Mediatek Trusted Execution Environment (M-TEE) hypervisor isolation platform security target

Date: 2022-12-22

Created by

Change History

Version	Date	Author	Comment
0.1	2021/10/18	Mediatek Inc.	Initial version
0.2	2021/10/25	Mediatek Inc.	Updated as per MTK comments
0.3	2021/11/03	Mediatek Inc.	Updated as per MTK comments
0.4	2021/11/04	Mediatek Inc.	Included guidance's as per MTK comments
0.5	2021/11/10	Mediatek Inc.	Included guidance's as per MTK comments
0.6	2021/11/17	Mediatek Inc.	Updated reference for TOE and guidance's
0.7	2021/12/13	Mediatek Inc.	Clarification of concepts
0.8	2022/02/09	Mediatek Inc.	Change to virtualization server PP
0.9	2022/03/01	Mediatek Inc.	Updated SFRs
0.91	2022/03/16	Mediatek Inc.	Adjusted TOE scope
0.92	2022/03/25	Mediatek Inc.	Adjusted TOE physical scope and provided further information about isolation in TSS
0.93	2022/03/31	Mediatek Inc.	Addressed comments from the evaluation laboratory.
0.94	2022/04/12	Mediatek Inc.	Addressed evaluation lab comments.
0.95	2022/05/03	Mediatek Inc.	Addressed evaluation lab comments.
0.96	2022/05/11	Mediatek Inc.	Changes in section 1.3.4
0.97	2022/05/27	Mediatek Inc.	Minor changes
0.98	2022/06/01	Mediatek Inc.	Changes in TOE version
0.99	2022/06/03	Mediatek Inc.	Minor changes
1.0	2022/07/08	Mediatek Inc.	Changes after EM1
1.1	2022/07/12	Mediatek Inc.	Changes to address evaluator comments
1.2	2022/08/03	Mediatek Inc.	Changes in hardware identification and TOE build dates
1.3	2022/08/12	Mediatek Inc.	Minor changes
1.4	2022/09/05	Mediatek Inc.	Changes after EM2
1.5	2022/10/03	Mediatek Inc.	Minor changes
1.6	2022/10/27	Mediatek Inc.	Added new SFR, threat and asset
1.7	2022/11/04	Mediatek Inc.	Minor changes
1.8	2022/12/07	Mediatek Inc.	Changes in sections 1.3.3, 1.3.4, 1.4.3; Updated tables 1 and 2;
1.9	2022/12/09	Mediatek Inc.	Updated figure 5
1.91	2022/12/22	Mediatek Inc.	Changes in table 4

Table of contents

1 ST INTRODUCTION

1.1 ST REFERENCE

Title: Mediatek Trusted Execution Environment (M-TEE) hypervisor isolation platform security target

Version: v1.91

Author: MediaTek Inc.

Evaluation Lab: Riscure BV

Date of publication: 2022-12-22

1.2 TOE REFERENCE

TOE Name: MediaTek Trusted Execution Environment (M-TEE) hypervisor isolation platform

TOE Developer: MediaTek Inc.

TOE Version: v. 1.0

TOE version 1.0 is the one that comprises the TOE items listed in *Table 1*.

From the perspective of the integrator and HA developer, the software parts of the TOE are uniquely identified by the identifier and version number (in format *<identifier>: <version>*) and built dates. Moreover, each of these software parts are distributed as prebuilt libraries whose integrity is statically verified using SHA256 digest as detailed in section 1.4.3. Then, on runtime, software parts can be identified by version and build string.

The following items compose the TOE release:

Type	Item	Identifier and version	Build date
TOE Software	**M-TEE Framework**	mTEE_SDK: 2.2.2.003.S0MP1_GZ	18:07:19 Mar 10 2022
	M-TEE Services HA	KERNEL_SRV_HA:2.0.001.S0MP1_GZ	18:07:24 Mar 10 2022

	Echo HA	ECHO_HA:2.0.002.S0MP1_GZ	18:07:28 Mar 10 2022
	Testing HA	GZ-TEST_HA:2.0.000.S0MP1_GZ	18:07:37 Mar 10 2022
TOE Hardware	**Stage 2 MMU**	The versions of the hardware models are listed in *Table 2*.	N/A

Table 1 - Reference of the items of the TOE

In the remainder of this document the term TOE refers to the M-TEE and the terms TOE and M-TEE are used interchangeably. The term TEE used in this document refers to the standard definition of TEE in **[GPD_SPE_009]** and is not considered as the TOE in this document.

1.3 TOE OVERVIEW

1.3.1 INTRODUCTION

The MediaTek Trusted Execution Environment (M-TEE) hypervisor isolation platform (the TOE) is the MediaTek isolation mechanism for a virtualized TEE-enabled environment which is inspired by the standard TEE architecture described in the GlobalPlatform TEE system architecture specification **[GPD_SPE_009]**.

The TOE runs in an environment which implements the standard TEE extension architecture from GlobalPlatform specification (one TEE for one CPU core) and a multiple TEE-comparable system architecture.

In this scenario, the typical environment that includes the REE (with CAs) and TEE (with TAs) is modified to include a virtualization layer that allows to run multiple guest operating systems (which run applications called HAs or Hypervisor Applications).

In particular, the TOE designed by Mediatek works in an evolution of the GP-defined TEE architecture, in which the *normal world* is divided in three isolated virtual domains that run at the same time as the Trusted Execution Environment. This implies the extension of a two-domain standard TEE architecture into a multiple-domain M-TEE architecture.

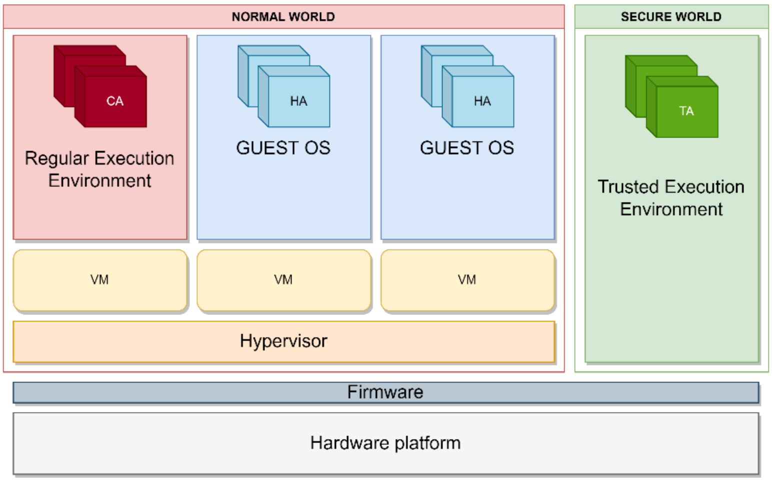

Figure 1 – Mediatek virtualization environment

In the traditional TEE environment or traditional TEE extensions in which Trusted Applications run provides a high level of security; however, these applications have severe performance constraints and demand heavy system resource.

Additionally, using a rich environment implies completely relying on the security capabilities of the operating system without the support of security measures given by the hardware platform. This non-secure environment could give better performance at the cost of exposing the system to external threats.

The environment designed by MediaTek provides the following capabilities:

- Extension of the architecture to run several operating systems and applications to provide high performance services.
- Extension of the isolation capability given by the platform, to complement the security given by the operating systems.

The Guest Operating Systems and Hypervisor Applications (HAs) are used to work together with the standard TAs to complement their functionality with the flexibility of supporting wide range of functionalities while maintaining the security level of a standard TEE with a cost-effective solution.

The TOE includes a subset of components that runs within this architecture in a closed configuration, where the Guest OS are two defined virtual machines: Nebula VM and Trusty VM.

The evaluated configuration and scope of the TOE are described in section 1.4.1.

1.3.2 TOE TYPE

The TOE is the MediaTek Trusted Execution Environment (M-TEE) Hypervisor isolation platform. It supports the execution of MediaTek hypervisor applications (HA) in Trusty VM, in a secured manner isolated from any applications running in REE, between them (isolating different HAs in Trusty) and isolated from the REE operating system. The TOE is integrated into a system-on-chip (SoC) and utilizes the underlying hardware platform of the SoC as outlined in more detail in the description of the physical scope of the TOE.

The TOE allows HAs to be loaded and installed post-issuance (in the OEM phase of the M-TEE-enabled device, not in the scope) and offers isolation between virtual domains (between REE and Trusty VM, between REE and Nebula VM, and between Trusty VM and Nebula VM) and access control between the components between Trusty HAs and those elements they attempt to access, according to the isolation mechanism.

To do this, the TOE incorporates the required hardware modules of GenieZone Hypervisor that seamlessly interacts with the virtualized operating systems to manage resources shared between them. In particular, the TOE operates with the highest (hypervisor) privileges (EL2) within the REE.

1.3.3 TOE USAGE & MAJOR SECURITY FEATURES

The TOE consists of the TOE-related software executing on a virtualized platform and one hypervisor hardware module which enforces the isolation between the components.

The TOE security functionalities in the scope of the evaluation that are available in the end user phase are:

- Isolation, covering:
    - Isolation between REE VM and Nebula VM.
    - Isolation between REE VM and Trusty VM, and between REE VM and HAs in Trusty.
    - Isolation between different HAs in Trusty.
    - Isolation between Nebula VM and Trusty VM
- Access control, constraining VMs (Nebula and Trusty) direct access to hardware and physical platform resources.

Figure 5 – Scope of the TOE can be referred to in order to illustrate the different components mentioned in these functionalities.

This security functionality defines the logical boundary of the TOE. The interfaces of this boundary consist of the software external interface and the hardware external interface, introduced in section *1.4 TOE Description* of this document.

The TOE comprises the following firmware, software and hardware used to provide the security functionality:

Software:

- Belonging to TRUSTY (M-TEE):
  - The **prebuilt HAs**: all of them installed in the Trusty virtual machine. **M-TEE Services HA**, **Echo HA** and **Testing HA** installed by default in the virtual machine, which provides security services to the Client Applications used by the end user. The Testing HA and Echo HA are disabled by the user (the OEM) before the delivery to the end-user.
  - The **M-TEE Framework**, which provides the interface so that the HAs installed in Trusty can interact with the virtualization system and provide the services.

Hardware:

- The Stage 2 MMU hardware module of GenieZone Hypervisor

Documentation:

- The guidance's for the secure usage of the M-TEE after delivery described in section **TOE Physical Scope**.

1.3.4 NON-TOE HARDWARE/SOFTWARE/FIRMWARE

The TOE is intended to be deployed in an environment having the following components which provides communication between them and which fulfills the security objectives for the operational environment described in section 4.2.

The TOE depends on the following non-TOE software, hardware and firmware components for proper functioning:

- **The Regular Execution Environment**

  intended as the operating system together with their applications running in the virtual machine (virtual machine considered as part of the hypervisor), including the Client Applications (CAs). The TOE can communicate with this operating system and the CAs to request rich services or use these components to indirectly access trusted services.

- **The Nebula OS and HAs in Nebula**

  The Nebula virtual machine is included in the required operational environment to interact with the REE and the TEE. This virtual machine includes an operating system (Nebula OS), could contain hypervisor applications (HAs) and other M-TEE related software such as libraries for communication.

  This Nebula OS, the HAs (preinstalled or not) and the related software are considered as TOE environment.

- **The 3rd party hypervisor applications (HAs)** the HAs deployed by the OEM after delivery of the TOE, usually to extend the services provided by Nebula (not in TOE scope) and Trusty, are not included in the scope. Even if these applications can use the isolation services provided by the TSF, these applications themselves are considered out of the scope.

- **The Trusted Execution Environment**

  Intended as the trusted operating system and all of its software components, including Trusted Applications (TAs) to which the TOE can communicate in order to request trusted services.

- **The GenieZone hypervisor**

  All hardware and software components of the GenieZone hypervisor apart from the *Stage 2 MMU.* These components constitute the VMM and are used for supporting the virtualization services in the non-secure world, the secure boot and the communication with the other system components.

