 and FCS_CKM.1

FCS_COP.1/(4) is declared in the ST in order to model how the TOE performs an HMAC-SHA-256 operation over credentials in order to store them in a non-plaintext form.

As stated at FCS_CKM_EXT.2, the TOE uses a PBKDF function. PBKDF applies a pseudorandom function, such as hash-based message authentication code (HMAC), to the
input password or passphrase along with a salt value and repeats the process many times to produce a derived key, which can then be used as a cryptographic key in subsequent operations.

Thus, the cryptographic operation uses the following inputs:

- The pseudorandom function (HMAC-SHA)
- Salt, the public random data used as an additional input to one-way functions

- Number of iterations desired
- The desired bit-length of the derived key (256)

None of the inputs it's a secret key; therefore, none of them needs to be created nor destroyed.

FCS_CKM.1 dependency on FCS_CKM.4

FCS_CKM.1 is declared in the ST in order to model the ephemeral-asymmetric keys generated as part of the TLS session whose purpose is to derive the session symmetric key and perform message authentication.

With DHE, the server private key (the permanent one, the one which is stored in a file, and whose public key is in the server certificate) is of type RSA (DHE_RSA ciphersuites), and is used only for signatures. The server generates a new random DH key pair (the private key will not be stored, which is how perfect forward secrecy is achieved: a private key cannot be stolen afterwards if it has never been stored), and sends the public key to the client, in a message which the server signs with its RSA private key.

Moreover, session-ephemeral keys are removed whenever the session gets closed. OpenSSL, which is the one responsible for this communication channel, is configured by default to provide a session timeout after 300 seconds. Another way of closing a session is by the exchange of the message *close_notify* between the client and the server, mandatory on TLS sessions as specified at [RFC 5246].

FCS_COP.1/(1) dependency on FCS_CKM.4

FCS_COP.1/(1) is declared in the ST to model the ephemeral-symmetric keys derived from the ephemeral-asymmetric keys generated as part of the TLS session, whose purpose is to perform AES256-based message encryption.

In addition, session-ephemeral keys are removed whenever the session gets closed. OpenSSL, which is the one responsible for this communication channel, is configured by default to provide a session timeout after 300 seconds. Another way of closing a session is by the exchange of the message *close_notify* between the client and the server, mandatory on TLS sessions as specified at [RFC 5246].

FCS_COP.1/(3) dependency on FCS_CKM.4

FCS_COP.1/(3) is declared in the ST mainly to model message authentication based on the previously-generated asymmetric ephemeral keys.

Session-ephemeral keys are removed whenever the session gets closed. OpenSSL, which is the one responsible for this communication channel, is configured by default to provide a session timeout after 300 seconds. Another way of closing a session is by the exchange of the message *close_notify* between the client and the server, mandatory on TLS sessions as specified at [RFC 5246].

However, the first message on which the ephemeral public key of the server is shared with the client is sent signed by the server's disk-stored private key. The disk private key is imported during the preparation of the TOE as described in the security guidelines and does not change during the lifetime of the TOE. Therefore, it must be destroyed by the environment at the end of its lifetime, as described in the security guides.

FCS_COP.1/(2) dependency on FCS_CKM.4 and FCS_CKM.1

FCS_COP.1/(2) is declared in the ST to model the SHA384 algorithm used to ensure message integrity on TLS session. This cryptographic algorithm does not need any key/private data as input in order to generate an output. Therefore, no key generation nor destruction is needed.

6.3.4 SAR RATIONALE

The SARs were chosen according to the market expected evaluation assurance level for the TOE type.

Moreover, some extended SARs have been defined. They have been extracted from [PP_APP] due to the Security Functional Requirements included in this ST have been extracted from the Protection Profile and minimally adapted to the security functionalities implemented by the TOE. Then, the extended SARs from [PP_APP] have been included in this Security Target.

6.3.5 SAR DEPENDENCY RATIONALE

6.3.5.1 TABLE OF SAR DEPENDENCIES

SAR	Required	Fulfilled	Missing
ASE_CCL.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.1	ASE_INT.1, ASE_ECD.1, ASE_REQ.2 (hierarchically above ASE_REQ.1)	None
ASE_ECD.1	None	None	None
ASE_INT.1	None	None	None
ASE_OBJ.2	ASE_SPD.1	ASE_SPD.1	None
ASE_REQ.2	ASE_OBJ.2, ASE_ECD.1	ASE_OBJ.2, ASE_ECD.1	None
ASE_TSS.1	ASE_INT.1, ASE_REQ.1, ADV_FSP.1	ASE_INT.1, ASE_REQ.2 (hierarchically above ASE_REQ.1), ADV_FSP.2 (hierarchically above ADV_FSP.1)	None

