SESIP Security Target for pKVM Hypervisor

Google

Version 1.3, dated 13/06/2025

Based on SESIP methodology

DOCUMENT HISTORY

Version	Date	Description
1.0	28/01/2025	Added commit TOE's version and clarification about its 64 bits compilation
1.1	12/02/2025	Changes to cover Evaluation Meeting 1
1.2	06/03/2025	Removed iteration mark from "Software Attacker Resistance: Isolation of Platform Parts", added reference to ATE deliverables and corrected reference in section 4.
1.3	13/06/2025	Minor changes to section 4

TABLE OF CONTENTS

LIST OF TABLES

1 INTRODUCTION

The Security Target describes the platform (in this chapter) and the exact security properties of the platform that are evaluated against SESIP.

This Security Target describes the exact security properties of the pKVM Hypervisor that are evaluated against the CEN/CENELEC Security Evaluation Standard for IoT Platforms as defined by [REF1].

The security functional requirements are specified in section 3.2. The assurance level that this Security Target claims for the TOE is SESIP5 and is detailed in section 3.1.

1.1 ST Reference

See Title page.

1.2 Platform Reference

Reference	Value
Platform name	pKVM Hypervisor
Platform version	pKVM version contained in android14-6.1-2024-10_r11 (commit d6f926cfde54) – 64 bits Note: Check section 1.1 in [REF5] to check if a specific kernel version contains a pKVM certified version.
Platform identification	pKVM Hypervisor version
Platform Type	Kernel-based virtual machine hypervisor for Android

Table 1. Platform Reference

1.3 Included Guidance Documents

The following documents are included with the platform:

Document	Reference
Online Developer Guide	AVF architecture [2]
Online Developer Guide	Microdroid [3]

Table 2. Guidance Documents

1.4 Platform Functional Overview and Description

1.4.1 Overview

Google pKVM is a Type-1 (bare-metal) hypervisor designed specifically for Android devices. As a Type-1 hypervisor, pKVM runs directly on the underlying hardware, below the host operating system (the Android

kernel). This allows it to manage hardware resources such as CPU, memory, and I/O devices directly, providing virtualization services without the need for an intervening host OS layer that Type-2 hypervisors rely on.

pKVM always comes together with the kernel and is included in the binary package provided by the OEM, ensuring that both the kernel and hypervisor are tightly integrated and maintained as a single unit, allowing for cohesive updates and security management.

As a Type-1 hypervisor, the TOE supports running unmodified guest operating systems within virtual machines (pVMs). The guest OS operates as if it were running on actual physical hardware, without needing modifications or special drivers.

The virtual machines are defined as *Protected Virtual Machines* (pVMs), which are isolated from both the host OS and other VMs. This ensures that even if the host OS is compromised, the integrity and confidentiality of the pVMs remain intact. The TOE is intended to be used to support the execution of operations that are critical in terms of confidentiality and integrity (e.g. medical or financial data). Executing these operations in a pVM reduces the attack surface, as a breach found in the host OS at any exception level, including the Linux kernel, does not affect the pVM security.

The TOE leverages ARM's Virtualization Extensions (VE) to enable efficient virtualization. These hardware features allow pKVM to offload critical virtualization tasks to the hardware, reducing performance overhead and improving efficiency. It utilizes *Stage-2 Translation Tables* for memory virtualization, which manage guest physical memory securely by controlling memory mappings and enforcing access permissions at the hardware level. Being a Type-1 hypervisor, it controls any access to physical memory from both the host OS and the pVMs, so neither the host OS nor the pVMs has direct access to the physical memory without the intervention of pKVM access control.

