Linux Tee Documentation

Release 6.8.0

The kernel development community

Jan 16, 2026

CONTENTS

CHAPTER

ONE

TEE (TRUSTED EXECUTION ENVIRONMENT)

This document describes the TEE subsystem in Linux.

1.1 Overview

A TEE is a trusted OS running in some secure environment, for example, TrustZone on ARM CPUs, or a separate secure co-processor etc. A TEE driver handles the details needed to communicate with the TEE.

This subsystem deals with:

- Registration of TEE drivers
- Managing shared memory between Linux and the TEE
- Providing a generic API to the TEE

CHAPTER

TWO

OP-TEE (OPEN PORTABLE TRUSTED EXECUTION ENVIRONMENT)

The OP-TEE driver handles OP-TEE [1] based TEEs. Currently it is only the ARM TrustZone based OP-TEE solution that is supported.

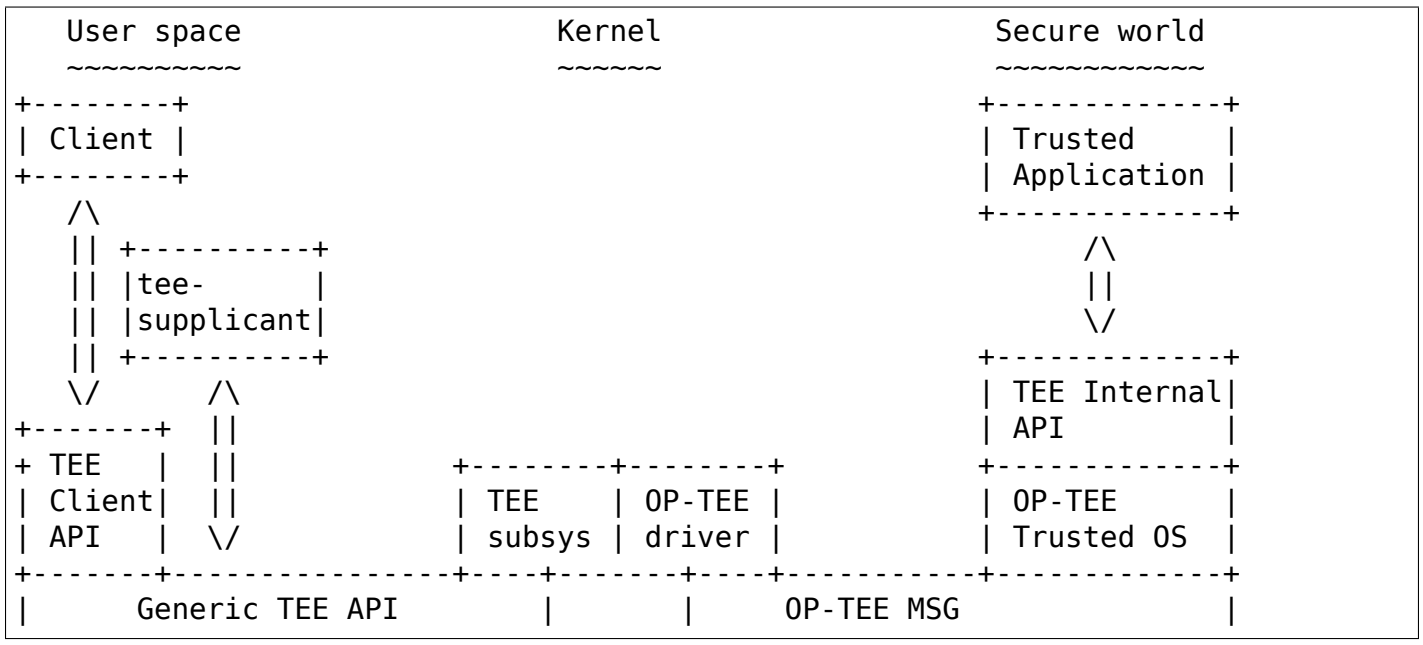
Lowest level of communication with OP-TEE builds on ARM SMC Calling Convention (SMCCC) [2], which is the foundation for OP-TEE's SMC interface [3] used internally by the driver. Stacked on top of that is OP-TEE Message Protocol [4].