  GenieZone Hypervisor is distributed together with the TOE in the same binary image as the M-TEE Framework TOE component (*mtk_armv8_el2-kernel.*a). The identification of the hypervisor is done through the version and build date of its logical components, as shown below:

Name	Identifier with version	Build
GZ_CORE_hypervisor	GZ_CORE_hypervisor: 3.2.0.039.S0MP1_GZ	18:07:16 Mar 10 2022
GZ_hypervisor	GZ_hypervisor: 3.2.0.039.S0MP1_GZ	18:08:05 Mar 10 2022

Table 2 GenieZone Hypervisor components

- **The Mediatek Platform**

  All the hardware and firmware (ATF firmware) of the TrustZone-based Mediatek platform used to provide isolation and communication between the secure and non-secure execution environments. In particular the following series are intended to support the TOE: MT67XX/MT68XX/MT69XX/MT81XX/MT86XX/MT87XX

- **Preloader**

  This component is located in the hardware platform, and is loaded and verified by the BROM. It is responsible for loading and verifying the following images: AFT, TEE, M-TEE and

LK (Little Kernel). In addition, it checks the authenticity and integrity of each image, using RSA-2048 and SHA-256.

The software part of the TOE is delivered to the user (OEM integrator) in the form of binary image, though the hardware part is delivered inside the final SoC. The software contains three preloaded HAs (M-TEE Services HA, Echo HA and Testing HA). The OEM is responsible for disabling Echo HA and Testing HA before the final delivery to the end user by following the guidance's in section 1.4.3 and the adequate usage of M-TEE Services HA when installing 3rd party HAs.

1.4 TOE DESCRIPTION

1.4.1 INTRODUCTION

A generic view of the Mediatek architecture

Common user applications usually operate under an operating system which usually provides rich functionality and high flexibility, which is commonly referred as REE (Rich Execution Environment).

In this rich environment, the security functionality relies in the operating system, thus the platform has no additional security capabilities to protect the applications and therefore, they are exposed to a wide range of security threats.

To mitigate these threats, an environment can be set to provide a set of trusted applications to be executed securely in a specialized execution environment which includes hardware-based security capabilities, known as the Trusted execution Environment (TEE).

The Trusted Execution Environment (TEE) is designed to reside alongside the REE and provides a safe area to protect asset and to execute trusted applications.

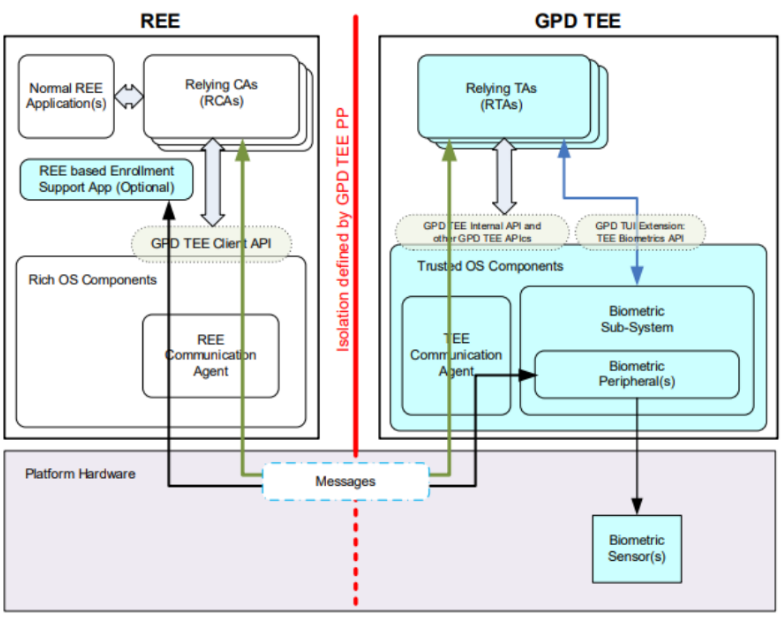

Figure 2 – Traditional GP TEE architecture

Such TEE architecture has been offered by ARM and has been widely accepted with the commercial name of *TrustZone*, which is supported by several standards (external and internal API) to allow that existing TAs can be easily ported to TEE. Additionally, the GlobalPlatform TEE standard further extends the basic TEE architecture (i.e. one TEE for one CPU core) to multiple-TEE system architecture.

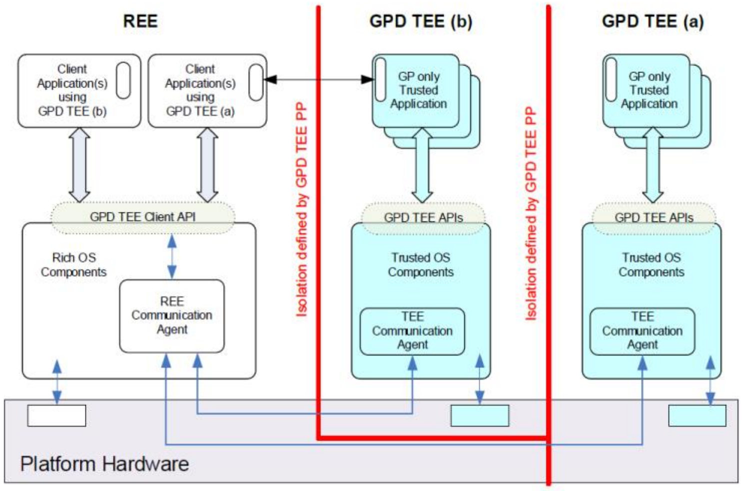

Figure 3 – Extension of the traditional GP TEE architecture with two TEE

Such an extension addresses a very important trend in the Trusted Applications: the TAs have a wide range of security requirement, system resource requirement (memory, CPU and acceleration engine), and performance requirement. For example, a Trusted Application designed for IoT, due to the nature of its intended service, can be lightened and can be satisfied by the basic TEE architecture with one TEE running on one CPU core. In contrast, an AI-based smart TA (such as face recognition for mobile payment) requires multiple CPU cores, large amount of memory, and real-time response.

In this regard, the traditional basic TEE architecture needs to be extended to support the future resource-intensive smart Trusted Applications, which has a direct impact on costs if the extension is made using the traditional TEE hardware-based architecture.

In this sense, the TOE designed by Mediatek provides an implementation of such an extension based in a virtualization platform to give the user the following security capabilities:

- The extension of the two-domain architecture “Normal/Secure” worlds into several security domains to allow the execution of several and simultaneous virtualized operating systems together with the traditional REE and TEE.
- The extension of the hardware-based isolation capability of a TEE-enabled platform, which is applied to the new virtualized environment to provide platform-based security in the new security domains.

Therefore, these environment covers the wide range of resource-demanding TAs with increased complexity and diversity in a third-party operating system, in a cost-effective way and without sacrificing protection capabilities given by a trusted platform.

The following figure provides a high-level view of the general architecture of a TOE-enabled device:

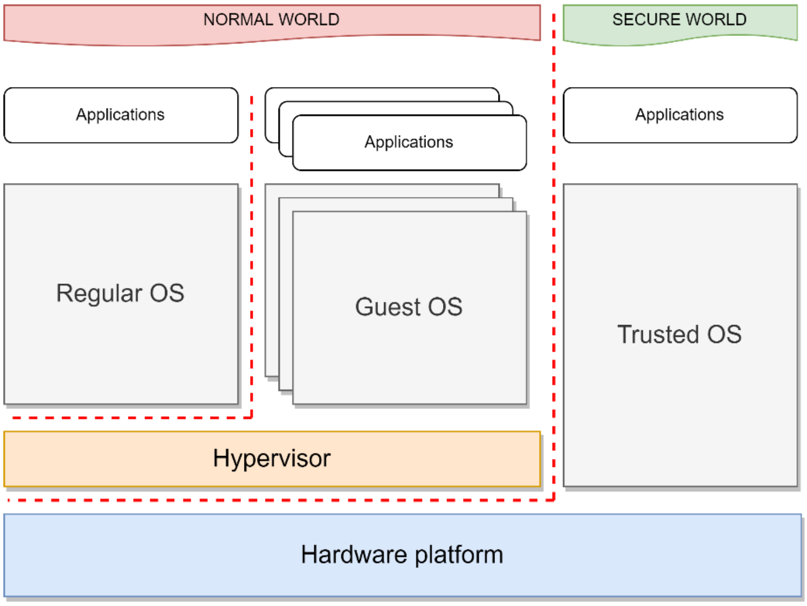

Figure 4 – TOE TrustZone-based architecture

The architecture in which the TOE relies is based in the ARM TrustZone architecture in which the hardware platform divides the execution space into *Normal World* and *Secure World*. In this architecture, the enabling of both worlds and the isolation of them is managed by the hardware platform.

The particular system designed by Mediatek includes a new isolation barrier in the *Normal World* by means of a hypervisor in which the TOE is hosted, which allow the execution of multiple guest operating systems together with the Trusted OS and Regular OS.

In the final architecture, there exits two isolation barriers:

(a) One driven by the hardware platform, which divides the environment between Normal world and secure world.
(b) One driven by the hypervisor, which divides the Normal world into several domains, one of them for the Regular OS and others for security services.

Using this architecture, the TOE and its environment are defined for support and enable processing sensitive data for light-weight applications (for example IoT applications) as well as resource-intensive smart applications (for example face-recognition based mobile payment) in scenarios where they require more flexibility and performance than the standard Trusted Execution Environment can offer.

It usually can be distinguished the following main parts and its components in this system from a generic point of view:

- The Regular Execution Environment:

- A Regular OS running in the virtual machine, as the main operating system but less-privileged. In charge of providing services to the applications running on it.
- Client Applications, which use dedicated APIs to access the secure services offered by the TAs running on the TEE;
- The APIs, to communicate the OS with the rest of the components of the system.

▪ A set of several Guest Operating Systems (each of them running in a separate virtual machine) each of them having the following components:

- The Guest OS running in the virtual machine, in charge of providing services to the applications running in a virtual machine.
- The Guest OS Applications (HAs) that run in the Guest OS and access its services through the APIs.
- The APIs, to communicate the OS with the rest of the components of the system.
- The virtualization components (virtual machines and hardware) providing communication services with the other components of the system.

▪ The Trusted Execution Environment:

- The Trusted OS as a background operating system which is not directly accessible by the user and having the most privileged level of operation.
- The Trusted Applications that run on the TEE and access its services through the APIs;
- The APIs, to communicate the OS with the rest of the components of the system.

Evaluated configuration

The *Figure 5* is a generic view of the architecture in which the TOE is based, however the TOE described in the current security target considers a much more restricted and precise environment in which the TOE is deployed. This environment includes the REE, a fixed set of two guest virtual machines (Nebula and Trusty) and the TEE, together with the virtualization and underlying platforms.