The main security features of the TOE are as follows:

- **pKVM Secure initialization:** The secure boot of the TOE is relying on the secure boot of the environment, and in particular on the Android Bootloader which verifies the integrity and authenticity of the kernel and pKVM in the same process. In the initialization process, the kernel and the pKVM are separated and given different exception levels. The Arm architecture allows up to four exception levels, with Exception Level 0 (EL0) being the least privileged, and Exception Level 3 (EL3) the most. When the system boots, the bootloader starts the generic kernel at Exception Level 2 (EL2). Recognizing that it's running at EL2, the generic kernel deliberately lowers its privilege to Exception Level 1 (EL1). Meanwhile, the pKVM (TOE) and its modules remain operational at EL2. The kernel then continues with the normal boot process, initializing all required device drivers until it reaches user space. At this point, pKVM is the only component that remains at EL2 with the higher privileges.
- **pVMs Secure boot:** In the pVM secure initialization, pKVM makes sure that the first code to run is pvmfw. The pvmfw validates the virtual platform and verifies the pVM payload images.
- **Separation of domains:** Once the secure initialization has finished, the TOE manages the stage-2 page tables from EL2 protected space. Memory isolation is achieved using Stage-2 page tables, which control memory mappings and access permissions at the hardware level, ensuring each pVM operates within its own isolated memory space without overlapping with others (see exception in the next point “Communication pVMs-Host”). Furthermore, the hypervisor leverages the register context during context switches to ensure that no residual state or information is leaked between pVMs.
- **Communication pVMs-Host:** Communication between the host and the guests is made possible by controlled memory sharing between them. Guests are allowed to share some of their pages back with the host using a hypercall, which instructs the hypervisor to remap those pages in the host stage-2 page table.

The aspect of availability of resources is out of the scope of this Security Target:

The TOE does not guarantee availability to guest virtual machines (pVMs) because it delegates critical resource management tasks to the host kernel. Specifically, pKVM relies on the host kernel to schedule the pVM's virtual CPUs and handle guest interrupts. This delegation reduces the hypervisor's trusted computing base (TCB) and allows the host to make more optimized scheduling decisions.

1.4.2 TOE scope

The Target of Evaluation (TOE) is limited exclusively to software, specifically to the pKVM component. The certified version is android14-6.1-2024-10_r11 in its 64-bit compilation and identified in the configuration management with commit d6f926cfde54.

Although pKVM is integrated within a specific version of the Android kernel, the evaluation will focus solely on aspects related to pKVM. The TOE does have unique identification within the source code repository, but as the component is part of the larger kernel, the kernel version will be used as the version number for the product as tested in the evaluation.

The TOE can hence be defined as the pKVM source code that is then compiled as part of the binary compilation of the kernel. The pKVM code is open source, which means that anyone can access the code and documentation, while any updates are strictly controlled before releases.

The TOE is the hypervisor, also referred to as pKVM, which is the virtualization technology used by Android Virtualization Framework (AVF). The hypervisor is responsible for maintaining the integrity of executed code and ensuring the confidentiality of the pVM's assets, even in scenarios where the host Android or other pVMs are compromised.

The TOE modules that make up the TOE and that are included in the evaluation are the following:

- Core Hypervisor Module: It manages the lifecycle of virtual CPUs, processes different types of exceptions, handles hypercalls, and ensures synchronization between the hypervisor and the host system.
- Memory Management Module: This module is pivotal in managing memory protection, sharing and donation mechanisms between the host and protected Virtual Machines (pVMs) within the hypervisor architecture.
- Module Management System: The module management system serves as the backbone for managing hypervisor modules, system register configurations, memory mappings during early initialization, and dynamic hypercall registrations. It ensures that the hypervisor operates smoothly by handling low-level system configurations and facilitating interactions between different modules.
- IOMMU Management Module: This module is integral to managing Input-Output Memory Management Units (IOMMUs) in the context of protected Virtual Machines (pVMs), ensuring secure and efficient device memory access. It defines the contracts that IOMMU drivers must adhere to and provides the necessary abstractions for interacting with IOMMU devices.
- PSCI (Power State Coordination Interface) Module: This module is pivotal in managing Power State Coordination Interface (PSCI) calls, ensuring they are securely and effectively relayed between the host and protected Virtual Machines (pVMs). The module is responsible for handling PSCI calls, which are standardized interfaces for power management on ARM architectures. PSCI calls facilitate operations such as CPU power state transitions, system resets, and suspend/resume functionalities.
- Fast Function ABI Module: This module handles the implementation of the interface and communication between secure and non-secure worlds

- Protected VM Module: This module handles protected virtual machine management and security features.
- Debug and Tracing: This module provides debug functionality and tracing capabilities.

pKVM is shown in Figure 1, marked with a red box. All other components are outside the scope of this evaluation.