SAR	Required	Fulfilled	Missing
ALC_CMC.2	ALC_CMS.1	ALC_CMS.2 (hierarchically above ALC_CMS.1)	None
ALC_CMS.2	None	None	None
ADV_FSP.2	ADV_TDS.1	ADV_TDS.1	None
AGD_OPE.1	ADV_FSP.1	ADV_FSP.2 (hierarchically above ADV_FSP.1)	None
AGD_PRE.1	None	None	None
ATE_IND.2	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1	ADV_FSP.2, AGD_OPE.1, AGD_PRE.1, ATE_COV.1, ATE_FUN.1	None
AVA_VAN.2	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	None
ADV_TDS.1	ADV_FSP.2	ADV_FSP.2	None
ASE_SPD.1	None	None	None
ALC_DEL.1	None	None	None
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1	ADV_FSP.2 (hierarchically above ADV_FSP.1), ADV_TDS.1	None
ATE_COV.1	ADV_FSP.2, ATE_FUN.1	ADV_FSP.2, ATE_FUN.1	None
ATE_FUN.1	ATE_COV.1	ATE_COV.1	None
AGD_OPE.1-PP13-GUIDANCE.1	ADV_FSP.1	ADV_FSP.1	None

SAR	Required	Fulfilled	Missing
AGD_PRE.1-PP13-GUIDANCE.1	None	None	None
ALC_CMC.2-PP13-GUIDANCE.1	ALC_CMS.1	ALC_CMS.1	None
ALC_CMS.2-PP13-GUIDANCE.1	None	None	None
ATE_IND.2-PP13-GUIDANCE.1	ADV_FSP.2 AGD_OPE.1 AGD_PRE.1 ATE_COV.1 ATE_FUN.1	ADV_FSP.2 AGD_OPE.1 AGD_PRE.1 ATE_COV.1 ATE_FUN.1	None
AVA_VAN.2-PP13-GUIDANCE.1	ADV_ARC.1 ADV_FSP.2 ADV_TDS.1 AGD_OPE.1 AGD_PRE.1	ADV_ARC.1 ADV_FSP.2 ADV_TDS.1 AGD_OPE.1 AGD_PRE.1	None

Table 7 SAR dependencies

7 TOE SUMMARY SPECIFICATION

7.1 CRYPTOGRAPHIC SUPPORT (FCS)

This section describes how the TOE meets each security functional requirement that belong to class FCS (defined in **[CC31R5P2]**) and that is listed in Section 6 of this ST.

The TOE uses third parties for cryptographic functionalities, and these third parties uses at their deepest level OpenSSL v1.1.1m and OpenSSL v1.1.1m + QUIC.

7.1.1 FCS_CKM_EXT.2 CRYPTOGRAPHIC KEY GENERATION

All administrative credentials data are stored using HMAC-SHA-256 as indicated at **FCS_COP.1/(4)**. It performs 1000 iterations and outputs a 256-bit strength key. Password-based derived keys are formed using a 128-bit salt that is randomly generated by the TOE's DRBG. The password is encoded in ASCII, concatenated with the salt and used as input to an OpenSSL module. Its output is scrambled and obfuscated through a group of byte-level operations.

7.1.2 FCS_CKM.1 CRYPTOGRAPHIC ASYMMETRIC KEY GENERATION

The TOE generates asymmetric keys in support of trusted communications. The TSF generates ECC keys using P-256 and P-384. These keys are generated in support of the ECDHE key establishment schemes that are used for TLS/HTTPS and TLS/MQTTv3.1.1 communications. These asymmetric ephemeral session keys are used to generate other symmetric keys with which the TOE performs message encryption.

7.1.3 FCS_CKM.2 CRYPTOGRAPHIC KEY ESTABLISHMENT

In order to meet this requirement, the TOE supports elliptic curve diffie-hellman key exchange. The CTR_DRBG (AES) is used for every random bit generation from the TOE in the key establishment process. DH Keys are generated using ECDHE from **[RFC 4492]**, Section 2. All supported TLSv1.2 ciphersuites use elliptic curves as the method of key establishment. The TSF presents secp256r1 and secp384r1 as the supported values in the Supported Groups extension and uses the same NIST curves for key establishment.

7.1.4 FCS_COP.1/(1) CRYPTOGRAPHIC OPERATION (ENCRYPTION/DECRYPTION)

The TOE provides symmetric encryption and decryption capabilities using AES algorithm with key size 256 bits in GCM mode. These AES algorithm modes are used in the secure channel between the TOE and the web browser in HTTPS and between the TOE and the external IT entities in MQTTv3.1.1 as part of the TLSv1.2 protocol.

7.1.5 FCS_COP.1/(2) CRYPTOGRAPHIC OPERATION (HASHING)

The TOE provides cryptographic hashing implementation using SHA-384 as specified in **[FIPS Pub 180-4]**, also meeting the **[ISO/IEC 10118-3:2004]**. The association of the hash function with other TSF cryptographic functions is described in the table below:

Cryptographic Functions	Hash Function
TLSv1.2 Integrity	SHA-384

7.1.6 FCS_COP.1/(3) CRYPTOGRAPHIC OPERATION (SIGNATURE GENERATION AND VERIFICATION)

The TOE implements RSA cryptography with key size of 3072 bits as specified in **[FIPS 186-4]** "Digital Signature Standard (DSS)". It is used for signature generation and verification of TLSv1.2 secure channel, giving authentication to the endpoints, used in the communication with the web browser via HTTPS and with external IT entities via MQTTv3.1.1.

7.1.7 FCS_COP.1/(4) CRYPTOGRAPHIC OPERATION (PASSWORDS)

The TOE perform keyed-hash message authentication HMAC-SHA-256 to store administrators', users' and SNMPv3's credentials securely.