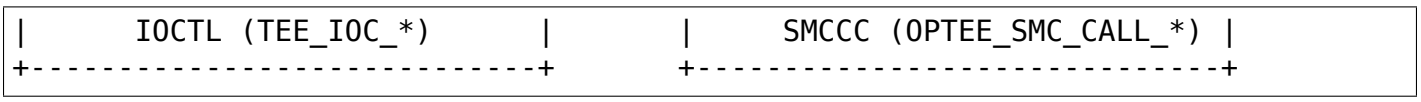
OP-TEE SMC interface provides the basic functions required by SMCCC and some additional functions specific for OP-TEE. The most interesting functions are:

- OPTEE_SMC_FUNCID_CALLS_UID (part of SMCCC) returns the version information which is then returned by TEE_IOC_VERSION
- OPTEE_SMC_CALL_GET_OS_UUID returns the particular OP-TEE implementation, used to tell, for instance, a TrustZone OP-TEE apart from an OP-TEE running on a separate secure co-processor.
- OPTEE_SMC_CALL_WITH_ARG drives the OP-TEE message protocol
- OPTEE_SMC_GET_SHM_CONFIG lets the driver and OP-TEE agree on which memory range to used for shared memory between Linux and OP-TEE.

The GlobalPlatform TEE Client API [5] is implemented on top of the generic TEE API.

Picture of the relationship between the different components in the OP-TEE architecture:





RPC (Remote Procedure Call) are requests from secure world to kernel driver or tee-supplicant. An RPC is identified by a special range of SMCCC return values from OPTEE_SMC_CALL_WITH_ARG. RPC messages which are intended for the kernel are handled by the kernel driver. Other RPC messages will be forwarded to tee-supplicant without further involvement of the driver, except switching shared memory buffer representation.

2.1 OP-TEE device enumeration

OP-TEE provides a pseudo Trusted Application: drivers/tee/optee/device.c in order to support device enumeration. In other words, OP-TEE driver invokes this application to retrieve a list of Trusted Applications which can be registered as devices on the TEE bus.

2.2 OP-TEE notifications

There are two kinds of notifications that secure world can use to make normal world aware of some event.

1. Synchronous notifications delivered with `OPTEE_RPC_CMD_NOTIFICATION` using the `OPTEE_RPC_NOTIFICATION_SEND` parameter.
2. Asynchronous notifications delivered with a combination of a non-secure edge-triggered interrupt and a fast call from the non-secure interrupt handler.

Synchronous notifications are limited by depending on RPC for delivery, this is only usable when secure world is entered with a yielding call via `OPTEE_SMC_CALL_WITH_ARG`. This excludes such notifications from secure world interrupt handlers.

An asynchronous notification is delivered via a non-secure edge-triggered interrupt to an interrupt handler registered in the OP-TEE driver. The actual notification value are retrieved with the fast call `OPTEE_SMC_GET_ASYNC_NOTIF_VALUE`. Note that one interrupt can represent multiple notifications.

One notification value `OPTEE_SMC_ASYNC_NOTIF_VALUE_DO_BOTTOM_HALF` has a special meaning. When this value is received it means that normal world is supposed to make a yielding call `OPTEE_MSG_CMD_DO_BOTTOM_HALF`. This call is done from the thread assisting the interrupt handler. This is a building block for OP-TEE OS in secure world to implement the top half and bottom half style of device drivers.

2.3 OPTEE_INSECURE_LOAD_IMAGE Kconfig option

The OPTEE_INSECURE_LOAD_IMAGE Kconfig option enables the ability to load the BL32 OP-TEE image from the kernel after the kernel boots, rather than loading it from the firmware before the kernel boots. This also requires enabling the corresponding option in Trusted Firmware for Arm. The Trusted Firmware for Arm documentation [6] explains the security threat associated with enabling this as well as mitigations at the firmware and platform level.

There are additional attack vectors/mitigations for the kernel that should be addressed when using this option.