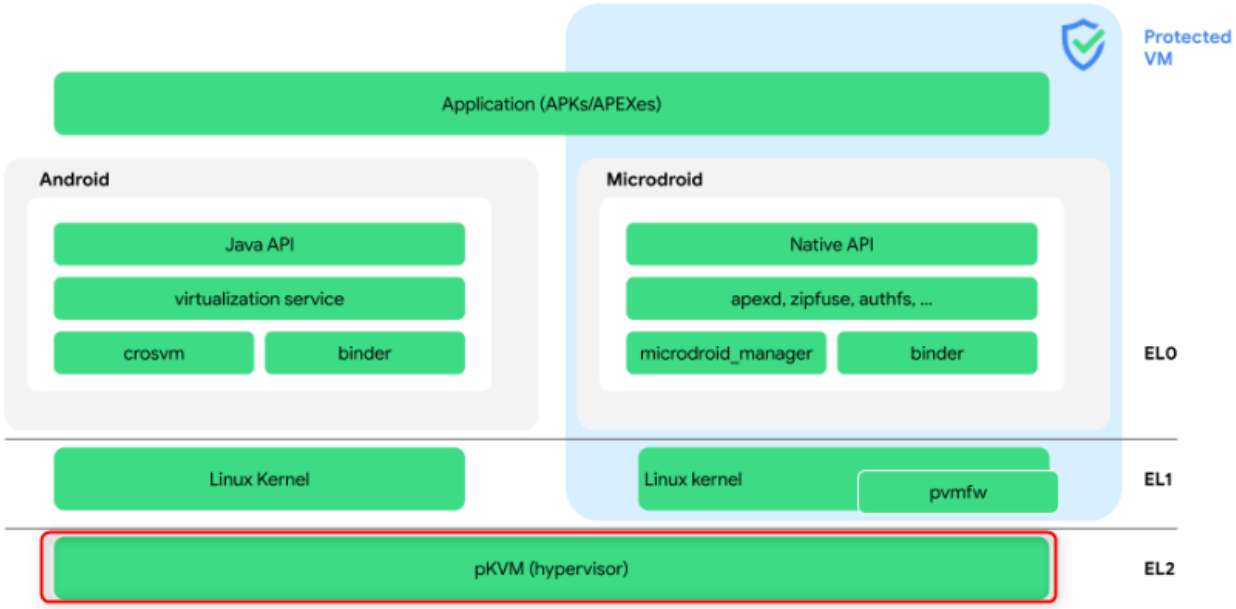

Figure 1. TOE Scope

To clarify explicit dependence, the out of scope parts comprises:

- **Linux Kernel, Android OS and Applications:** The Host and pVMs kernel (and above) operates at a lower exception level and does not implement the security functionality defined in this security target. This includes security functionality such as secure boot and secure update, where the pKVM relies on the Android Boot and update mechanisms.
- **Hardware, including processor (ARM 64):** This evaluation assumes the correct implementation of the exception level security features by the processor. Furthermore, the pKVM makes explicit use of the virtualization support provided by the ARM processor's hardware virtualization extensions, the Memory Management Unit (MMU), the Input-Output Memory Management Unit (IOMMU), and the Generic Interrupt Controller (GIC). Operating primarily at Exception Level 2 (EL2), which is designated for hypervisors in the ARM architecture, pKVM leverages these components to manage virtual machines securely and efficiently. The MMU handles virtual-to-physical address translations, ensuring memory isolation between different virtual instances. The IOMMU manages direct memory access (DMA) operations from peripherals, providing an additional layer of security by controlling device memory accesses. Meanwhile, the GIC oversees the distribution and handling of hardware interrupts, ensuring that each virtual machine receives the appropriate interrupt signals without interference. By utilizing ARM's exception levels and these specialized virtualization features, pKVM ensures robust isolation for virtualized environments on ARM hardware.
- **pKVM vendor modules:** pKVM vendor module is a hardware-specific module containing device-specific functionality, such as input-output memory management unit (IOMMU) drivers. These modules let you port security features requiring Exception Level 2 (EL2) access to pKVM. These modules are developed by vendors through the pKVM vendor module API that is a struct

encapsulating callbacks to the pKVM hypervisor. This software implemented by third parties is outside the evaluation scope.