7.1.8 FCS_RBG_EXT.1 AND FCS_RBG_EXT.2 EXTENDED: CRYPTOGRAPHIC OPERATION (RANDOM BIT GENERATION AND RANDOM BIT GENERATION FROM APPLICATION)

The TOE implements a deterministic random bit generator (DRBG) which is conformant to **[ISO/IEC 18031:2011]** using the DRBG mechanism CTR_DRBG (AES) as specified in **[SP800-90A]**, chap. 10.2.1. The entropy source is based on software (one internal noise source). Random numbers from the software noise source are only used for seeding the DRBG. The TOE sets a new seed using at least 256 bits entropy before generating random bits as cryptographic key. TSF uses CTR_DRBG (AES) to perform deterministic random bit generation. These random numbers are used for generating the pre master secret used in HTTPS/TLS sessions. The identified hash functions (SHA256) are allowed for CTR_DRBG (AES).

According to the entropy source, OpenSSL makes use of the USE_BCRYPTGENRANDOM function for random number generation. The entropy is obtained by considering the following parameters: running processes, threats, processor, selected window, modules, heap addresses, process identifier, memory usage and others. For more information on the entropy source used by OpenSSL, the analysis of the *rand_wind.c* file is encouraged, which will allow to understand in sufficient detail how the entropy source works when running OpenSSL on the Windows operating system.

OpenSSL performs a set of health tests to ensure the proper operation of the entropy source used. For more information on the tests used, it is recommended to access the *randtest.c* file in the OpenSSL library source code. Module users (the calling applications) shall use entropy sources that meet the security strength required for the random number generation mechanism as shown in [SP 800-90] *Table 3: Definitions for the CTR_DRBG.* This entropy is supplied by means of callback functions. Those functions must return an error if the minimum entropy strength cannot be met.

7.1.9 **FCS_CKM_EXT.1** CRYPTOGRAPHIC KEY GENERATION SERVICES

There are TLS-based secure communication channels which need asymmetric cryptographic keys. There are two different asymmetric cryptographic keys, first generated ones are originated following security guidelines and second ones are generated by the TOE, based on the first ones, for TLSv1.2 session. This SFR refers to the ephemeral asymmetric keys generated of the TLS session. These ephemeral asymmetric keys are used by the TOE to derive a shared secret between the ends of the communication for the encryption of the messages as described in **FCS_COP.1/(1)**.

7.1.10 **FCS_STO_EXT.1** STORAGE OF CREDENTIALS

User and administrator's passwords and SNMPv3 credentials are securely stored in the application installation directory in a non-plaintext form, both using HMAC-SHA-256 algorithm. These are the user and administrator’s authentication credentials and SNMPv3 templates credentials.

7.1.11 **FCS_HTTPS_EXT.1** HTTPS PROTOCOL

The TOE is managed via Web-based Management Interface (WebUI) rendered by one of the web browsers specified at 1.3.4. The TOE uses Nginx v 1.21.6 web server, configured for enforcing secure HTTPS **[RFC 2818]** connections. Nginx 1.21.6 gives TOE’s core functionality and secures the secure channel with the web browser through HTTPS. To achieve this configuration, external OpenSSL v.1.1.1 last version has to be used to generate asymmetric keys and TOE’s server certificates (this is covered as part of the TOE preparative procedures in **[AGD_PRE]**). The web server implements the TLSv1.2 protocol that protects administrators access through a secure HTTPS channel.

7.1.12 **FCS_TLSS_EXT.1** & **FCS_TLS_EXT.1** TLS SERVER PROTOCOL

The TOE denies all connection requests from TLS version 1.1 or older. Only TLSv1.2, **[RFC 5246]**, connections are supported. When a client requests an unsupported version, the TOE terminates the connection. The TOE negotiates key establishment using elliptic curves *secp256r1* and *secp384r1.* TLS ciphersuites are mentioned in **FCS_TLSS_EXT.1** definition and will be used in all TLS sessions. Nginx implements TLSv1.2 in order to establish a secure channel between the TOE and the browser, and NetinDS Agent uses TLSv1.2 when the secure MQTTv3.1.1 channel is established between the TOE and an external IT entity. **[AGD_PRE]** describes how to configure TLSv1.2 secure channel for both, MQTTv3.1.1 and HTTPS, scenarios.

The TOE denies the use of the following versions of TLS: SSL 2.0, SSL 3.0, TLS 1.0 and TLS 1.1

7.2 SECURITY MANAGEMENT (FMT)

7.2.1 **FMT_SMF.1** SPECIFICATION OF MANAGEMENT FUNCTIONS

The TOE provides all the capabilities necessary to securely manage the TOE. Security functions are managed through the use of artifacts. Templates can be edited in the repository configuration tab. These templates provide the possibility to configure the algorithms that will provide security in the communication with external IT entities. The management functionality provided by the TOE includes the following administrative functions:

- Ability to manage artifacts. Artifacts are pieces of code that enable the TOE to communicate with external IT entities through various protocols. In this case the two secure protocols evaluated are SNMPv3 and MQTTv3.1.1. Configuration Administrator can start and stop running artifacts, thus managing communication with external IT entities.

- Ability to manage communication channels, is the ability to configure the cryptographic functionality by SNMPv3 artifact's template configuration, where different encryption and hashing protocols can be selected (HMAC-SHA1 and AES-128-CFB), as well as the passwords used for the secure channel.