The following picture illustrates this particularized environment where the components included in the scope have been highlighted:

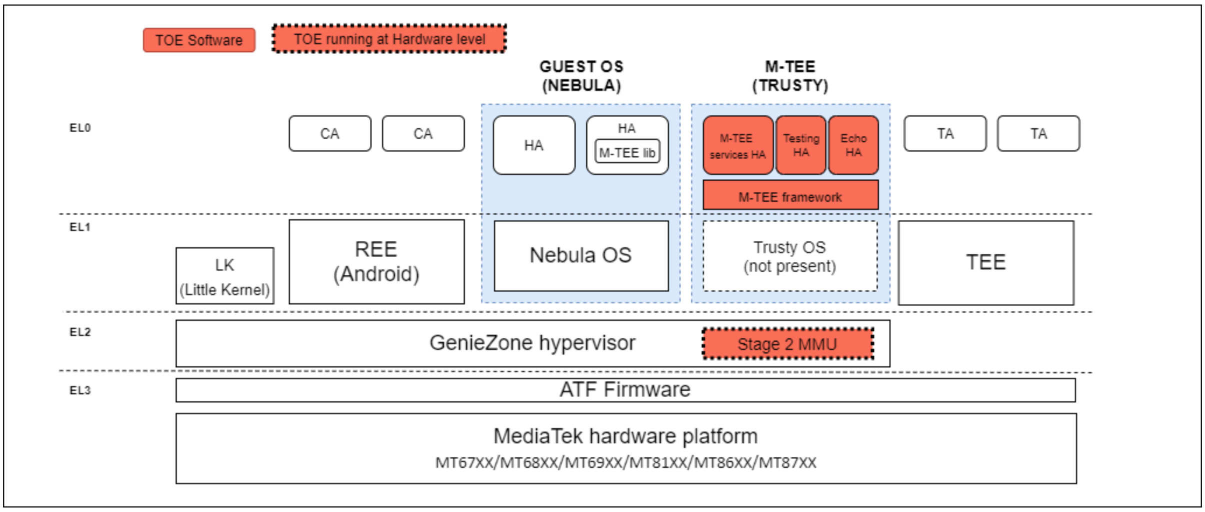

Figure 5 – Scope of the TOE

In this more detailed picture, it can be seen the particular vision of the architecture in which the TOE runs:

- The REE together with the CAs (not in the scope).
- A set of VMs (using the terminology of [NIAP-PP-VIRT]):
  - Nebula virtual machine, also called “Guest OS” in the TOE environment, running the Nebula OS (not in TOE scope), the Nebula HAs (not in TOE scope) and including the M-TEE lib (not in TOE scope), which is intended to be integrated in the Nebula HAs to be executed in EL0.
  - Trusty virtual machine, also just called “M-TEE” in the TOE environment, without operating system. It must be noticed that, even though the Trusty VM has no Operating System, an empty box with dashed line has been depicted in *Figure 5 – Scope of the TOE* in order to explicitly illustrate that this particular VM doesn’t include an operating system, unlike Nebula VM. Within Trusty VM, the following applications run as part of the TOE:
    - **M-TEE services HA**: which provides access to the hypervisor kernel to the CAs.
    - **Testing HA** and **Echo HA**: both HAs are disabled by the user (OEM) before the delivery to the end-user.
    - Finally, the Trusty virtual machine includes the **M-TEE framework** which provides the interface for HAs to communicate with the rest of the system.
  - Little Kernel (LK), is a virtual machine that is involved in the secure system boot process (not in scope of the TOE). It is in the Non-Secure World and is a bootloader that complete all staging and boots to the Linux Kernel.
- The TEE together with the TAs (not in scope).

- The GenieZone hypervisor, which is a hybrid type-II hypervisor, which contains the **Stage 2 MMU** as the only module of this element in scope.
- The underlying platform and the related firmware, which includes: BROM (non-TOE) and preloader (non-TOE), components required for secure system boot (not in scope).

As this security target used different terms based in the standard TEE terminology, a distinction should be made. Along this ST, the following concepts will be used:

- The term "TEE" refers to a standard Trusted Execution Environment which fully fulfils the GlobalPlatform system architecture. Also, for the TAs, that refer to the Trusted Applications in the standard TEE environment. The term TEE can be used to describe the complete architecture of REE + TEE or simply the Trusted OS.
- The term "M-TEE" refers to the components in the Trusty virtual machine.

1.4.2 TOE LOGICAL SCOPE

As is described in the section above, the TOE includes the elements of the architecture which supports the security functionality.

In particular, the TOE is composed of:

- Included in the Trusty virtual machine or just M-TEE:
    - **M-TEE services HA**: a piece of software used to provide services to the CAs. In particular it allows the user to load 3rd party HAs in the.
    - **Testing HA:** a piece of software which runs in EL0 used to allow the user to test their implementation. This HA is disabled by the user (OEM) before delivery to the end-user.
    - **Echo HA:** a piece of software which provides an echo interface for testing purposes. This HA is disabled by the user (OEM) before delivery to the end-user.
    - **M-TEE framework**: a set of software libraries which allow to run the HA directly in the VM without operating system, and provides the HA read/write access to which specific access permission.

Included in the GenieZone hypervisor:

- Stage 2 MMU as a hardware module which provides the following security functionalities under evaluation:
    - **Isolation**: the TOE ensures the isolation between components in the TOE and:
        - Between REE VM and Trusty VM or HAs running in Trusty VM.
        - Between REE VM and Nebula VM.

    - Between different HAs running in Trusty VM
    - Between Nebula VM and Trusty VM

- **Access control to memory and resources**: the TOE provides sharing memory mechanisms and communication between components in a controlled and secure way.

In this security target, the TSF is enforced by the *Stage 2 MMU*, where the TOE is defined to include several additional components which provides the interfaces to the TSF to the end user (integrator and Trusty VM HA developers).

1.4.3 TOE PHYSICAL SCOPE

1.4.3.1.1.1 PHYSICAL COMPONENTS

The TOE is composed of an image file (*.img) which includes all the TOE software components which provides the security functionality. In particular the Trusty virtual machine, M-TEE framework, Hypervisor Applications and Operating Systems as described in *Figure 5 – Scope of the TOE*.

The TOE is also composed by the Stage 2 MMU hardware module.

1.4.3.1.1.2 PHYSICAL COMPONENTS AND DELIVERY

Following is described the physical components that compose the TOE that are delivered to the intended user.

The software part of the TOE is delivered to the customer as a number of pre-built libraries. The AGD guidance provides instructions to the OEM on how to package and sign such libraries (along with other non-TOE SW parts) into a single binary image that is loaded into the hardware platform.

The identification and delivery method are also described:

TOE

Item type	**Deliverable name**	**Deliverable SHA256**	**Delivery method**	**Notes**
TOE Software (Binary images)	*mtk_armv8_el2-kernel.a*	SHA256: 8C0BE0AF87E7C1543C 5AABC154C1F4FD5604 5F37034639522145DD 458B0A1733	electronic delivery	This image contains the following TOE logical components • M-TEE Framework This image also contains the following non-TOE logical components: • GenieZone Hypervisor
	mtee_kernel_service.elf	SHA256: BF3E2D4352559B71DC F9A2BC723792D90CC8 E599F17B353E00D86D	electronic delivery	This image contains the following TOE logical component:

		DA30A838C6		• M-TEE services HA:
	echo.elf	SHA256: F8C2DD6423D98730C2 42333F3E04DFF5BF41C 8CAE74EEB5041958F76 A0473842	electronic delivery	This image contains the following TOE logical component: • Echo HA
	gz-test.elf	SHA256: 938B889DBD5A76E1F16 63F6CBBDDF6720E9849 498F04FFDE7C10ADAEA 30B9FE9	electronic delivery	This image contains the following TOE logical component: • Testing HA
TOE Hardware	Stage 2 MMU in GenieZone hypervisor	The Stage 2 MMU is identified by the following SoC models in which it is integrated: • MT67XX: ○ 6771 ○ 6765 ○ 6779 ○ 6768 ○ 6785 ○ 6781 ○ 6789 • MT68XX: ○ 6883 ○ 6885 ○ 6889 ○ 6893 ○ 6873 ○ 6853 ○ 6833 ○ 6877 ○ 6879 ○ 6895 ○ 6855 ○ 6886 ○ 6835 • MT69XX: ○ 6983 ○ 6985 • MT81XX: ○ 8195 • MT86XX: ○ 8675 • MT87XX: ○ 8768 ○ 8771 ○ 8781 ○ 8786 ○ 8788 ○ 8791 ○ 8795 ○ 8797 ○ 8798	As part of the SoC, which is embedded in a package or a device.	

Table 3 - Identification of the physical components of the TOE

Guidance's for SoC integrators:

Item	Name	Version	Type	Delivery method
Document	Mediatek Trusted Execution Environment (M-TEE) AGD_OPE	v1.1	pdf file	Electronic delivery
Document	Mediatek Trusted Execution Environment (M-TEE) AGD_PRE	V1.1	pdf file	electronic delivery
Document	[MTEE] Feature Enabling Quick Guide	v1.0	pdf file	electronic delivery
Document	Platform Security Solution Manual	v2.7a, 08/07/2022	pdf file	electronic delivery

Table 4 - Identification of the guidance's for SoC integrators

Guidance's for HA developers:

Item	Name	Version	Type	Delivery method
Document	M-TEE Development Guide	v1.3, 12/03/2021	pdf file	electronic delivery
Document	KREE API of M-TEE	1.0, 2021/10/04	pdf file	electronic delivery

Document	M-TEE Dynamic Loading HA User Guide	1.0, 2021/03/18	pdf file	electronic delivery
Document	Meditatek API Specification	0.1	Zip file containing HTML documentation	electronic delivery

Table 5 - Identification of the guidance's for HA developers

The SoC integrator could be a CA/HA developer or not, depending on whether the SoC integrator develops the applications as part of the integration process.

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].

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 EAL3 + ALC_FLR.1.

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

This Security Target uses the protection profile **[NIAP-PP-VIRT]** as a source for describing some items of the following sections:

- Security Problem Definition
- Security Objectives
- Security Requirements

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

The following Assets have been extracted from the description of the original Threats from section 3.1 of [NIAP-PP-VIRT], which this Security Target includes. In this regard, these Assets use the original descriptions and terminology of [NIAP-PP-VIRT]. The exception is “HA Address Space” asset, which is added in this ST and does not exist in [NIAP-PP-VIRT].

In this case, the original meaning from [NIAP-PP-VIRT] is kept, so that “Guest” is understood in the generic meaning of the term, in particular any Virtual Machine running in a hypervisor, regardless of whether the architecture calls a particular virtual machine "Guest OS".

The particular Threats taken from the Protection Profile and used in this Security Target from which these Assets are taken, are listed in section 3.3.

This Security Target describes the following Assets:

- **DOMAIN DATA:** The confidentiality or accessibility to user data or TSF data which are constrained to a domain.
  VMM FUNCTIONALITY: The access control and integrity of the Virtual Machine Manager.
- **HA ADDRESS SPACE:** The integrity and confidentiality of the address spaces of the hypervisor Applications (HAs) running in Trusty VM.

3.2 THREAT AGENTS

The following Threat Agents have been extracted from the description of the original Threats from section 3.1 of [NIAP-PP-VIRT], which this Security Target includes. In this regard, these agents use the original descriptions and terminology of [NIAP-PP-VIRT]. The exception is “Unauthorized Trusty HA” threat agent, which is added in this ST and does not exist in [NIAP-PP-VIRT].

The particular Threats taken from the Protection Profile and used in this Security Target from which these Threat Agents are taken, are listed in section 3.3.

This Security Target describes the following Threat Agents:

- **UNAUTHORIZED DOMAIN ENTITY:** An adversary on one domain or network that is not allowed to access to data or functionality of another domain.
- **NON-TOE SOFTWARE:** Non-TOE software, such as that running in Guest or Helper VMs or on the host platform.
- **UNAUTHORIZED TRUSTY HA:** Non-TOE Hypervisor Application (HA) running in Trusty that is not allowed to access data in the memory space of other HAs (TOE or non-TOE) running in Trusty VM.

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.

The Threats described in this section are based on the original Threats from section 3.1 of [NIAP-PP-VIRT], which this Security Target includes. The exception is "T.APP_COMPROMISE" threat, which is added in this ST and does not exist in [NIAP-PP-VIRT].

In the case of this Security Target, the descriptions of the Threats have been done by taking the descriptions directly from section 3.1 of [NIAP-PP-VIRT] , in order to preserve the original descriptions and terminology, except for the mentions to the threatened Assets and Threat Agents, that have been described separately in sections 3.1 and 3.2 to preserve a schematic description. The same terminology for those assets and threats agents defined in this ST has been used in the threats described below.

In particular, the original meaning from [NIAP-PP-VIRT] is kept, so that "Guest" is understood in the generic meaning of the term, in particular any Virtual Machine running in a hypervisor, regardless of whether the architecture calls a particular virtual machine "Guest OS".

To make a mapping between the terminology used in these Threats and the actual parts of the current architecture, please refer to sections 8 and 9 of this Security Target which allow to identify the related components of the actual operational environment of the TOE.

The following Threats are described in this Security Target:

- **T.DATA_LEAKAGE:** It is a fundamental property of VMs that the domains encapsulated by different VMs remain separate unless data sharing is permitted by policy. For this reason, all Virtualization Systems shall support a policy that prohibits information transfer between VMs.

  It shall be possible to configure VMs such that data cannot be moved between domains from VM to VM, or through virtual or physical network components under the control of the VS. When VMs are configured as such, it shall not be possible for data to leak between domains, neither by the express efforts of software or users of a VM, nor because of vulnerabilities or errors in the implementation of the VMM or other VS components.