- **pVM firmware (pvmfw):** The first code that runs on a pVM, pvmfw, verifies the payload and handles the per-VM secret.

2 SECURITY OBJECTIVES FOR THE OPERATIONAL ENVIRONMENT

2.1 Platform Objectives for the Operational Environment

For the platform to fulfil its security requirements, the operational environment (technical or procedural) shall fulfil the following objectives:

- **OE.PHYSICAL:** The platform shall only be deployed in environments where physical attacks protections are not required. Any kind of physical (e.g. hardware modification) or electromagnetic (e.g. power consumption, electromagnetic emanations) attack is not considered by the security problem definition.
  To fulfil this requirement, read the section 2.1 from the "pKVM Integrator and user guide" [REF5].
- **OE.KERNEL_BOOT:** When booting the device, and before executing the TOE, the initial boot phase is executed by the Linux kernel. When the system starts, the processor operates in EL2 (Hypervisor mode), the highest non-secure privilege level, with all interrupts disabled to prevent any unauthorized interference. As the kernel begins to initialize, it implements temporary protection mechanisms by creating protected memory regions for critical code and data, setting up memory barriers, establishing guard pages to detect stack overflows, and marking kernel code as read-only to prevent unauthorized modifications. These measures ensure that even before the hypervisor is active, the kernel maintains a secure environment against potential threats. Finally, the kernel prepares for the critical transition by validating the presence of required CPU features and security extensions, setting up separate memory spaces for the hypervisor and kernel, and configuring initial page tables to ensure memory isolation.
  To fulfil this requirement, read the section 2.2 from the "pKVM Integrator and user guide" [REF5].
- **OE.UPDATE:** The TOE is part of the Android's Linux kernel distributions. The TOE does not implement its own secure update mechanisms; it relies on the secure update mechanism of the Android kernel of which it is part. The Android kernel is updated through the Android OTA update. The Android OTA update process is designed with multiple layers of security to ensure that updates are authentic, untampered, and safely installed. Initially, the device securely downloads the OTA package via HTTP, performing partial verification of metadata like size and expected hash during the download. Before rebooting, an initial signature check of the OTA package is conducted.
  To fulfil this requirement, read the section 2.3 from the "pKVM Integrator and user guide" [REF5].
- **OE.HW:** The hardware where the platform is deployed shall provide security exception levels and memory management making it possible to implement memory and execution protections. pKVM on ARM utilizes hardware virtualization extensions, the MMU for memory isolation, the IOMMU for secure DMA operations, and the GIC for managing interrupts. Operating at Exception Level 2 (EL2), these components ensure robust isolation on ARM 64-based hardware.
  To fulfil this requirement, read the section 2.4 from the "pKVM Integrator and user guide" [REF5].
- **OE.INTEGRITY:** No modification to the TOE or addition of pKVM vendor modules is allowed.
  To fulfil this requirement, read the section 2.5 from the "pKVM Integrator and user guide" [REF5].

3 SECURITY REQUIREMENTS AND IMPLEMENTATION

3.1 Security Assurance requirements

The claimed assurance requirements package is **SESIP5** as defined [REF1] Chapter 4. The assurance requirements for this assurance level are:

Assurance Class	Assurance Families	
ASE: Security Target evaluation	ASE_INT.1	ST Introduction
	ASE_OBJ.1	Security requirements for the operational environment
	ASE_REQ.3	Listed security requirements
	ASE_TSS.1	TOE summary specification
ADV: Development	ADV_ARC.1	Security architecture description
	ADV_FSP.4	Complete functional specification
	ADV_TDS.3	Basic modular design
	ADV_IMP.2	Complete mapping of the implementation representation of the TSF
AGD: Guidance documents	AGD_OPE.1	Operational user guidance
	AGD_PRE.1	Preparative procedures
ALC: Life-cycle support	ALC_CMC.4	Production support, acceptance procedures and automation
	ALC_CMS.4	Problem tracking CM coverage
	ALC_DEL.1	Delivery procedures
	ALC_DVS.2	Sufficiency of security measures
	ALC_FLR.2	Flaw reporting procedures
	ALC_TAT.1	Well-defined development tools
ATE: Tests	ATE_COV.1	Evidence of coverage
	ATE_DPT.1	Testing: basic design
	ATE_FUN.1	Functional testing
	ATE_IND.1	Independent testing: conformance
AVA: Vulnerability Assessment	AVA_VAN.5	Advanced methodical vulnerability analysis

3.1.1 Flaw Reporting Procedure (ALC_FLR.2)

In accordance with the requirement for a flaw reporting procedure (ALC_FLR.2), including a process to report, process and handle any needed update and distribute it, the developer has defined the following procedure [4].

3.2 Security Functional Requirements

The platform fulfils the following Security Functional Requirements. This functionality has been tested by the developer following the testing considerations found in [REF4].

3.2.1 Verification of platform identity

The platform provides a unique identification of the platform, including all its parts and their versions.

Conformance rationale:

Because the TOE is a component of the kernel rather than an independent entity, its versioning is inherently tied to the kernel version itself. Accessing the Android kernel's source code repository (AOSP), it is possible to review the commit history related to pKVM.

Vendors are responsible for selecting the kernel version, and they decide when to update it to address bugs, fix vulnerabilities, and enhance performance, among other reasons.

Final users can verify the kernel version (and so infer TOE version) setting the Android Operating system via multiple methods:

- Through the User Interface, the final user can verify the kernel version in the mobile device settings.
- In debug mode and using the Android Debug Bridge (ADB) to access a shell. From the shell, the following commands can be executed to display the kernel version “cat /proc/version” or “uname -a”.

3.2.2 Privileged Access Control

The platform allows access to the pVMs' shared memory space by the Host OS only based on the pVMs' explicit requests to share memory.

Conformance rationale:

Communication between the host and pVMs is made possible by controlled memory sharing between them. pVMs are allowed to share some of their pages back with the host using a hypercall, which instructs the hypervisor to remap those pages in the host stage-2 page table.

The hypervisor is responsible for tracking ownership of all memory pages in the system and whether they're being shared or lent to other entities. Most of this state tracking is done using metadata attached to the host and guests’ stage-2 page tables, using reserved bits in the page table entries (PTEs) which, as their name suggests, are reserved for software use.

The host must ensure that it doesn’t attempt to access pages that have been made inaccessible by the hypervisor. An illegal host access causes a synchronous exception to be injected into the host by the hypervisor, which can either result in the responsible user space task receiving a SEGV signal, or the host kernel crashing. To prevent accidental accesses, pages donated to guests are made ineligible for swapping or merging by the host kernel.

3.2.3 Software Attacker Resistance: Isolation of Platform Parts

The platform provides isolation between platform parts, such that an attacker able to run code in kernels over the TOE (Host OS or pVMs) can compromise neither the confidentiality and integrity of other kernels over the TOE nor the provision of any other Security Functional Requirements.

Conformance rationale:

The TOE ensures robust memory isolation between different Protected Virtual Machines (pVMs) and the host operating system by leveraging ARM's hardware-assisted virtualization features, particularly Stage-2 page tables. These page tables play a crucial role in creating distinct virtual address spaces for each pVM, including the host. By meticulously managing memory mappings, the TOE ensures that each pVM operates within its own isolated memory region, preventing any overlap or unauthorized access between them.

Central to this memory isolation is the use of hardware-enforced access control mechanisms provided by the Memory Management Unit (MMU). The MMU works with Stage-2 page tables to enforce strict permissions on memory regions assigned to each pVM. These permissions dictate whether a pVM can read, write, or execute specific memory areas, effectively preventing any pVM from accessing or modifying the memory allocated to another pVM or the host. Any attempt by a pVM to breach these permissions triggers hardware traps, which are then handled securely by the TOE, ensuring that unauthorized access attempts are promptly blocked and do not compromise the system's security.