- Ability to perform user management, create new users and assign them a user role, delete users and change users and administrator passwords. The creation or modification of passwords requires a minimum level of complexity: Length of at least 8 characters, mixing numbers, uppercase, lowercase and special characters. It is necessary to add extra information for users such as username, mail address and, additionally, location.

7.2.2 **FMT_MEC_EXT.1** SUPPORTED CONFIGURATION MECHANISM

To configure the secure channels, the application user must follow the steps described in the security guidelines:
TLSv1.2 configuration is performed in configuration files and it has to be done during the installation process by following the security guidelines. While SNMPv3 configuration is performed in artifact templates within the TOE.

MQTTv3.1.1 configuration is done at <Installation path>\Netin\NDS-Kernel\repository\artifacts\netin-ds-drv-mqtt\.env and <Installation path>\Netin\NDS-Kernel\repository\artifacts\netin-ds-drv-mqtt\artifact.dna. HTTPS configuration is done at <Installation path>\Netin\NDS-WebUI\conf\servers\http-server.conf. The configuration of such files ends with a TLSv1.2 channel as described in **FCS_TLSS_EXT.1** for each new session.

SNMPv3 configuration is done in the "repository" tab from the application. This repository can only be accessed by the Configuration Administrator role. Configuration changes are saved to <installation path>\Netin\NDS-Kernel\repository\templates. Following SNMPv3 configuration as described in security guidelines, ends up with a SNMPv3 channel that ensures confidentiality, integrity and authentication using AES-128-CFB and HMAC-SHA1 cryptographic algorithms.

Default credentials must be changed first of all as described in the security guidelines. This configuration can only be done by User Administrator role. It is done in "Users" tab, changing username and password.

The credentials are securely stored within the MongoDB database of the TOE system, where all its data resides in the dedicated installation directory of TOE.

All configuration changes are saved at the installation directory.

7.2.3 **FMT_CFG_EXT.1** SECURE BY DEFAULT CONFIGURATION

The application is installed with three default users and passwords, each with one of the three roles. In the security guidelines, the user is prompted to change the default credentials, followed by the guidelines to do so. Application's directory integrity is protected from non-administrator users by applying read-only permissions to this directory.

7.3 TRUSTED PATH/CHANNELS (FTP)

7.3.1 **FTP_ITC.1/SNMPV3** INTER-TSF TRUSTED CHANNEL

The communication between the TOE and external IT entities (devices connected with the TOE through artifacts) can be done through a secure channel given by SNMPv3 that assures authentication and confidentiality of packets. SNMPv3 is set to authPriv security level in the security guidelines while external IT entities must be also configured to this security level and the same cryptographic algorithms by a trusted administrator. Authentication and integrity are provided by a string match of the username or community string and a password (password is sent in non-plaintext hashed with cryptographic algorithm HMAC-SHA-1), and confidentiality is provided with AES-128-CFB.

Both confidentiality and authentication need a password. These passwords are different from each other and users guide force them to follow complexity rules.

Note that HMAC-SHA-1 in the current use case is declared as a legacy mechanism by [SOG-IS] and [STIC-807].

**Legacy mechanisms that are deployed on a large scale, currently offer a security level of at least 100 bits and are considered to provide an acceptable short-term security, but should be phased out as soon as practical because they do no longer fully reflect the state of the art and suffer from some security assurance limitations as compared with recommended mechanisms. As a consequence, a validity period is defined for legacy mechanisms. Refer to [SOG-IS] section 1.1 for additional information.*

7.3.2 **FTP_DIT_EXT.1** PROTECTION OF DATA IN TRANSIT

Connections to NetinDS Web Interface are made using HTTPS, based on TLSv1.2, which provides the security algorithms mentioned in **FCS_TLSS_EXT.1** definition. This communication is done by default at TCP port 443.

Connections to external IT entities are made by NetinDS Agent using the secure protocol MQTTv3.1.1. MQTTv3.1.1 security is based on TLS, which provides the security algorithms mentioned in **FCS_TLSS_EXT.1** definition. This communication is done by default at TCP port 8883. The secure channel is used to make requests to, monitor and receive alerts from external IT devices that support MQTTv3.1.1.

7.4 USER DATA PROTECTION (FDP)

7.4.1 **FDP_DEC_EXT.1** ACCESS TO PLATFORM RESOURCES

The TOE only make use of platform hardware resources for making network connections and doesn't access any sensitive information repositories.

7.4.2 **FDP_NET_EXT.1** NETWORK COMMUNICATIONS

The application restricts its network communications to those that NetinDS Agent uses to discover external IT entities through DCP and ICMP protocols and, once the artifact templates are configured, communicate with them through SNMPv3 and MQTTv3.1.1.
The application also communicates with the web browser through Nginx using a secure HTTPS channel. Administrators and users make use of this secure channel to remotely administer the application.

7.4.3 **FDP_DAR_EXT.1** ENCRYPTION OF SENSITIVE APPLICATION DATA

The application manages the following sensitive data:

- Usernames. Are stored by the application at disk, located at C:\ProgramData directory. This data is not transmitted by the application. It is encrypted at full drive encryption level via BitLocker.

- Users' mail addresses. Are stored by the application at disk, located at C:\ProgramData directory. This data is not transmitted by the application. It is encrypted at full drive encryption level via BitLocker.