If it is possible for data to leak between domains when prohibited by policy, then an **Unauthorized domain entity** can obtain Domain data from another domain. Such cross-domain data leakage can, for example, cause classified information, corporate proprietary information, or personally identifiable information to be made accessible to unauthorized entities.

- **T.VMM_COMPROMISE:** The VS is designed to provide the appearance of exclusivity to the VMs and is designed to separate or isolate their functions except where specifically shared. Failure of security mechanisms could lead to unauthorized intrusion into or modification of the **VMM functionality** or bypass of the VMM altogether by **non-TOE software**, such as that running in Guest or Helper VMs or on the host platform. This must be prevented to avoid compromising the VS.

- **T.APP_COMPROMISE:** Hypervisor applications running in Trusty VM are designed to isolate their address spaces from each other except when specifically shared through authorized channels. Failure of the isolation mechanisms between hypervisor applications could result in unauthorized intrusion and modification of these **HA Address Space** by a **Unauthorized Trusty HA**. This must be avoided in order to avoid compromising hypervisor applications running in Trusty VM.

Any other Threat from section 3.1 of [NIAP-PP-VIRT] not included in this section, has not been considered in this Security Target.

3.4 ASSUMPTIONS

Some of the Assumptions described in this section are based from the original Assumptions from section 3.2 of [NIAP-PP-VIRT], which this Security Target includes (A.PLATFORM_INTEGRITY).Additionally, there are additional Assumptions not present in the protection profile that have been included in order to fulfill the new Security Objectives for the Operational Environment in this Security Target which was described initially as Security Objectives for the TOE in [NIAP-PP-VIRT] (A.AUDIT, A.VMM_INTEGRITY).

The original meaning from [NIAP-PP-VIRT] is kept, so that “Guest” is understood in the generic meaning of the term, in particular any Virtual Machine running in a hypervisor, regardless of whether the architecture calls a particular virtual machine "Guest OS".

The following mapping can be done between them:

Assumptions in [NIAP-PP-VIRT]	Assumptions in this Security Target
A.PLATFORM_INTEGRITY	A.PLATFORM_INTEGRITY
not present	A.AUDIT
not present	A.VMM_INTEGRITY
A.NON_MALICIOUS_USER	A.NON_MALICIOUS_USER

Table 6 – Correspondence between the Assumptions in the PP and the ST

In the case of this Security Target, for those Assumptions taken from the protection profile, their descriptions have been done by taking the descriptions directly from section 3.2 of [NIAP-PP-VIRT] and have not been modified, in order to preserve the original descriptions and terminology.

To make a mapping between the terminology used in these Assumptions and the actual parts of the current architecture, please refer to sections 8 and 9 of this Security Target which allow to identify the related components of the actual operational environment of the TOE.

The assumptions when using the TOE are the following:

- **A.PLATFORM_INTEGRITY:** The platform has not been compromised prior to installation of the VS.
- **A.AUDIT:** There is an audit mechanism available to capture and protect data about what happens on a system so that it can later be examined to determine what has happened in the past.
- **A.VMM_INTEGRITY:** The integrity of each VMM component (intended not of the third-party software running inside of VMs or of the physical platform) is established and maintained.
- **A.NON_MALICIOUS_USER:** The user of the VS is not willfully negligent or hostile, and uses the VS in compliance with the applied enterprise security policy and guidance. At the same time, malicious applications could act as the user, so requirements which confine malicious applications are still in scope.

Any other Assumption from section 3.2 of [NIAP-PP-VIRT] not included in this section, has not been considered in this Security Target.

3.5 ORGANIZATIONAL SECURITY POLICIES

This document does not define any OSPs.

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.

The Security Objectives for the TOE described in this section are based from the original Security Objectives for the TOE from section 4.1 of [NIAP-PP-VIRT], which this Security Target includes and have not been modified, in order to preserve the original descriptions and terminology. The exception is the objective "O.APP_ISOLATION", that has been added to this ST and is not present in [NIAP-PP-VIRT]. Other Security Objectives for the TOE from section 4.1 of [NIAP-PP-VIRT] have not been considered in this Security Target.

To make a mapping between the terminology used in these objectives and the actual parts of the current architecture, please refer to sections 8 and 9 of this Security Target which allow to identify the related components of the actual operational environment of the TOE.

In particular, the original meaning from [NIAP-PP-VIRT] is kept, so that "Guest" is understood in the generic meaning of the term, in particular any Virtual Machine running in a hypervisor, regardless of whether the architecture calls a particular virtual machine "Guest OS".

The following Security Objectives of the TOE have been considered in this Security Target:

- **O.VM_ISOLATION:** VMs are the fundamental subject of the system. The VMM is responsible for applying the system security policy (SSP) to the VM and all resources. As basic functionality, the VMM must support a security policy that mandates no information transfer between VMs.

  The VMM must support the necessary mechanisms to isolate the resources of all VMs. The VMM partitions a platform's physical resources for use by the supported virtual environments. Depending on customer requirements, a VM may need a completely isolated environment with exclusive access to system resources or share some of its resources with other VMs. It must be possible to enforce a security policy that prohibits the transfer of data between VMs through shared devices. When the platform security policy allows the sharing of resources across VM boundaries, the VMM must ensure that all access to those resources is consistent with the policy. The VMM may delegate the responsibility for the mediation of resource sharing to select Service VMs; however, in doing so, it remains responsible for mediating access to the Service VMs, and each Service VM must mediate all access to any shared resource that has been delegated to it in accordance with the SSP.

Both virtual and physical devices are resources requiring access control. The VMM must enforce access control in accordance with system security policy. Physical devices are platform devices with access mediated via the VMM per the OE.VMM_INTEGRITY objective. Virtual devices may include virtual storage devices and virtual network devices. Some of the access control restrictions must be enforced internal to Service VMs, as may be the case for isolating virtual networks. VMMs may also expose purely virtual interfaces. These are VMM specific, and while they are not analogous to a physical device, they are also subject to access control. The VMM must support the mechanisms to isolate all resources associated with virtual networks and to limit a VM's access to only those virtual networks for which it has been configured.

The VMM must also support the mechanisms to control the configurations of virtual networks according to the SSP.

Application Note

Regarding this Security Target, the Virtualization Manager is indistinguishable from the Virtualization System, that is the complete Hypervisor. The only part of this component which is in the scope is the Stage 2 MMU, and the fulfillment of this objective relies in it.

- **O.DOMAIN_INTEGRITY:** While the VS is not responsible for the contents or correct functioning of software that runs within Guest VMs, it is responsible for ensuring that the correct functioning of the software within a Guest VM is not interfered with by other VMs.

  Application Note

  While the Virtualization System refers to the complete Hypervisor, the only part of this component which is in the scope is the Stage 2 MMU, and the fulfillment of this objective relies in it.

- **O.APP_ISOLATION:** The address space of HAs in Trusty VM shall be isolated from the address space of other HAs in Trusty. Only authorized channels shall be used to perform communication between HAs in Trusty.

Any other Security Objective for the TOE from section 4.1 of [NIAP-PP-VIRT] not included in this section, has not been considered in this Security Target.

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.

Some of the Security Objectives for the Operational Environment described in this section are based in the original Security Objectives for the TOE and from the environment from section 4.1 and 4.2 of [NIAP-PP-VIRT], which this Security Target includes as Security Objectives for the Operational Environment (OE.AUDIT, OE.PLATFORM_INTEGRITY, OE.VMM_INTEGRITY).

Finally, two new Security Objective for the Operational Environment not present in [NIAP-PP-VIRT] have been included in this Security Target (OE.PLATFORM_ISOLATION, OE.NON_MALICIOUS_USER).

The following mapping can be done between them:

Security objectives in [NIAP-PP-VIRT]	Security Objectives for the Operational Environment in this Security Target
O.AUDIT	OE.AUDIT
O.PLATFORM_INTEGRITY	OE.PLATFORM_INTEGRITY
O.VMM_INTEGRITY	OE.VMM_INTEGRITY
not present	OE.PLATFORM_ISOLATION
not present	OE.NON_MALICIOUS_USER

Table 7 – Correspondence between the security objectives in the PP and the ST

In the case of this Security Target, for those Security Objectives for the Operational Environment taken from the Protection Profile, they have been described by taking the descriptions directly from section 4.1 of [NIAP-PP-VIRT] and have not been modified in order to preserve the original descriptions and terminology.

In particular, the original meaning from [NIAP-PP-VIRT] is kept, so that “Guest” is understood in the generic meaning of the term, in particular any Virtual Machine running in a hypervisor, regardless of whether the architecture calls a particular virtual machine "Guest OS".

To make a mapping between the terminology used in these objectives and the actual parts of the current architecture, please refer to sections 8 and 9 of this Security Target which allow to identify the related components of the actual operational environment of the TOE.

The following Security Objectives of the Operational Environment have been considered in this Security Target:

- **OE.AUDIT:** An audit log must be created that captures accesses to the objects the TOE protects. The log of these accesses, or audit events, must be protected from modification, unauthorized access, and destruction. The audit log must be sufficiently detailed to indicate the date and time of the event, the identity of the user, the type of event, and the success or failure of the event.
- **OE.PLATFORM_INTEGRITY:** The integrity of the VMM depends on the integrity of the hardware and software on which the VMM relies. Although the VS does not have complete control over the integrity of the platform, the VS should as much as possible try to ensure that no users or software hosted by the VS can undermine the integrity of the platform.
- **OE.VMM_INTEGRITY:** Integrity is a core security objective for Virtualization Systems. To achieve system integrity, the integrity of each VMM component must be established and maintained. This objective concerns only the integrity of the VS, not the integrity of software running inside of Guest VMs or of the physical platform. The overall objective is to ensure the integrity of critical components of a VS.

  Initial integrity of a VS can be established through mechanisms such as a digitally signed installation or update package, or through integrity measurements made at launch. These checks are performed by the preloader, a component running on the hardware platform that is responsible for loading and verifying the TOE images. Integrity is maintained in a running system by careful protection of the VMM from untrusted users and software. For example, it must not be possible for software running within a Guest VM to exploit a vulnerability in a device or hypercall interface and gain control of the VMM. The vendor must release patches for vulnerabilities as soon as practicable after discovery.
- **OE.PLATFORM_ISOLATION:** The platform provides and manages the isolation to the hardware resources and between Normal World and Secure World.
- **OE.NON_MALICIOUS_USER:** OEMs (HA developers and SoC integrators) accessing the TOE are trusted, so that they use the SV in accordance with the company's security policy and guidelines.

Application Note:

OE.PLATFORM_ISOLATION refers to the following non-TOE resources:

- Hardware and software components of the GenieZone hypervisor, except the *Stage 2 MMU* (in TOE scope), that support the virtualization services in the non-secure world, the secure boot and the communication with the other system components.

The Mediatek Platform, all the hardware and firmware (ATF firmware) of the TrustZone-based Mediatek platform providing isolation and communication between the secure and non-secure execution environments. The contemplated model series are those listed in section 1.3.4 Non-TOE Hardware/Software/Firmware of this ST.

Any other Security Objective for the Operational Environment from section 4.2 of [NIAP-PP-VIRT] not included in this section, has not been considered in this Security Target.

4.3 SECURITY OBJECTIVES RATIONALE

The following table provides a mapping of:

- threats traced back by the indicated security objectives
- OSPs enforced by the indicated security objectives. In this Security Target, there is no OSPs defined.
- assumptions upheld by the indicated security objectives for the operational environment

This illustrates that the security objectives counter all threats and the security objectives for the operational environment uphold all assumptions.