1. Boot chain security.
   - Attack vector: Replace the OP-TEE OS image in the rootfs to gain control of the system.
   - Mitigation: There must be boot chain security that verifies the kernel and rootfs, otherwise an attacker can modify the loaded OP-TEE binary by modifying it in the rootfs.
2. Alternate boot modes.
   - Attack vector: Using an alternate boot mode (i.e. recovery mode), the OP-TEE driver isn't loaded, leaving the SMC hole open.
   - Mitigation: If there are alternate methods of booting the device, such as a recovery mode, it should be ensured that the same mitigations are applied in that mode.
3. Attacks prior to SMC invocation.
   - Attack vector: Code that is executed prior to issuing the SMC call to load OP-TEE can be exploited to then load an alternate OS image.
   - Mitigation: The OP-TEE driver must be loaded before any potential attack vectors are opened up. This should include mounting of any modifiable filesystems, opening of network ports or communicating with external devices (e.g. USB).
4. Blocking SMC call to load OP-TEE.
   - Attack vector: Prevent the driver from being probed, so the SMC call to load OP-TEE isn't executed when desired, leaving it open to being executed later and loading a modified OS.
   - Mitigation: It is recommended to build the OP-TEE driver as builtin driver rather than as a module to prevent exploits that may cause the module to not be loaded.

2.3.1 References

[1] https://github.com/OP-TEE/optee_os

[2] http://infocenter.arm.com/help/topic/com.arm.doc.den0028a/index.html

[3] drivers/tee/optee/optee_smc.h

[4] drivers/tee/optee/optee_msg.h

[5] http://www.globalplatform.org/specificationsdevice.asp look for ”TEE Client API Specification v1.0” and click download.

[6] https://trustedfirmware-a.readthedocs.io/en/latest/threat_model/threat_model.html