- Users' passwords. Are stored by the application at disk, located at C:\ProgramData directory. This data is transmitted from the web browser to NetinDS Server for authentication and is protected by an HTTPS channel. Passwords are saved in a non-plaintext form using HMAC-SHA-256.

- Authentication passwords for SNMPv3 are stored by the application at disk, located at C:\ProgramData directory. This data is transmitted from NetinDS Agent to external IT entities for authentication. It is protected by the cryptographic algorithms explained in section 7.1.10. These passwords are encrypted at full drive encryption level via BitLocker. In addition, these passwords are stored in a HMAC-SHA-256 format.

- Privacy passwords for SNMPv3 are stored by the application at disk, located at C:\ProgramData directory. This data is not transmitted by the application, it is used for data encryption by the cryptographic algorithms explained in section 7.1.10. These passwords are encrypted at full drive encryption level via BitLocker. In addition, these passwords are stored in a HMAC-SHA-256 format.

7.5 PRIVACY (FPR)

7.5.1 FPR_ANO_EXT.1 USER CONSENT FOR TRANSMISSION OF PERSONALLY IDENTIFIABLE INFORMATION

The application does not transmit any PII with other external IT entities.

7.6 PROTECTION OF THE TSF (FPT)

7.6.1 FPT_API_EXT.1 USE OF SUPPORTED SERVICES AND APIS

The TOE is designed to run on a general-purpose computer with Windows10 as the OS. The TOE uses only documented platform APIs. Appendix A.1 lists the APIs used by the TOE.

7.6.2 FPT_AEX_EXT.1 ANTI-EXPLOITATION CAPABILITIES

The TOE is composed of the .exe and .dll files listed below, along with their purpose and list of compilation flags related to FPT_AEX_EXT.1:

Purpose	Binary	NXCompact	GS	ASLR
Compression utility executable	compact.exe	Yes	Yes	No
Launcher NodeJS projects as Windows services executable	netinservicecontroller.exe	Yes	Yes	Yes
Uninstaller executable	uninstall_Netin.exe	Yes	Yes	No
Artifacts for MQTTv3.1.1 and SNMPv3 executable	node.exe	Yes	Yes	Yes

Mongo DB database service executable	mongod.exe	Yes	Yes	Yes
Redis Server executable	memurai.exe	Yes	Yes	No
Web server executable	nginx.exe	Yes	Yes	No

The flags with which each of them has been compiled can be found in the SFR definition.

The TOE implements several mechanisms to protect against exploitation. The application executables that meet the requirement, implement address space layout randomization through the use of the ASLR protection with compiler flags and relies on its underlying host platforms to perform memory mapping. The compiler flags are detailed below:

- */HIGHENTROPYVA* which specifies whether the executable image supports high-entropy 64-bit ASLR.
- */DYNAMICBASE* which specifies whether to generate an executable image that can be randomly rebased at load time by using the ASLR feature of Windows Operating Systems.

The application executables that meet the requirement, don't use both write and execution permissions on the same memory regions, this is assured by /NXCompact compiler flag. There is no situation where the TSF maps memory to an explicit address. They are compiled with stack overflow protection through the use of the /GS compiler flag. The TOE has a web-based front-end, based on HTML5 and CSS3. On the other hand, the TOE uses NodeJS v16.14.0 (which uses OpenSSLv1.1.1m + QUIC) in the implementation of the artifacts. HTML5, CSS3 and NodeJS are interpreted code to which compilation instructions is not applicable for this requirement. There are parts of the application code developed in Java, these are not applicable either.

7.6.3 **FPT_IDV_EXT.1** SOFTWARE IDENTIFICATION AND VERSIONS

The TOE is versioned using semver (Semantic Versioning) in the format x.y(.z) where x is the major version, y is the minor version, and the optional z is the patch version.

7.6.4 **FPT_TUD_EXT.1** AND **FPT_TUD_EXT.2** INTEGRITY FOR INSTALLATION AND UPDATE

The TOE is a standalone application that is not natively bundled as part of a host OS, but installer and upgrade packages are distributed as an independent software installer (.exe) that are digitally signed. The OS mechanisms are leveraged for verification of the digital signature of the package, signed by Netin Systems SL SHA256 certificate. The algorithm used to perform the signature is RSA-PCKSv1.5 with SHA256.

The application doesn’t allow to download, modify, replace or update its own binary code and only the code distributed with the installer is run and its uninstallation removes all its traces.

TOE updates are performed in compliance with the operational guidance.

The TOE server provides information about the installed TOE version.

The TOE is distributed with an uninstaller executable file called “uninstall.exe” which is responsible for uninstalling the application as described in the FPT_TUD_EXT.2.2 element.

8 ACRONYMS

The following table shows the acronyms used in this document.