	O.VM_ISOLATION	O.DOMAIN_INTEGRITY	O.APP_ISOLATION	OE.AUDIT	OE. PLATFORM_INTEGRITY	OE.VMM_INTEGRITY	OE.PLATFORM_ISOLATON	OE.NON_MALICIOUS_USER
T.DATA_LEAKAGE	X	X					X	
T.VMM_COMPROMISE	X					X		
T.APP_COMPROMISE			X					
A.PLATFORM_INTEGRITY					X			
A.AUDIT				X				
A.VMM_INTEGRITY						X		
A.NON_MALICIOUS_USER								X

Table 8 - Security Objectives vs Security Problem Definition

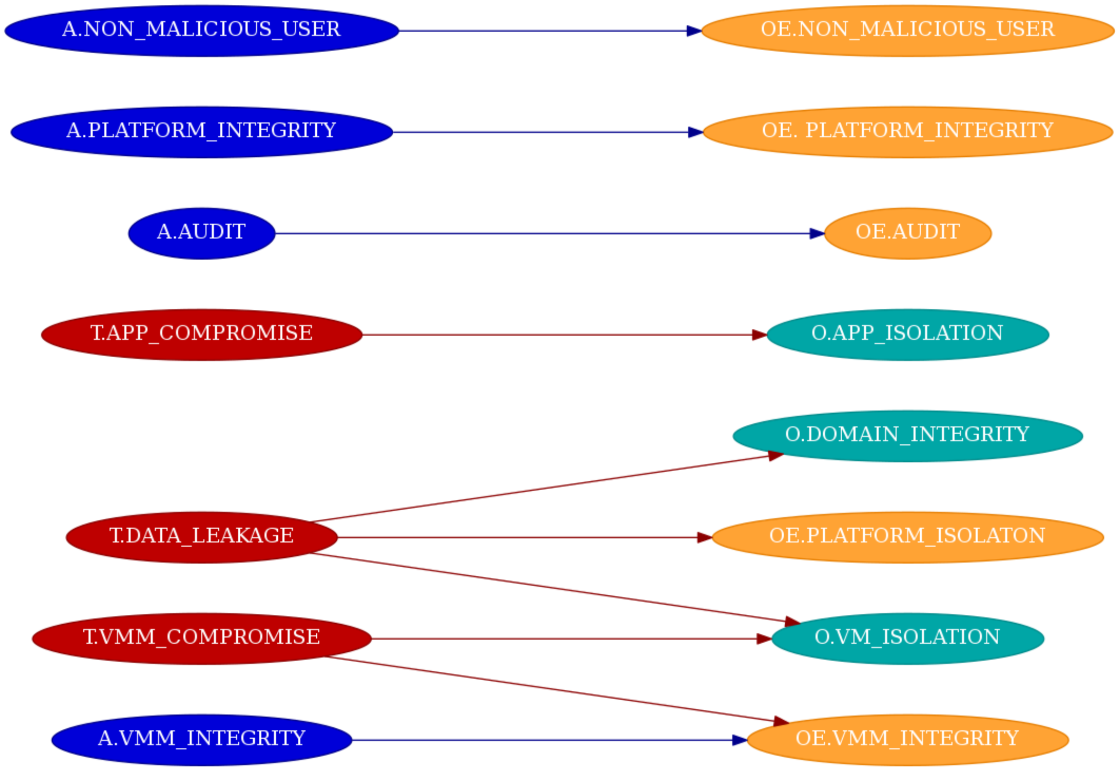

Figure 6 - *Mapping of Security Problem Definition to Security Objectives*

4.3.1 THREATS

Following is described how the threats are countered by the security objectives. The ones for the operational environment have been remarked in **YELLOW** to distinguish them from the security objectives for the TOE, which are remarked in **BROWN**.

- **T.DATA_LEAKAGE:**
  - Logical separation of VMs and enforcement of domain integrity prevent unauthorized transmission of data from one VM to another (**O.VM_ISOLATION**).
  - Logical separation of VMs and enforcement of domain integrity prevent unauthorized transmission of data from one VM to another (**O.DOMAIN_INTEGRITY)**.
  - Additionally, the environment will provide means to isolate and manage the access from the virtual environment to the physical resources and the non-virtual environment (**OE.PLATFORM_ISOLATON**).
- **T.VMM_COMPROMISE:**
  - Maintaining the integrity of the VMM and ensuring that VMs execute in isolated domains mitigate the risk that the VMM can be compromised or bypassed. (**OE.VMM_INTEGRITY**).
  - Maintaining the integrity of the VMM and ensuring that VMs execute in isolated domains mitigate the risk that the VMM can be compromised or bypassed (**O.VM_ISOLATION**).
- **T.APP_COMPROMISE:**
  - Maintaining the integrity and confidentiality of memory spaces of HAs running in Trusty VM, through isolation of their memory spaces, limiting HA to HA (both in Trusty VM) communications only through authorized channels, and mitigating the risk that they can be compromised (**O.APP_ISOLATION**).

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.DATA_LEAKAGE	O.VM_ISOLATION O.DOMAIN_INTEGRITY OE.PLATFORM_ISOLATION
T.VMM_COMPROMISE	O.VM_ISOLATION

Threats	Security Objectives
	OE.VMM_INTEGRITY
T.APP_COMPROMISE	O.APP_ISOLATION

Table 9 - Threats vs Security Objectives

4.3.2 ASSUMPTIONS

Following is described how the Assumptions upheld by the security objectives for the operational environment remarked in **YELLOW**.

- **A.PLATFORM_INTEGRITY:** The integrity of the VMM depends on the integrity of the hardware and software on which the VMM relies. This assumption is directly upheld by **OE.PLATFORM_INTEGRITY**.
- **A.AUDIT:** The purpose of audit is to capture and protect data about what happens on a system so that it can later be examined to determine what has happened in the past. This assumption is directly upheld by **OE.AUDIT**.
- **A.VMM_INTEGRITY:** The integrity of the platform software is verified during the boot up process. This assumption is directly upheld by **OE.VMM_INTEGRITY**.
- **A.NON_MALICIOUS_USER:** The purpose of this assumption is that no malicious OEM (HA developers and SoC integrators) compromises the integrity of the VS. This assumption is directly upheld by **OE.NON_MALICIOUS_USER**.

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_INTEGRITY	OE.PLATFORM_INTEGRITY
A.AUDIT	OE.AUDIT
A.VMM_INTEGRITY	OE.VMM_INTEGRITY
A.NON_MALICIOUS_USER	OE.NON_MALICIOUS_USER

Table 10 - Assumptions vs Security Objectives for the Operational Environment

5 EXTENDED COMPONENTS DEFINITION

5.1 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 original families in this class are organized into four groups:

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

The following extended families use the definition and structure of the pre-defined class FDP.

5.1.1 HARDWARE-BASED ISOLATION (FDP_HBI_EXT)

Family behavior

This family is defined according to [NIAP-PP-VIRT].

Component levelling

FDP_HBI_EXT.1, Hardware-Based Isolation Mechanisms, requires the TSF to identify the mechanisms used to isolate Guest VMs from platform hardware resources.

Management: FDP_HBI_EXT.1

No management activities are foreseen.

Audit: FDP_HBI_EXT.1

No actions are defined to be auditable.

FDP_HBI_EXT.1: Hardware-Based Isolation Mechanisms

Hierarchical to:

No other components.

Dependencies:

FDP_VMS_EXT.1

FDP_HBI_EXT.1.1: ***The TSF shall use [selection: no mechanism, [assignment: list of platform-provided, hardware-based mechanisms]] to constrain the VM's direct access to the following physical devices: [selection: no devices, [assignment: physical devices to which the VMM allows VMs physical access]]***.

5.1.2 PHYSICAL PLATFORM RESOURCE (FDP_PPR_EXT)

Family behavior

This family is defined according to [NIAP-PP-VIRT]. However, unlike in [NIAP-PP-VIRT], the dependency of FDP_PPR_EXT.1 component on FMT_SMR.1 has been excluded from the definition of such extended component and FDP_PPR_EXT.1 component is modified to indicate that Guest VM access to physical platform resources is controlled directly by the TSF and not by and administrator through the TSF. This change makes the dependency with FMT_SMR.1 unnecessary. Moreover, this ST doesn't include any component from FAU_GEN family, therefore no auditable actions are defined in FDP_PPR_EXT family components in this ST. For those reasons, FDP_PPR_EXT family components as defined in this ST don't include any management activities or auditable actions.

Component levelling

FDP_PPR_EXT.1, Physical Platform Resource Controls, requires the TSF to define the hardware resources that Guest VMs may always access, may never access, and may conditionally access under TSF control.

Management: FDP_PPR_EXT.1

No management activities are foreseen.

Audit: FDP_PPR_EXT.1

No actions are defined to be auditable.

FDP_PPR_EXT.1: Physical Platform Resource Controls

Hierarchical to:

No other components.

Dependencies:

FDP_HBI_EXT.1

FDP_PPR_EXT.1.1: ***The TSF shall control VM access to the following physical platform resources: [assignment: list of physical platform resources the VMM is able to control access to]***

FDP_PPR_EXT.1.2: ***The TSF shall explicitly deny all Guest VMs access to the following physical platform resources: [selection: no physical platform resources, [assignment: list of physical platform resources to which access is always denied]]***

FDP_PPR_EXT.1.3:

The TSF shall explicitly allow all Guest VMs access to the following physical platform resources: [selection: no physical platform resources, [assignment: list of physical platform resources to which access is always allowed]].

5.1.3 VM SEPARATION (FDP_VMS_EXT)

Family behavior

This family is defined according to [NIAP-PP-VIRT] except for the modifications described in the application note at the end of this section.

Component levelling

FDP_VMS_EXT.1, VM Separation, requires the TSF to maintain logical separation between Guest VMs except through the use of specific mechanisms.

Management: FDP_VMS_EXT.1

No management activities foreseen.

Audit: FDP_VMS_EXT.1

No actions are defined to be auditable.

FDP_VMS_EXT.1: VM Separation

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FDP_VMS_EXT.1.1: ***The VS shall provide the following mechanisms for transferring data between Guest VMs: [selection: no mechanism, virtual networking, [assignment: other inter-VM data sharing mechanisms]]***

FDP_VMS_EXT.1.2: ***The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs.***

FDP_VMS_EXT.1.3: ***The VS shall ensure that no Guest VM is able to read or transfer data to or from another Guest VM except through the mechanisms listed in FDP_VMS_EXT.1.1***

Application Note:

Original component FDP_VMS_EXT.1.3 from [NIAP-PP-VIRT] has been removed from this definition. The rest of the components have been kept in consecutive order. The reason for removal is that the mechanisms to be selected in FDP_VMS_EXT.1.1 are not configurable by an administrator in this TOE. Moreover, because those mechanisms are not configurable by an administrator, this extended SFR has been defined in this ST without any associated management activities.

5.1.4 HYPERVISOR APPLICATION ISOLATION (FDP_HAI_EXT)

Family behavior

This family is defined to address specific requirements for address space isolation between Hypervisor Applications running in Trusty VM.

Component levelling

Management: FDP_HAI_EXT.1

There are no management activities foreseen.

Audit: FDP_HAI_EXT.1

There are no auditable events foreseen.

FDP_HAI_EXT.1: Hypervisor Application Isolation

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FDP_HAI_EXT.1.1: ***The TSF shall enforce isolation of address space between Hypervisor Applications running on Trusty VM.***

FDP_HAI_EXT.1.2: ***The TSF shall allow the following exceptions to isolation of address space between Hypervisor Applications: [assignment: exception mechanisms]***

5.2 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.

The following extended families use the definition and structure of the pre-defined class FPT.

5.2.1 HYPERCALL (FPT_HCL_EXT)

Family behavior

This family is defined according to [NIAP-PP-VIRT]. However, unlike in [NIAP-PP-VIRT], the dependency of FPT_HCL_EXT.1 component on FMT_SMR.1 has been excluded from the definition of such extended component since there are no roles or identification TSF linked to management of the VMs in this TOE. Moreover, no audit activities are included in the definition of the components in this family because no FAU_GEN components are included in this ST.

Component levelling

FPT_HCL_EXT.1, Hypercall Controls, requires the TSF to implement appropriate parameter validation to protect the VMM from unauthorized access through a hypercall interface.

Management: FPT_HCL_EXT.1

No management activities foreseen.

Audit: FPT_HCL_EXT.1

No actions are defined to be auditable.

FPT_HCL_EXT.1: Hypercall Controls

Hierarchical to:

No other components.

Dependencies:

No dependencies.

FPT_HCL_EXT.1.1: ***The TSF shall validate the parameters passed to the hypercall interface prior to execution of the VMM functionality exposed by that interface.***

5.2.2 VMM ISOLATION (FPT_VIV_EXT)

Family behavior

This family is defined according to [NIAP-PP-VIRT].