The TOE operates at EL2, the highest privilege level designated for hypervisors, granting it comprehensive control over lower exception levels. The stage 2 MMU is controlled by EL2 and enables the application of a second address translation on the output address of the stage 1 MMU, resulting in a physical address (PA).

The host and pVMs run at EL1. This hierarchical structure ensures that the TOE can effectively manage and enforce memory isolation by controlling access and permissions across different exception levels. By operating at EL2, the TOE can oversee and regulate the memory allocations and access rights of EL1 entities, thereby maintaining strict boundaries between the host and each pVM.

The unique exception is when the owner explicitly shares it with another pVM (see 3.2.2 Privileged Access Control).

3.2.4 Secure initialization of platform

The platform ensures its integrity and authenticity during platform initialization. If the platform integrity or authenticity cannot be ensured, the platform will go to none.

Conformance rationale:

The boot procedure trusts the bootloader to maintain the integrity of the kernel image only during early boot (OE.KERNEL_BOOT**)**. When the kernel is deprivileged, it's no longer considered trusted by the hypervisor, which is then responsible for protecting itself even if the kernel is compromised.

The TOE boot procedure is executed as follows:

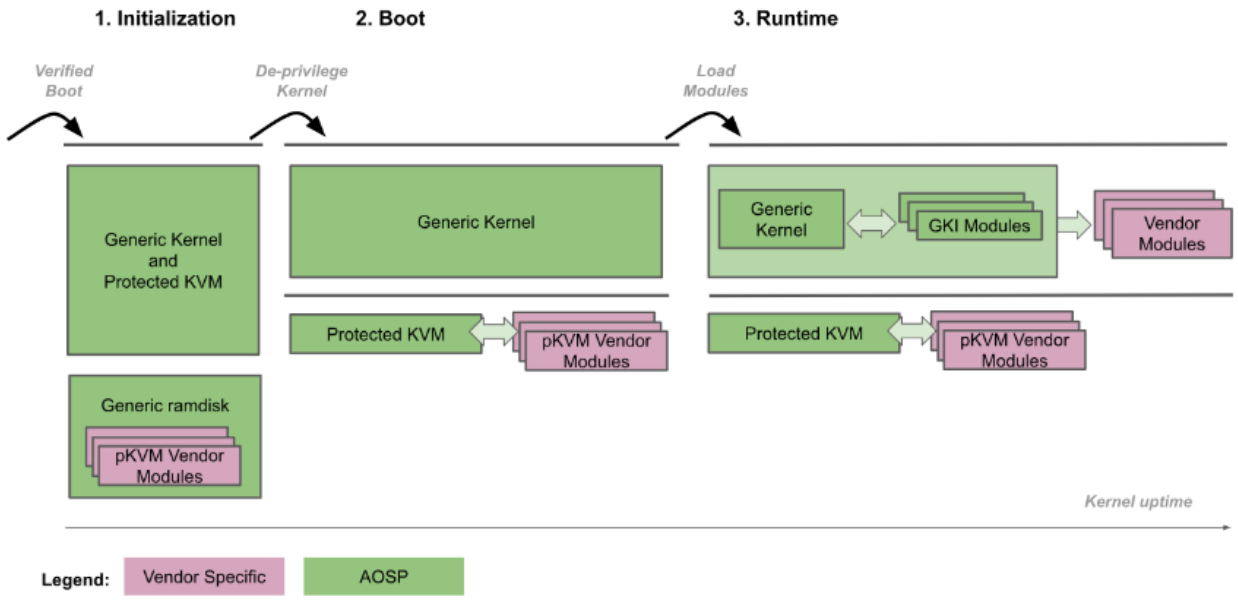

Figure 2 pKVM boot procedure

1. The bootloader enters the generic kernel at EL2.
2. The generic kernel detects that it's running at EL2 and deprivileges itself to EL1 while the TOE and its modules continue to run at EL2. Additionally, the TOE vendor modules are loaded at this time.
3. The generic kernel proceeds to boot normally, loading all necessary device drivers until reaching user space. At this point, the TOE is in place and handles the stage-2 page tables.