Acronym	Meaning
PP	Protection Profile
CC	Common Criteria
TSFi	TSF Interface
OSP	Organisational Security Policies
EAL	Evaluation Assurance Level
ST	Security Target
IIoT	Industrial Internet of Things
HTML5	HyperText Markup Language version 5
CSS3	Cascading Style Sheets version 3
OT	Operational Technology
RSA	Rivest, Shamir and Adleman
SHA	Secure Hash Algorithm
GCM	Galois/Counter Mode
PII	Personally Identifiable Information
DCP	Discovery and Configuration Protocol
ICMP	Internet Control Message Protocol
SFR	Security Functional Requirement
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
DH	Diffie-Hellman
AES	Advanced Encryption Standard
TSF	TOE Security Functionality
DRBG	Deterministic Random Bit Generator
TLS	Transport Layer Security
TLSv1.2	Transport Layer Security version 1.2
MIB	Management Information Base
GSDML	General Station Description Markup Language
TCP	Transmission Control Protocol
MQTTv3.1.1	Message Queuing Telemetry Transport version 3.1.1
HTTPS	HyperText Transfer Protocol Secure
WebUI	Web User Interface
SNMPv3	Simple Network Management Protocol version 3
IT	Information Technology

Acronym	Meaning
OS	Operating System
TOE	Target of Evaluation
SFP	Security Function Policy

Table 8 Abbreviations

9 GLOSSARY OF TERMS

Term	Meaning
Augmentation	Addition of one or more requirement(s) to a package
Evaluation Assurance Level	Set of assurance requirements drawn from CC Part 3, representing a point on the CC predefined assurance scale, that form an assurance package
Operational Environment	Environment in which the TOE is operated
Protection Profile	Implementation-independent statement of security needs for a TOE type
Security Target	Implementation-dependent statement of security needs for a specific identified TOE
Target Of Evaluation	Set of software, firmware and/or hardware possibly accompanied by guidance

Table 9 *Glossary of terms*

10 DOCUMENT REFERENCES

The following table shows the acronyms used in this document.

Reference	Document
[CC31R5P1]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 1: Introduction and general model
[CC31R5P2]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 2: Security functional components
[CC31R5P3]	Common Criteria for Information Technology Security Evaluation, Version 3.1, Revision 5, Part 3: Security assurance components
[CEM31R5]	Common Criteria Evaluation methodology, Version 3.1, Revision 5
[FIPS 186-4]	National Institute of Standards and Technology, Digital Signature Standard (DSS), Federal Information Processing Standards Publication FIPS PUB 186-4, July 2013
[RFC 4346]	The Transport Layer Security (TLS) Protocol Version 1.1
[RFC 5246]	The Transport Layer Security (TLS) Protocol Version 1.2
[ISO/IEC 18031:2011]	Information technology -- Security techniques -- Random bit generation
[ISO/IEC 10118-3:2004]	Information technology -- Security techniques -- Hash-functions
[ISO 19772]	Information technology — Security techniques — Authenticated encryption
[RFC 5288]	AES Galois Counter Mode (GCM) Cipher Suites for TLS
[FIPS Pub 180-3]	Secure Hash Standard (SHS)
[SP800-90A]	Recommendation for Random Number Generation Using Deterministic Random Bit Generator
[RFC 2818]	HTTP Over TLS
[AGD_OPE]	NetinDS Operational User Guidance, version 0.12
[AGD_PRE]	NetinDS Preparative Guidance, version 0.12
[RFC 2570]	Introduction to Version 3 of the Internet-standard Network Management Framework
[MQTT Version 3.1.1]	MQTT Version 3.1.1 OASIS Standard 09 October 2014
[PP_APP]	Protection Profile for Application Software version 1.3
[RFC 4492]	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)
[FIPS Pub 198-1]	The Keyed-Hash Message Authentication Code (HMAC)
[FIPS Pub 180-4]	Secure Hash Standard
[NIST SP 800-132]	Recommendation for Password-Based Key Derivation: Part 1: Storage Applications

Reference	Document
[SP 800-90]	NIST Special Publication 800-90A Revision 1 Recommendation for Random Number Generation Using Deterministic Random Bit Generators
[SOG-IS]	SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms. Version 1.2. January 2020
[STIC-807]	Guía de Seguridad de las TIC CCN-STIC 807. Criptología de empleo en el Esquema Nacional de Seguridad. May 2022

Table 10 *List of document references*

11 APPENDICES

11.1 APPENDIX A - TOE USAGE OF THIRD-PARTY COMPONENTS

This Appendix lists the platform APIs that are used by the TOE.

11.1.1 A.1 PLATFORM APIS

Listed below are the platform APIs used by the TOE. Note that these APIs do not necessarily relate to the TOE functionality claimed in the Security Target; however, since they are bundled with the product itself they are disclosed since a vulnerability in outside the logical boundary of the product could still present an exploitable vulnerability.

- FindWindowA
- SendMessageA
- SetStdHandle
- CloseHandle
- UnmapViewOfFile
- MapViewOfFile
- CreateFileMappingA
- GetCurrentThreadId
- GetLastError
- HeapFree
- HeapAlloc
- EnterCriticalSection
- LeaveCriticalSection
- GetCommandLineA
- HeapSetInformation
- WriteFile
- WideCharToMultiByte
- GetConsoleCP
- GetConsoleMode

- UnhandledExceptionFilter
- SetUnhandledExceptionFilter
- IsDebuggerPresent
- EncodePointer
- DecodePointer
- TerminateProcess
- GetCurrentProcess
- HeapCreate
- MultiByteToWideChar
- ReadFile
- GetProcAddress
- GetModuleHandleW
- ExitProcess
- GetStdHandle
- GetModuleFileNameW
- SetHandleCount
- InitializeCriticalSectionAndSpinCount
- GetFileType
- GetStartupInfoW
- DeleteCriticalSection
- Sleep
- IsProcessorFeaturePresent
- GetModuleFileNameA
- FreeEnvironmentStringsW
- GetEnvironmentStringsW
- TlsAlloc
- TlsGetValue