Component levelling

FPT_VIV_EXT.1, VMM Isolation from VMs, requires the TSF to ensure that there is no mechanism by which a Guest VM can interface with the TOE, other VMs, or the hardware platform without authorization.

Management: FPT_VIV_EXT.1

No management activities foreseen.

Audit: FPT_VIV_EXT.1

No actions are defined to be auditable.

FPT_VIV_EXT.1: VMM Isolation from VMs

Hierarchical to:

No other components.

Dependencies:

FDP_PPR_EXT.1

FDP_VMS_EXT.1

FPT_VIV_EXT.1.1: ***The TSF must ensure that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform.***

FPT_VIV_EXT.1.2: ***The TSF must ensure that a Guest VM is unable to invoke platform code that runs at a privilege level equal to or exceeding that of the VMM without involvement of the VMM.***

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

The following Security Functional Requirements have been taken from section 5.1 of [NIAP-PP-VIRT]. The description made in these SFRs are made using the same terminology of the Protection Profile. In particular, the original meaning from [NIAP-PP-VIRT] is kept, so that "Guest" is understood in the generic meaning of the term, in particular any Virtual Machine running in a hypervisor, regardless of whether the architecture calls a particular virtual machine "Guest OS".

6.1.1 FDP: USER DATA PROTECTION

6.1.1.1 FDP_HBI_EXT.1: HARDWARE-BASED ISOLATION MECHANISMS

FDP_HBI_EXT.1.1 The TSF shall use ***[selection: [assignment: Stage 2 MMU]]*** to constrain the VM's direct access to the following physical devices: ***[selection: [runtime memory, system timer, GIC, RTC]]***.

6.1.1.2 FDP_PPR_EXT.1: PHYSICAL PLATFORM RESOURCE CONTROLS

FDP_PPR_EXT.1.1 The TSF shall control VM access to the following physical platform resources: ***[assignment: dynamic secure MEMORY for SVP (Secure Video Path Frame Buffer), TEE sec_mem, PROT 2D_FR (Face authentication Frame Buffer), WFD (WiFi Display Frame Buffer), Dynamic HA (M-TEE APP) Buffer]***.

FDP_PPR_EXT.1.2 The TSF shall explicitly deny all Guest VMs access to the following physical platform resources: ***[selection: [assignment: GPU, TUI (Trusted UI), Power Management Unit, APU, ISP***

Processor, Secure APU, Secure APU controller, Sensor Hub, Audio DSP, MMuP, GPU uP, MD, CONSYS]].

FDP_PPR_EXT.1.3 The TSF shall explicitly allow all Guest VMs access to the following physical platform resources: ***[selection: VM's own runtime memory]***.

6.1.1.3 FDP_VMS_EXT.1: VM SEPARATION

FDP_VMS_EXT.1.1 The VS shall provide the following mechanisms for transferring data between Guest VMs: ***[selection: [assignment:***

- ***IPC used to transfer data between:***
  - ***REE and HA in Trusty***
  - ***HA and HA within Trusty***
  - ***REE and Nebula VM***
- ***shared memory used to transfer data between:***
  - ***REE and HA in Trusty***
  - ***REE and Nebula VM]]***.

FDP_VMS_EXT.1.2 The TSF shall by default enforce a policy prohibiting sharing of data between Guest VMs.

FDP_VMS_EXT.1.3 The VS shall ensure that no Guest VM is able to read or transfer data to or from another Guest VM except through the mechanisms listed in FDP_VMS_EXT.1.1.

Application note:

IPC communication between HAs in Nebula is not covered by this SFR. IPC communication between HAs in Nebula and HAs in Trusty is not supported.

6.1.1.4 FDP_HAI_EXT.1: HYPERVISOR APPLICATION ISOLATION

FDP_HAI_EXT.1.1 The TSF shall enforce isolation of address space between Hypervisor Applications running on Trusty VM.

FDP_HAI_EXT.1.2 The TSF shall allow the following exceptions to isolation of address space between Hypervisor Applications: ***[assignment: IPC mechanism used to transfer data between HAs in Trusty VM, as specified in FDP_VMS_EXT.1.1]***

6.1.2 FPT: PROTECTION OF THE TSF

6.1.2.1 FPT_HCL_EXT.1: HYPERCALL CONTROLS

FPT_HCL_EXT.1.1 The TSF shall validate the parameters passed to the hypercall interface prior to execution of the VMM functionality exposed by that interface.

Application note:

Only the hypercalls to Stage 2 MMU are covered.

6.1.2.2 FPT_VIV_EXT.1: VMM ISOLATION FROM VMS

FPT_VIV_EXT.1.1 The TSF must ensure that software running in a VM is not able to degrade or disrupt the functioning of other VMs, the VMM, or the Platform.

FPT_VIV_EXT.1.2 The TSF must ensure that a Guest VM is unable to invoke platform code that runs at a privilege level equal to or exceeding that of the VMM without involvement of the VMM.

6.2 SECURITY ASSURANCE REQUIREMENTS

The development and the evaluation of the TOE shall be done in accordance to the following security assurance requirements: **EAL3 + ALC_FLR.1**

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
ALC: Life-cycle support	ALC_DEL.1: Delivery procedures ALC_CMC.3: Authorisation controls ALC_CMS.3: Implementation representation CM coverage ALC_DVS.1: Identification of security measures ALC_LCD.1: Developer defined life-cycle model ALC_FLR.1: Basic flaw remediation

Assurance Class	Assurance Components
ADV: Development	ADV_ARC.1: Security architecture description ADV_FSP.3: Functional specification with complete summary ADV_TDS.2: Architectural design
AGD: Guidance documents	AGD_OPE.1: Operational user guidance AGD_PRE.1: Preparative procedures
ATE: Tests	ATE_FUN.1: Functional testing ATE_IND.2: Independent testing – sample ATE_COV.2: Analysis of coverage ATE_DPT.1: Testing: basic design
AVA: Vulnerability assessment	AVA_VAN.2: Vulnerability analysis

Table 11 – Security Assurance Requirements

6.3 SECURITY REQUIREMENTS RATIONALE

6.3.1 NECESSITY AND SUFFICIENCY ANALYSIS

Map of the coverage of the security objectives for the TOE by the Security Functional Requirements:

SFR / TOE Security Objective	O.VM_ISOLATION	O.DOMAIN_INTEGRITY	O.APP_ISOLATION
FDP_HBI_EXT.1	X	X	

SFR / TOE Security Objective	O.VM_ISOLATION	O.DOMAIN_INTEGRITY	O.APP_ISOLATION
FDP_PPR_EXT.1	X		
FDP_VMS_EXT.1	X	X	X
FPT_VIV_EXT.1	X	X	
FPT_HCL_EXT.1	X	X	
FDP_HAI_EXT.1			X

Table 12 – SFRs / TOE Security Objectives coverage

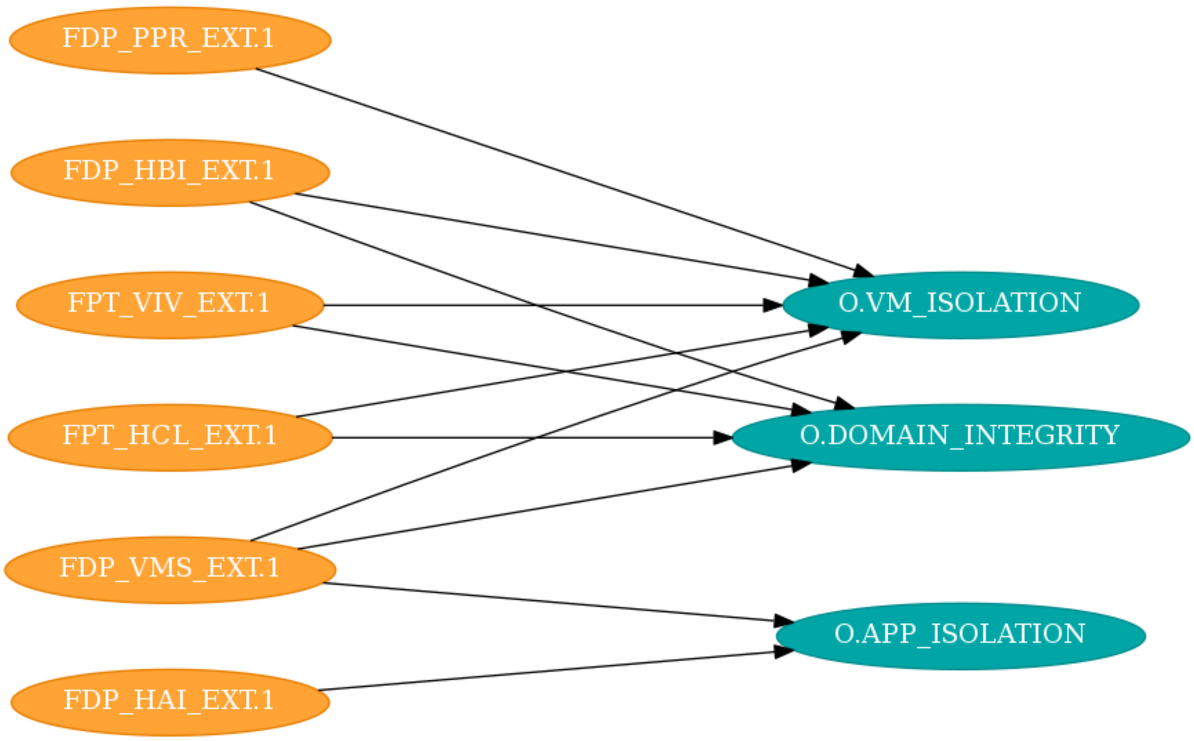

Figure 7 – Mapping of SFRs to TOE Security Objectives

6.3.2 SECURITY REQUIREMENT SUFFICIENCY

The following rationale provides justification for each security objective for the TOE, showing that the SFRs are suitable to meet and achieve the security objectives:

O.VM_ISOLATION: The following Security Functional Requirements are used to address this objective, according to **[NIAP-PP-VIRT]**:

- **FDP_VMS_EXT.1** ensures that authorized data transfers between VMs are constrained to a limited number of communication paths and enforcing a policy that prohibits sharing of data between Guest VMs.
- **FDP_PPR_EXT.1** requires support for reducing attack surface through disabling access to unnecessary physical platform resources.
- **FDP_HBI_EXT.1** requires that the TOE use platform-supported mechanisms for access to physical devices.
- **FPT_VIV_EXT.1** ensures that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
- **FPT_HCL_EXT.1** Requires that hypercall interfaces are provided and that the TSF validates the parameters passed to the hypercall interfaces prior to executing the VMM functionality exposed through them.

O.DOMAIN_INTEGRITY: The following Security Functional Requirements are used to address this objective, according to **[NIAP-PP-VIRT]**:

- **FDP_VMS_EXT.1** ensures that authorized data transfers between VMs are done securely.
- **FDP_HBI_EXT.1** requires that the TOE use platform-supported mechanisms for access to physical devices.
- **FPT_VIV_EXT.1** ensures that untrusted VMs cannot invoke privileged code without proper hypervisor mediation.
- **FPT_HCL_EXT.1** requires that hypercall interfaces be provided.

O.APP_ISOLATION: The following Security Functional Requirements are used to address this objective:

- **FDP_HAI_EXT.1** ensures isolation between HAs memory space and that data transfers between hypervisor applications (HAs) are done only through authorized channels.
- **FDP_VMS_EXT.1** ensures that dedicated IPC channels exist for authorized data exchange between HAs running in Trusty.

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.

SFR	Required	Fulfilled	Missing
FDP_HBI_EXT.1	FDP_VMS_EXT.1	FDP_VMS_EXT.1	None
FDP_PPR_EXT.1	FDP_HBI_EXT.1	FDP_HBI_EXT.1	None
FDP_VMS_EXT.1	None	None	None
FPT_VIV_EXT.1	FDP_PPR_EXT.1, FDP_VMS_EXT.1	FDP_PPR_EXT.1, FDP_VMS_EXT.1	None
FPT_HCL_EXT.1	None	None	None
FDP_HAI_EXT.1	None	None	None

Table 13 – SFR Dependencies

6.3.4 SAR RATIONALE

The assurance level defined is the predefined assurance package EAL3 + ALC_FLR.1 in order to reach the Enhanced-basic attack potential in accordance with the threat environment that is experienced by typical consumers of the TOE.