CHAPTER

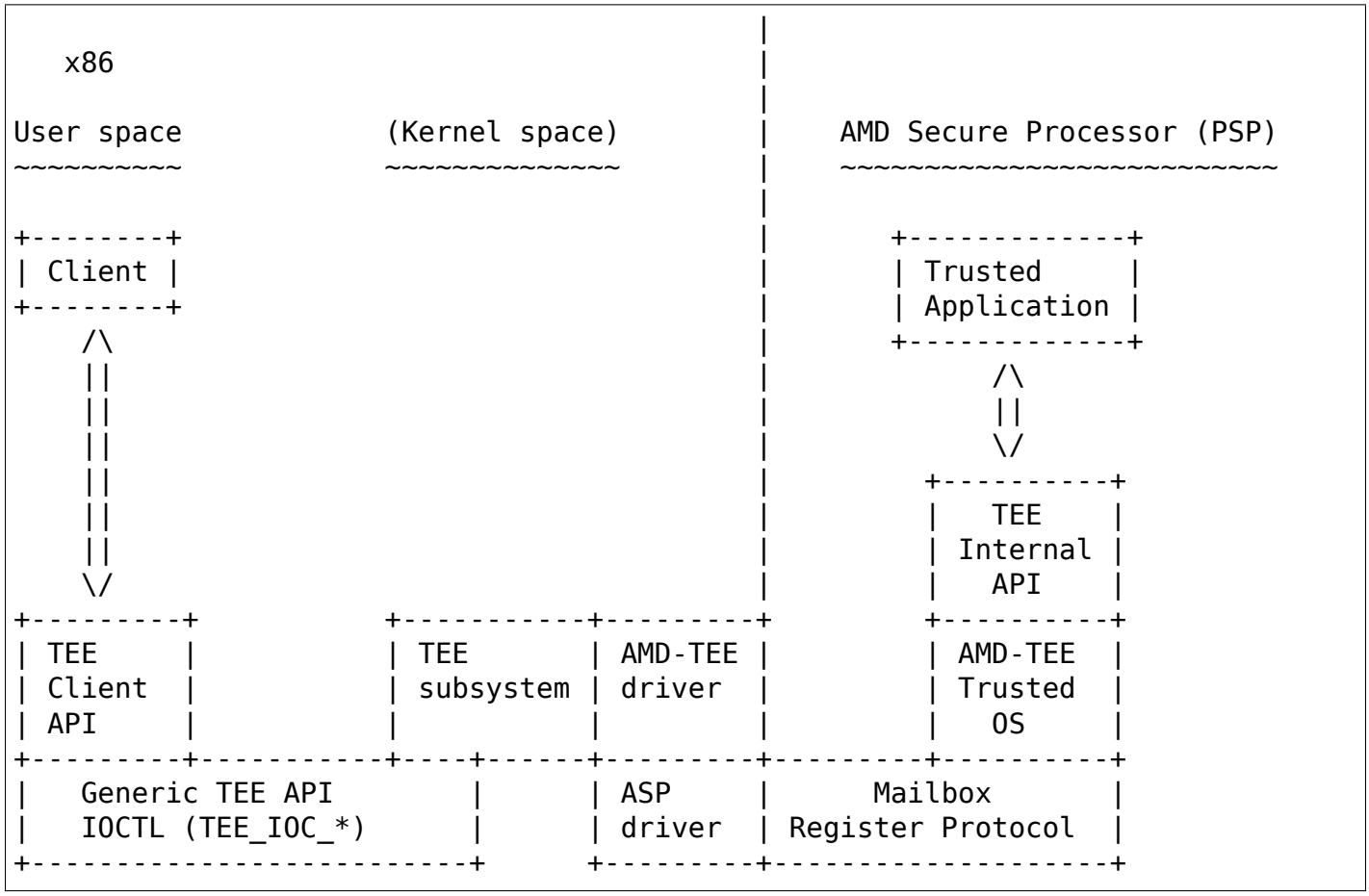
THREE

AMD-TEE (AMD'S TRUSTED EXECUTION ENVIRONMENT)

The AMD-TEE driver handles the communication with AMD's TEE environment. The TEE environment is provided by AMD Secure Processor.

The AMD Secure Processor (formerly called Platform Security Processor or PSP) is a dedicated processor that features ARM TrustZone technology, along with a software-based Trusted Execution Environment (TEE) designed to enable third-party Trusted Applications. This feature is currently enabled only for APUs.

The following picture shows a high level overview of AMD-TEE:



At the lowest level (in x86), the AMD Secure Processor (ASP) driver uses the CPU to PSP mailbox register to submit commands to the PSP. The format of the command buffer is opaque to the ASP driver. It's role is to submit commands to the secure processor and return results to AMD-TEE driver. The interface between AMD-TEE driver and AMD Secure Processor driver can be found in [1].

The AMD-TEE driver packages the command buffer payload for processing in TEE. The command buffer format for the different TEE commands can be found in [2].

The TEE commands supported by AMD-TEE Trusted OS are:

- **TEE_CMD_ID_LOAD_TA - loads a Trusted Application (TA) binary into** TEE environment.
- TEE_CMD_ID_UNLOAD_TA - unloads TA binary from TEE environment.
- TEE_CMD_ID_OPEN_SESSION - opens a session with a loaded TA.
- TEE_CMD_ID_CLOSE_SESSION - closes session with loaded TA
- TEE_CMD_ID_INVOKE_CMD - invokes a command with loaded TA
- TEE_CMD_ID_MAP_SHARED_MEM - maps shared memory
- TEE_CMD_ID_UNMAP_SHARED_MEM - unmaps shared memory

AMD-TEE Trusted OS is the firmware running on AMD Secure Processor.

The AMD-TEE driver registers itself with TEE subsystem and implements the following driver function callbacks:

- get_version - returns the driver implementation id and capability.
- open - sets up the driver context data structure.
- release - frees up driver resources.
- open_session - loads the TA binary and opens session with loaded TA.
- close_session - closes session with loaded TA and unloads it.
- invoke_func - invokes a command with loaded TA.

cancel_req driver callback is not supported by AMD-TEE.

The GlobalPlatform TEE Client API [3] can be used by the user space (client) to talk to AMD's TEE. AMD's TEE provides a secure environment for loading, opening a session, invoking commands and closing session with TA.

3.1 References

[1] include/linux/psp-tee.h

[2] drivers/tee/amdtee/amdtee_if.h

[3] http://www.globalplatform.org/specificationsdevice.asp look for ”TEE Client API Specification v1.0” and click download.