- TlsSetValue
- TlsFree
- InterlockedIncrement
- SetLastError
- InterlockedDecrement
- QueryPerformanceCounter
- GetTickCount
- GetCurrentProcessId
- GetSystemTimeAsFileTime
- SetFilePointer
- WriteConsoleW
- FlushFileBuffers
- RtlUnwind
- GetCPInfo
- GetACP
- GetOEMCP
- IsValidCodePage
- LoadLibraryW
- HeapReAlloc
- CreateFileW
- GetStringTypeW
- LCMapStringW

11.1 APPENDIX B – ENTROPY DOCUMENTATION AND ASSESSMENT

This appendix describes the required supplementary information for the entropy source used by the TOE. The documentation of the entropy source should be detailed enough that, after reading, the evaluator will thoroughly understand the entropy source and why it can be relied upon to provide sufficient entropy. This documentation should include multiple detailed sections: design description, entropy justification, operating conditions, and health testing.

11.1.1 DESIGN DESCRIPTION

Documentation shall include the design of the entropy source as a whole, including the interaction of all entropy source components. Any information that can be shared regarding the design should also be included for any third-party entropy sources that are included in the product.

The documentation will describe the operation of the entropy source to include, how entropy is produced, and how unprocessed (raw) data can be obtained from within the entropy source for testing purposes. The documentation should walk through the entropy source design indicating where the entropy comes from, where the entropy output is passed next, any post-processing of the raw outputs (hash, XOR, etc.), if/where it is stored, and finally, how it is output from the entropy source. Any conditions placed on the process (e.g., blocking) should also be described in the entropy source design. Diagrams and examples are encouraged.

This design must also include a description of the content of the security boundary of the entropy source and a description of how the security boundary ensures that an adversary outside the boundary cannot affect the entropy rate. If implemented, the design description shall include a description of how third-party applications can add entropy to the RBG. A description of any RBG state saving between power-off and power-on shall be included.

11.1.2 ENTROPY JUSTIFICATION

There should be a technical argument for where the unpredictability in the source comes from and why there is confidence in the entropy source delivering sufficient entropy for the uses made of the RBG output (by this particular TOE). This argument will include a description of the expected min-entropy rate (i.e. the minimum entropy (in bits) per bit or byte of source data) and explain that sufficient entropy is going into the TOE randomizer seeding process. This discussion will be part of a justification for why the entropy source can be relied upon to produce bits with entropy.

The amount of information necessary to justify the expected min-entropy rate depends on the type of entropy source included in the product.

For developer provided entropy sources, in order to justify the min-entropy rate, it is expected that a large number of raw source bits will be collected, statistical tests will be performed, and the min-entropy rate determined from the statistical tests. While no particular statistical tests are required at this time, it is expected that some testing is necessary in order to determine the amount of min-entropy in each output.

For third party provided entropy sources, in which the TOE vendor has limited access to the design and raw entropy data of the source, the documentation will indicate an estimate of the amount of min-entropy obtained from this third-party source. It is acceptable for the vendor to “assume” an amount of min-entropy, however, this assumption must be clearly stated in the documentation provided. In particular, the min-entropy estimate must be specified and the assumption included in the ST. Regardless of type of entropy source, the justification will also include how the DRBG is initialized with the entropy stated in the ST, for example by verifying that the min-entropy rate is multiplied by the amount of source data used to seed the DRBG or that the rate of entropy expected based on the amount of source data is explicitly stated and compared to the statistical rate. If the amount of source data used to seed the DRBG is not clear or the calculated rate is not explicitly related to the seed, the documentation will not be considered complete.

The entropy justification shall not include any data added from any third-party application or from any state saving between restarts

11.1.3 OPERATING CONDITIONS

The entropy rate may be affected by conditions outside the control of the entropy source itself. For example, voltage, frequency, temperature, and elapsed time after power-on are just a few of the factors that may affect the operation of the entropy source. As such, documentation will also include the range of operating conditions under which the entropy source is expected to generate random data. It will clearly describe the measures that have been taken in the system design to ensure the entropy source continues to operate under those conditions. Similarly, documentation shall describe the conditions under which the entropy source is known to malfunction or become inconsistent. Methods used to detect failure or degradation of the source shall be included.

11.1.4 HEALTH TESTING

More specifically, all entropy source health tests and their rationale will be documented. This will include a description of the health tests, the rate and conditions under which each health test is performed (e.g., at startup, continuously, or on-demand), the expected results for each health test, and rationale indicating why each test is believed to be appropriate for detecting one or more failures in the entropy source

11.2 APPENDIX C – CLARIFICATION TO THE ENTROPY DOCUMENTATION AND ASSESSMENT ANNEX

The generation of random bits is vital for key generation and the security strength of our cryptosystems. During analysis of different products, discoveries of weak random values are common, making the remaining security measures irrelevant. There has been significant research into generating pseudorandom bits using a Deterministic Random Bit Generator (DRBG) and an unknown seed value, but ensuring that unknown seed is truly ‘unknown’ is not as well documented. As a result, entropy implementations traditionally have not been tested to ensure that bits with suitable entropy are input into various DRBG implementations.