This EAL3 + ALC_FLR.1 permits the developer to gain sufficient assurance from positive security engineering based on good M-TEE commercial development practices that are compatible with industry constraints, particularly the life cycle of the TOE and TOE-enabled devices. The developer has provided evidence of security engineering at design, testing, guidance, configuration management and delivery levels as required by EAL3 + ALC_FLR.1. In order to cope with the high exposure of the M-TEE and the interest that M-TEE-enabled devices and their embedded services may represent to attackers, the product has to show resistance to Enhanced-basic attack potential. This attack potential matches the threat analysis performed on typical architectures and attackers' profiles in the field.

Particularly, the standard component AVA_VAN.2 provides a good level of assurance against SW attacks, for instance mobile application malware that is spreading through uncontrolled application stores, exploiting already known software vulnerabilities. Standard AVA_VAN.2 is well-fit for devices managed within a controlled environment for services which the end user may not have any interest in attacking.

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.3 (hierarchically above ADV_FSP.1)	None
ALC_CMC.3	ALC_CMS.1, ALC_DVS.1, ALC_LCD.1	ALC_CMS.3 (hierarchically above ALC_CMS.1), ALC_DVS.1, ALC_LCD.1	None
ALC_CMS.3	None	None	None
ADV_FSP.3	ADV_TDS.1	ADV_TDS.2 (hierarchically above ADV_TDS.1)	None
AGD_OPE.1	ADV_FSP.1	ADV_FSP.3 (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.3 (hierarchically above ADV_FSP.2), AGD_OPE.1, AGD_PRE.1, ATE_COV.2 (hierarchically	None

SAR	Required	Fulfilled	Missing
		above ATE_COV.1), ATE_FUN.1	
AVA_VAN.2	ADV_ARC.1, ADV_FSP.2, ADV_TDS.1, AGD_OPE.1, AGD_PRE.1	ADV_ARC.1, ADV_FSP.3 (hierarchically above ADV_FSP.2), ADV_TDS.2 (hierarchically above ADV_TDS.1), AGD_OPE.1, AGD_PRE.1	None
ASE_SPD.1	None	None	None
ALC_DEL.1	None	None	None
ADV_ARC.1	ADV_FSP.1, ADV_TDS.1	ADV_FSP.3 (hierarchically above ADV_FSP.1), ADV_TDS.2 (hierarchically above ADV_TDS.1)	None
ATE_FUN.1	ATE_COV.1	ATE_COV.2 (hierarchically above ATE_COV.1)	None
ADV_TDS.2	ADV_FSP.3	ADV_FSP.3	None
ALC_DVS.1	None	None	None
ALC_LCD.1	None	None	None
ALC_FLR.1	None	None	None
ATE_COV.2	ADV_FSP.2, ATE_FUN.1	ADV_FSP.3 (hierarchically above ADV_FSP.2), ATE_FUN.1	None
ATE_DPT.1	ADV_ARC.1, ADV_TDS.2, ATE_FUN.1	ADV_ARC.1, ADV_TDS.2, ATE_FUN.1	None

Table 14 – SAR dependencies

7 TOE SUMMARY SPECIFICATION

The Hypervisor Applications are designed to provide secure services similar to those provided by a Trusted Application, to the Client Applications used by the end user.

The prebuilt HAs in Trusty are packed and signed during the final stages of the life cycle. These signatures are verified together with the TOE image during the secure boot, so that they are considered as trusted.

The TOE admits the installation of 3rd party HAs or OEM HAs, which are dynamically installed in M-TEE before the delivery to the end-user as part of the handheld device. In those cases, the signature verification can be done on each load, but it is defined by the OEMs during the life cycle.

The TOE is composed of several components which include, hardware and software elements, those of them are detailed in section 1.4

However, even if the TOE is a set of several components, the core of the TSF is implemented in the Stage 2 MMU hardware module, belonging to the Hypervisor. This element is in charge of providing in fact the security to the TOE which is based in access control that provides isolation between elements. Other parts of the architecture (namely the components in Trusty VM) contribute to this TSF in a by implementing hypercall interfaces, and communication APIs for IPC and data sharing mechanisms that trigger the isolation functionality.

The below paragraphs provide a summary of how the TOE implements isolation functionality. This isolation occurs for a number of communication paths between components of the architecture in which the TOE participates. Such paths are those involving one HA in Trusty as the beginning or end of a communication, and also those communications between Nebula VM and REE VM. Since there are no logical components in TOE scope residing in the Nebula VM, communications between HAs in Nebula VM and communications between an HA in Nebula VM and a component in the TEE are not part of the TSF related to communication (however, some of these communication paths involve participation of isolation-related TSF as will be explained below).

These communications controlled by TSF-implemented isolation are made mainly by the usage of SMC calls, Shared Memory and hypercall mechanisms. In particular, the M-TEE framework is the interface for the HA in Trusty to call a Platform service call, which sends hypervisor calls to GenieZone hypervisor in EL2 (which execute corresponding GenieZone kernel services), or SMC calls that are handled in EL3.

Isolation provided by the TSF is described in detail below:

TOE by means of the Stage 2 MMU provides the access control mechanisms to allow or deny the communication between the different parts of the TOE and provides the isolation barrier within the virtualized layer: in EL0, the M-TEE Services HA provides shared memory APIs to other HAs that trigger isolation functionality; in EL1, M-TEE framework provides APIs to HAs (in a system calls fashion) to expose functionalities such as IPC, Shared Memory or secure memory management.

The isolation mechanisms in scope are those which includes one HA in Trusty VM as the requestor or destination of a communication, and also those communications between Nebula VM and REE VM. For some paths, these include also isolation mechanisms belonging to the operational environment. The exception is isolation between HAs in Nebula OS, as this is handled by logical components in Nebula OS that are not in TOE scope.

Some of the isolation mechanisms described below rely not only in the TSF, but also use components of the operational environment. Specifically, the following modules of the Mediatek hardware platform participate in some isolation use cases:

- ARM TrustZone, in EL3.
- M-TEE Trustzone Extension, in EL3, composed by a Hardware Virtual Machine and the MPU. In some use cases, this MPU is used alone without collaboration from the HW VM of the M-TEE Trustzone Extension.

Following it is described the communication paths in which the isolation provided by TSF is involved, together with the components from the operational environment in the cases in which they are involved:

- HA in Trusty ↔ CA: as the Trusty virtual machine does not have operating system and is supported directly by the hypervisor, this communication does not include the Stage 1 MMU and is controlled as follows:
  - In EL2, which is done by the TSF (Stage 2 MMU).
  - In EL3, which is done by the M-TEE Trustzone Extension (Hardware Virtual Machine and MPU, components of the operational environment).
- HA ↔ TA: this communication is always made through the REE VM, so this involves all the isolation mechanisms that are made from the Nebula/Trusty to REE VM and then, from REE VM to TEE. In particular it involves:
  - In EL1 by the Stage 1 MMU (component present in Nebula OS belonging to the operational environment). This step is applicable only for HAs in Nebula.
  - In EL2, which is done by the TSF (Stage 2 MMU).
  - In EL3 by the ARM TrustZone of the underlying platform driven by the M-TEE Trustzone Extension (Hardware Virtual Machine and MPU, components of the operational environment).
- HA in Trusty ↔ HA in Trusty:
  - Managed directly in EL2 by the Stage 2 MMU.
- HA in Nebula ↔ CA: this communication is controlled by three isolation mechanisms:
  - In EL1 by the Stage 1 MMU (component present in Nebula OS belonging to the operational environment).
  - In EL2 managed by the Stage 2 MMU.

- In EL3, which is done by the M-TEE Trustzone Extension.

For the step that involves Stage 2 MMU (TOE) in EL2 working together with the M-TEE Trustzone Extension in EL3 (operational environment) that are mentioned in the above isolation cases, the description of their interaction is as follows: the application initiating the communication uses Virtual Addresses to access memory, which will be translated to Physical Addresses. If the translation of virtual to physical address fails (which implies an invalid memory access), a translation abort will occur first. Next, the VM + MPU in the M-TEE Trustzone extension checks the validity of the physical address which is translated from Stage 2 MMU.

Functionality not in scope of this Security Target

The following functionalities and isolation controls are not in the scope of this Security Target due to they do not mainly involve the TSF. Although some of them which uses the TSF, they are also not covered:

- CA ↔ TA: this communication is managed by the Hypervisor and the underlying platform, as it does not involve any other virtual machine rather than the particular for the REE VM. In this case, the communication is controlled completely by the operational environment (out of the scope) in the following way:
    - In EL3 by the ARM TrustZone of the underlying platform, and the MPU within the M-TEE Trustzone extension, both hardware modules (components of the operational environment).
- HA in Nebula ↔ HA in Nebula:
    - In EL1 by the Stage 1 MMU.
- HA in Nebula ↔ HA in Trusty: this communication is not supported.

Summary

The TOE has the ability to enforce:

- Separation between information domains: implemented as the Virtual Machines through the implementation of shared virtual or physical devices and API-based mechanisms such as hypercalls, the SMC and shared memory, which are subject to the authorization of the Stage 2 MMU (**FDP_VMS_EXT.1**, **FPT_HCL_EXT.1**) which is also in charge of forbidding the data sharing between virtual machines (**FDP_VMS_EXT.1**).
- Access control to the physical platform resources: VM's direct access to the physical devices indicated by **FDP_HBI_EXT.1** is constrained by the TSF; moreover, the TSF controls VM access to a number of physical platform resources and explicitly denies all Guest VMs access to a subset of physical platform resources as indicated in **FDP_PPR_EXT.1.** Stage 2 MMU hardware module provides this functionality.
- VMM protection through isolation: as the memory is protected by the TSF by means of the Stage 2 MMU, which controls that the running code does not have access to uncontrolled

access to unexpected memory addresses or degrade the functionality of the VMM itself (**FPT_VIV_EXT.1**).

- HAs (in Trusty VM) protection through isolation: as address spaces are protected by the TSF through the stage 2 MMU, which controls that an HA does not have access to the address spaces of another HA (both in Trusty VM), except if carried out by authorized IPC mechanisms (**FDP_HAI_EXT.1**). The mechanisms for authorized data sharing between Trusty HAs are allowed by the isolation requirements of **FDP_HAI_EXT.1** and are implemented by **FDP_VMS_EXT.1.**

8 ACRONYMS

The following table shows the acronyms used in this document.

Acronym	Meaning
AI	Artificial Intelligence
API	Application Programming Interfaces
ATF	ARM Trusted Firmware
CC	Common Criteria
CPU	Central Processing Unit
DSP	Digital Signal Processing
EAL	Evaluation Assurance Level
EL	Execution Level
GIC	Generic Interrupt Controller
GPU	Graphics Processing Unit
HA	M-TEE Hypervisor Application
IT	Information Technology
MMU	Memory Management Unit
MPU	Memory Protection Unit
M-TEE	MediaTek Trusted Execution Environment
OSP	Organisational Security Policies
PP	Protection Profile
REE	Rich/Regular Execution Environment
RTC	Real Time Clock
SMC	Secure Monitor Call
SoC	System-on-chip
ST	Security Target
TA	Trusted Application
TEE	Trusted Execution Environment
TOE	Target of Evaluation
TSF	TOE Security Functionality
TSFi	TSF Interface
VM	Virtual Machine
VMM	Virtual Machine Manager
VS	Virtualization System
SVP	Secure Video Path
UI	User Interface

Acronym	Meaning
APU	Accelerated Processing Unit
ISP	Image Signal Processor
MMuP	Multi-Media Micro Processor
MD	Modem
CONSYS	Connection System

Table 15 - Abbreviations

9 GLOSSARY OF TERMS

The following table shows the definition of the main items used in the document.