With this procedure the TOE is protected from the host kernel and other pVMs by the ARM exception levels, isolating the TOE from them.

The **TOE vendor modules** are hardware-specific components that enable the implementation of advanced security features within the pKVM framework.

4 MAPPING AND SUFFICIENCY RATIONALES

Assurance Class	Assurance Family	Covered by	Rationale
ASE: Security Target evaluation	ASE_INT.1 ST introduction	Section “Introduction” and title page.	The ST reference is in the Title, the platform reference in the “Platform Reference”, the platform overview and description in “Platform Functional Overview and Description”.
	ASE_OBJ.1 Security requirements for the operational environment	Section “Security objectives for the operational environment”.	The objectives for the operational environment are stated in Section 2 and further information provided in the guidance documents.
	ASE_REQ.3 Listed security requirements	Sections “Security Functional Requirements” and “Secure initialization of platform”.	All SFRs in this ST are taken from SESIP. “Verification of Platform Identity” is included.
	ASE_TSS.1 TOE summary specification	Section “Security requirements and implementation”.	All SFRs are listed per definition, and for each SFR the implementation and verification are defined in “Security Functional Requirements”.
ADV: Development	ADV_ARC.1 Security architecture description	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ADV_FSP.4 Complete functional specification	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ADV_TDS.3 Basic modular design	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.

	ADV_IMP.2 Complete mapping of the implementation representation of the TSF	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
AGD: Guidance documents	AGD_OPE.1 Operational user guidance	Section “Included Guidance Documents”	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	AGD_PRE.1 Preparative procedures	Section “Included Guidance Documents”	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
ALC: Life-cycle support	ALC_CMC.4 Production support, acceptance procedures and automation	Section “Included Guidance Documents” and material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ALC_CMS.4 Problem tracking CM coverage	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ALC_DEL.1 Delivery procedures	Section “Platform Reference” and material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ALC_DVS.2 Sufficiency of security measures	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ALC_FLR.2 Flaw reporting procedures	Section “Flaw Reporting Procedure (ALC_FLR.2)”	The flaw reporting and remediation procedure is described.
	ALC_TAT.1 Well-defined development tools	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.

ATE: Tests	ATE_COV.1 Evidence of coverage	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ATE_DPT.1 Testing: basic design	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ATE_FUN.1 Functional testing	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
	ATE_IND.1 Independent testing: conformance	Material provided to the evaluator.	The platform evaluator determines whether the provided evidence is suitable to meet the requirement.
AVA: Vulnerability Assessment	AVA_VAN.5 Advanced methodical vulnerability analysis	N.A. A vulnerability analysis is performed by the platform evaluator to ascertain the presence of potential vulnerabilities.	The platform evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be exploited in the operational environment for the platform. Penetration testing is performed by the platform evaluator assuming a High attack potential.

5 Acronyms and References

5.1 Acronyms

ST	Security Target
TOE	Target of Evaluation
SESIP	Security Evaluation Standard for IoT Platforms
TOE	Target of Evaluation
pKVM	Protected Kernel-based Virtual Machine
VM	Virtual Machine
pVM	Protected Virtual Machine
VE	Virtualization Extensions
MMU	Memory Management Unit
EL	Exception Level
AOSP	Android Open Source Project
OTA	Over-the-Air (update)
ADB	Android Debug Bridge
SEGV	Segmentation Violation (Signal)
PTE	Page Table Entry
ST	Security Target
TCB	Trusted Computing Base

5.2 References

[REF1] Security Evaluation Standard for IoT Platforms (SESIP), NEN-EN 17927:2023

[REF2] “AVF Architecture,” 28 08 2024. [Online]. Available: https://source.android.com/docs/core/virtualization/architecture.

[REF3] “Microdroid,” 28 08 2024. [Online]. Available: https://source.android.com/docs/core/virtualization/microdroid.

[REF4] "Security updates and resources," 29 09 2024. [Online]. Available: https://source.android.com/docs/security/overview/updates-resources#context_types.

[REF5] “pKVM Integrator and user guide”, version 1.2

[REF6] “PKVM Testing plan”, version 1.3