Given that there is now an effort by NIST to describe entropy sources and to provide tests that can be used to validate the quality of an entropy source, it is appropriate for CC evaluations to move in that direction as well. Since this is still a new concept for both evaluators and for the vendors, a staggered approach being used by NIAP to bring all parties to a common understanding so that the common entropy testing is less arduous at the outset. The first step is a documentation exercise, commonly referred to as the “Annex D” requirement or the Entropy Assessment Report (EAR).

The goal of this requirement is to get vendors, evaluators, and validators thinking about the entropy and Random Bit Generation (RBG) implementation. In the short timeframe that this has been a requirement, the resulting reports have varied widely despite the fairly detailed guidance given in the assurance activity. The understanding of entropy between vendors varies - some vendors trust third-party sources they do not fully comprehend. Others have an in-depth understanding and are easily able to provide rationale regarding the security of their products. A small subset has identified problems with their implementations when putting together this information, and has instituted updates to fix the issues. While it may seem like little progress, all of these (acknowledgement of current lack of understanding, practice in rationale/justification submission or data testing, and mitigation of problems) are a significant step forward in ensuring quality entropy.

The US Scheme has developed a fairly standard method for evaluating the entropy documentation provided by the vendors and has identified three major types of entropy sources. Below, for information, are some examples of how the US Scheme reviews the EAR for each of these entropy source types. The information provided within this document is meant to provide general guidance on entropy reporting and should not be considered a “check-list” for successful EAR documentation. Questions on this topic should be directed to the project validator or NIAP.

Software Sources

A number of vendors have submitted entropy documentation for software sources. For such vendors we examine the documentation to verify that the entropy is described completely, from the raw noise source to the input to the DRBG. We verify that the vendor has correctly described what the raw entropy is (i.e., before any conditioning functions such as hashes, mixing functions, or shift registers) and has described how this raw entropy is collected for statistical tests used to justify the entropy claim. If the vendor has not correctly identified or described collecting raw entropy, updated documentation must be resubmitted and reviewed before the product is accepted into evaluation.

Self-Provided Hardware Source

A vendor could provide entropy documentation indicating that the product provides its own hardware source. We have, to date, seen one such report, and have found it appropriate to treat such sources in the same manner as we do software sources.

Third-Party Source

A few vendors have submitted entropy documentation for third-party sources. Unfortunately, due to the limited access to the design and raw entropy data of these third-party sources, the vendor is not able to test the raw entropy source and sometimes is not even able to fully describe the source. Any information that can be shared regarding the design should be included. At a minimum, the documentation must indicate an estimate of the amount of entropy obtained from a third-party source. We generally allow the vendor to “assume” an amount of entropy from the third-party source; however, this assumption must be clearly stated in the documentation provided, the expected amount of entropy must be specified, and a related entropy assumption needs to be made in the Security Target (ST). We still expect a full description of the processing of the output of the third-party source up to and including the seeding of the DRBG implemented in the TOE. Given the assumption, this description must indicate that the DRBG is seeded with the appropriate amount of entropy stated in the ST.

In the future, we intend to require that all third-party sources have been evaluated themselves. To that end, a “platform-based DRBG” source option is included as part of the FCS_RBG_EXT1.2 requirement. Due to the limited number of evaluated platforms, Entropy Assessment Reports for unevaluated third-party sources are still being accepted and treated in the manner described above.

Common Problematic Areas

Regardless of the type of entropy source claimed, there are common areas where EARs often fall short, requiring the documentation to be resubmitted for NIAP review prior to acceptance into evaluation. These are outlined below in order to offer some additional guidance.

Saved State

The capability to add the state of the RBG saved at power-off to use as input to the RBG, prevents an RBG that is slow to gather entropy, from producing the same output regularly and across reboots. This is an important feature for some RBGs so enough variation is introduced such that the initial RBG values are not predictable and exploitable. However, since there is no guarantee of the protections provided when the state is stored (and no requirement for any such protection), it must be assumed that the state is ‘known.’ Therefore, any saved state is not considered to contribute entropy to the seeding of the RBG in order to meet FCS_RBG_EXT.1.2.

Seeding

We expect the vendors to collect a large number of raw source bits, perform statistical tests, and from the statistical tests determine a rate of entropy (i.e. the minimum entropy (in bits) per bit or byte of source data). While no particular statistical tests are required, it is expected that some testing is necessary in order to determine the amount of entropy in each output. In order to verity that the DRBG is initialized with the entropy stated in the ST, we then verify that this rate is multiplied by the amount of source data used to seed the DRBG or that the rate of entropy expected based on the amount of source data is explicitly stated and compared to the statistical rate. If the amount of source data used to seed the DRBG is not clear or the calculated rate is not explicitly related to the seed, the vendor is required to resubmit documentation before the assurance activity for FCS_RBG_EXT.1.2 is considered acceptable.

Raw Samples

Unfortunately, it is sometimes the case that a vendor does not have access to the raw data for testing purposes, and the only data seen has already been processed by some conditioning function. This is especially true for products that use third-party sources; limited access is common. It is important to stress the fact that running statistical tests on conditioned data does not produce valid entropy results. Generally, we would not accept an estimate using conditioned data. At this time, if the raw source is unavailable, it must be stated along with a clear statement of the amount of entropy expected from the conditioned source. When testing eventually moves to the verification of the entropy estimate, lack of access to raw data will no longer be acceptable.