Term	Meaning
Administrator	Administrators perform management activities on the VS. These management functions do not include administration of software running within Guest VMs, such as the Guest OS. Administrators need not be human as in the case of embedded or headless VMs. Administrators are often nothing more than software entities that operate within the VM.
Auditor	Auditors are responsible for managing the audit capabilities of the TOE. An Auditor may also be an Administrator. It is not a requirement that the TOE be capable of supporting an Auditor role that is separate from that of an Administrator.
Augmentation	Addition of one or more requirement(s) to a package
Availability	Availability means having timely access to information because the entity accesing the information is authorized to do so.
Client Application (CA)	An application running outside of the Trusted Execution Environment (TEE) making use of the TEE Client API that accesses facilities provided by the Trusted Applications inside the TEE. Contrast Trusted Application.
Consistency	Is the property that ensures that in Read/Read or Write/Read operations, the data has not been changed in between unadvisely.
Device binding	The property of data being usable only on a unique given system instance, here a TEE.
Domain	A Domain or Information Domain is a policy construct that groups together execution environments and networks by sensitivity of information and access control policy. For example, classification levels represent information domains. Within classification levels, there might be other domains representing communities of interest or coalitions. In this context, the domain is implemented by each VM connected by virtual networks with the others.
End-user	The user of the device in which the TOE and the complete environment are deployed, which interacts directly with the CAs and use them to access to Rich, M-TEE and Trusted services. Typically, the owner of the mobile phone in which the TOE runs.
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
Execution Environment (EE)	An environment that hosts and executes software. In this case, the REE is based in an Android operating system and the software and applications running on it.
Execution level	The level of privileges in which the instructions are executed. Usually in the ARM architecture the execution level is divided in four levels (EL0 to EL3) where EL0 is the lowest privileged execution level (the level in which the applications are executed) and EL3 is the highest privileged level (in which the processor instructions are executed).

Term	Meaning
Guest Network	See Operational Network.
Guest Operating System	An operating system that runs within a Guest or Helper VM.
Guest Operating System (OS)	An operating system that runs within a Guest VM.
Guest VM	A Guest VM is a VM that contains a virtual environment for the execution of an independent computing system. Virtual environments execute mission workloads and implement customer-specific client or server functionality in Guest VMs, such as a web server or desktop productivity applications. In the current security target, the Guest VM is running the Guest OS implemented by the Nebula operating system.
Helper VM	A Helper VM is a VM that performs services on behalf of one or more Guest VMs, but does not qualify as a Service VM—and therefore is not part of the VMM. Helper VMs implements functions or services that are particular to the workloads of Guest VMs. For example, a VM that provides a virus scanning service for a Guest VM would be considered a Helper VM. For the purposes of this document, Helper VMs are considered a type of Guest VM, and are therefore subject to all the same requirements, unless specifically stated otherwise.
Host Operating System (OS)	An operating system onto which a VS is installed. Relative to the VS, the Host OS is part of the Platform. There need not be a Host OS, but often VSes employ a Host OS or Control Domain to support guest access to host resources. Sometimes these domains are themselves encapsulated within VMs.
Hypercall	An API function that allows VM-aware software running within a VM to invoke Hypervisor or VMM functionality.
Hypervisor	The Hypervisor is part of the VMM. It is the software executive of the physical platform of a Virtualization System. A Hypervisor's primary function is to mediate access to all CPU and memory resources, but it is also responsible for either the direct management or the delegation of the management of all other hardware devices on the hardware platform. In this security target, the hypervisor is the GenieZone platform and related software and firmware.
Information Domain	See Domain.
Integrity	Property of the TEE persistent storage that means that the value successfully read from a storage location is the last value that was written to this location.
Introspection	A capability that allows a specially designated and privileged domain to have visibility into another domain for purposes of anomaly detection or monitoring.
Management Network	A network, which may have both physical and virtualized components, used to manage and administer a VS. Management networks include networks used by VS Administrators to communicate with management components of the VS, and networks used by the VS for communications between VS components. For purposes of this document, networks that connect physical

Term	Meaning
	hosts and backend storage networks for purposes of VM transfer or backup are considered management networks.
Management Subsystem	Components of the VS that allow VS Administrators to configure and manage the VMM, as well as configure Guest VMs. VMM management functions include VM configuration, virtualized network configuration, and allocation of physical resources.
Monotonicity	The property of a variable whose value is either always increasing or always decreasing over time.
Normal World	The normal world is the execution environment, intended as the software and hardware means, to which a regular or non trustable operating system can run in the ARM TrustZone architecture. This execution environment only relies in the OS capabilities for offering confidentiality and integrity protections.
Non-secure world	See “Normal world”
Operational Environment	Environment in which the TOE is operated
Operational Network	An Operational Network is a network, which may have both physical and virtualized components, used to connect Guest VMs to each other and potentially to other entities outside of the VS. Operational Networks support mission workloads and customer-specific client or server functionality. Also called a “Guest Network.”
Physical Platform	The hardware environment on which a VS executes, in particular the Mediatek platform with ARM Trustzone capability. This physical platform includes processors, memory, devices, and associated firmware.
Platform	The hardware, firmware, and software environment into which a VS is installed and executes.
Power cycle	The lapse between the moment a device is turned on and the moment the device is turned off afterwards.
Production TEE	A TEE residing in a device that is in the end-user phase of its life cycle.
Protection Profile	Implementation-independent statement of security needs for a TOE type
REE Communication Agent	REE Regular OS driver that enables communication between the REE and the TEE. Contrast TEE Communication Agent.
Regular Execution Environment (REE, former Rich execution environment)	An Execution Environment comprising at least one Regular OS and all other components of the device (SoCs, other discrete components, firmware, and software) which execute, host, and support the Regular OS (excluding any Secure Components included in the device). From the viewpoint of a Secure Component, everything in the REE is considered untrusted, though from the Regular OS point of view there may be internal trust structures. (Formerly referred to as a Rich Execution Environment (REE).) Contrast Trusted Execution Environment.
Regular OS (former Rich OS)	An OS executing in a Regular Execution Environment. May be anything from a large OS such as Linux down to a minimal set of statically linked libraries

Term	Meaning
	providing services such as a TCP/IP stack. (Formerly referred to as a Rich OS or Device OS.) Contrast Trusted OS.
Root of Trust (RoT)	A computing engine, code, and possibly data, all co-located on the same platform; provides security services. No ancestor entity is able to provide a trustable attestation (in digest or other form) for the initial code and data state of the Root of Trust. Depending on the implementation, the Root of Trust is either a Bootstrapped or a Non-Bootstrapped Root of Trust.
Secure Component	GlobalPlatform terminology to represent either a Secure Element or a Trusted Execution Environment.
Secure Element	A tamper-resistant secure hardware component which is used in a device to provide the security, confidentiality, and multiple application environment required to support various business models. May exist in any form factor, such as embedded or integrated SE, SIM/UICC, smart card, smart microSD, etc.
Secure world	The secure world is the execution environment, intended as the software and hardware means, to which a trusted operating system can run in the ARM TrustZone architecture. This execution environment relies in the secure capabilities offered by the hardware platform to provide confidentiality and integrity protections. Secure world provides an environment that supports confidentiality and integrity, which can prevent softwareattacks.
Security Target	Implementation-dependent statement of security needs for a specific identified TOE
Service VM	A Service VM is a VM whose purpose is to support the Hypervisor in providing the resources or services necessary to support Guest VMs. Service VMs may implement some portion of Hypervisor functionality, but also may contain important system functionality that is not necessary for Hypervisor operation. As with any VM, Service VMs necessarily execute without full Hypervisor privileges—only the privileges required to perform its designed functionality. Examples of Service VMs include device driver VMs that manage access to physical devices, VMs that provide life-cycle management and provisioning of Hypervisor and Guest VMs, and name-service VMs that help establish communication paths between VMs.
System Security Policy (SSP)	The overall policy enforced by the VS defining constraints on the behavior of VMs and users
System-on-Chip (SoC)	An electronic system all of whose components are included in a single integrated circuit.
TA instance time / TA persistent time	Time value available to a Trusted Application through the TEE Internal Core API. The API offers two types of time values: (a) System Time exists only during runtime, and must be monotonic for a given TA instance. The returned value is called “TA instance time”. (b) Persistent time persists over resets, depends on the TA but not on a particular instance, and must be monotonic even across power cycles. Its monotonicity across power cycles is related to the Time and Rollback optional PP-Module.

Term	Meaning
Tamper-resistant secure hardware	Hardware designed to isolate and protect embedded software and data by implementing appropriate security measures. The hardware and embedded software meet the requirements of the latest Security IC Platform Protection Profile [PP-0084] including resistance to physical tampering scenarios described in that Protection Profile.
Target Of Evaluation	Set of software, firmware and/or hardware possibly accompanied by guidance
TEE Client API	The software interface used by clients running in the REE to communicate with the TEE and with the Trusted Applications executed by the TEE.
TEE Communication Agent	TEE Trusted OS driver that enables communication between REE and TEE. Contrast REE Communication Agent.
TEE Internal Core API	The software interface exposing TEE functionality to Trusted Applications.
TEE Service Library	A software library that includes all security related drivers.
Trusted Application (TA)	An application running inside the Trusted Execution Environment that provides security related functionality to Client Applications outside of the TEE or to other Trusted Applications inside the TEE. Contrast Client Application.
Trusted Applications Manager	Entity responsible for loading, installation, and removal of Trusted Applications.
Trusted Execution Environment (TEE)	An Execution Environment that runs alongside but isolated from an REE. A TEE has security capabilities and meets certain security-related requirements: It protects TEE assets against a set of defined threats which include general software attacks as well as some hardware attacks, and defines rigid safeguards as to data and functions that a program can access. There are multiple technologies that can be used to implement a TEE. Contrast Regular Execution Environment.
Trusted OS	An OS executing in the secure world of the architecture.
Trusted Storage	In GlobalPlatform TEE documents, storage that is protected to at least the robustness level defined for OMTP Secure Storage. It is protected either by the hardware of the TEE, or cryptographically by keys held in the TEE. If keys are used, they are at least of the strength used to instantiate the TEE. A GlobalPlatform TEE Trusted Storage is not considered hardware tamper resistant to the levels achieved by Secure Elements.
User	Users operate Guest VMs and are subject to configuration policies applied to the VS by Administrators. Users need not be human as in the case of embedded or headless VMs, users are often nothing more than software entities that operate within the VM. In terms of this Security Target, the user is intended not as the end-user, but as the integrator of the TOE (the OEM).
Virtual Machine (VM)	A Virtual Machine is a virtualized hardware environment in which an operating system may execute.

Term	Meaning
Virtual Machine Manager (VMM)	A VMM is a collection of software components responsible for enabling VMs to function as expected by the software executing within them. The VMM consists of the GenieZone Hypervisor, its VMs, and other components of the VS, such as virtual devices, binary translation systems, and physical device drivers. In particular, those in the scope of the evaluation. It manages concurrent execution of all VMs, virtualizes platform resources as needed and provides isolation capabilities in EL2. In the particular scope of this security target, the VMM can be considered as the Stage 2 MMU.
Virtualization System (VS)	A software product that enables multiple independent computing systems to execute on the same physical hardware platform without interference from one another. For the purposes of this document, the VS consists of the GenieZone Hypervisor, the Virtual Machines (VM) and the related components.

Table 16 - Glossary of terms

10 DOCUMENT REFERENCES

The following table shows the references used in this document.

Reference	Document
[BSI-CC-PP-0084-2014]	Security IC Platform BSI Protection Profile 2014 with Augmentation Packages, version 1.0, 13.01.2014
[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
[CEM31R5P3]	Common Criteria Evaluation methodology, Version 3.1, Revision 5
[GPD_SPE_007]	GlobalPlatform Device Technology TEE Client API Specification, version 1.0, July 2010
[GPD_SPE_009]	GlobalPlatform Device Technology TEE System Architecture, version 1.1, January 2017
[GPD_SPE_021]	GlobalPlatform Technology TEE Protection Profile, version 1.3, July 2020
[JIL_AAPS]	Joint Interpretation Library, Application of Attack Potential to Smartcards and Similar Devices, version 3.1, June 2020
[NIAP-PP-VIRT]	NIAP, Protection Profile for Virtualization, version 1.1, June 2021
[TEE_WP]	GlobalPlatform, The Trusted Execution Environment: Delivering Enhanced Security at a Lower Cost to the Mobile Market, June 2015

Table 17 - List of document references