You:

https://github.com/OP-TEE/optee_os/pull/3973

ChatGPT:

This comment is discussing the implementation of secure variable storage for UEFI variables in OP-TEE, which is critical for enabling and protecting UEFI Secure Boot. The implementation involves adding a user mode secure partition context and support for loading Standalone MM of EDK2 into it. There are several changes made to the code, addressing comments and suggestions from a previous version, and some additional improvements.

The main points of the comment:

1. Secure variable storage for EFI variables is critical for enabling and protecting UEFI Secure Boot.
2. The implementation uses a Standalone MM of EDK2, which has several advantages, including reusing well-debugged applications, inheriting Fault Tolerant Writes (FTW) functionality, and being a better choice considering the FFA changes of the future Arm architectures.
3. A separate syscall handling is added to serve as a different kind of ABI and syscall IDs.
4. CFG_STMM_PATH is assigned the path of BL32_AP_MM.fd.
5. A repository is provided for testing, which contains necessary code to compile and test the implementation on QEMU.
6. A list of changes made since the previous version (RFC) and version 1 is provided.

Overall, the comment presents the implementation of secure variable storage for EFI variables and addresses previous comments and suggestions, providing a comprehensive overview of the changes made.

You:

What is StMM?

ChatGPT:

StMM stands for Standalone MM, which refers to the Standalone Memory Manager in the EDK2 (UEFI Development Kit 2) ecosystem. EDK2 is a development environment for creating UEFI (Unified Extensible Firmware Interface) firmware, and the Standalone MM is a component within this environment.

Standalone MM is designed to provide a minimal execution environment that can run independent of the traditional UEFI firmware. It offers memory management, security, and communication services to the applications running within it. This makes it a suitable option for implementing secure services, such as secure variable storage, without the need to rewrite applications as Trusted Applications (TAs) in a Trusted Execution Environment (TEE) like OP-TEE.

In the context of OP-TEE and secure variable storage, StMM can be used in cooperation with OP-TEE RPMB APIs to store UEFI variables in a secure storage, leveraging its built-in fault-tolerant write functionality to protect variables against corruption during writing.

You:

What is FFA?

ChatGPT:

FFA stands for the Firmware Framework for Arm, which is a specification introduced by Arm to standardize the interfaces and communication mechanisms between different firmware components running on Arm-based systems. The FFA specification is designed to enable a more secure, robust, and efficient firmware ecosystem on Arm platforms.

FFA provides a standardized interface for communication and resource sharing among different components, such as:

1. The Normal World (NW) and the Secure World (SW)
2. Different Secure Partitions (SPs) within the Secure World
3. The Secure Partition Manager (SPM) and the Secure Payload Dispatcher (SPD)

By standardizing these interfaces, FFA aims to simplify the development of secure firmware solutions and promote interoperability between different firmware components, regardless of their source or implementation. The FFA specification has been increasingly adopted in the Arm ecosystem, with components like OP-TEE and EDK2 working to integrate support for FFA-based communication and management.

You:

Why are SPD and SPM mutually exclusive?

ChatGPT:

SPD (Secure Payload Dispatcher) and SPM (Secure Partition Manager) are considered mutually exclusive because they represent two different approaches to managing secure environments within ARM-based systems. These approaches are not designed to be used simultaneously, as they have different roles and responsibilities in the system.

SPD is a component within ARM Trusted Firmware (ATF) that manages the lifecycle of a single Secure Payload, such as OP-TEE, and handles interactions between the Normal World (NW) and Secure World (SW). It primarily focuses on managing a single secure environment and ensuring isolation between the two worlds.

On the other hand, SPM is a more recent innovation in ARM's security architecture that manages multiple Secure Partitions (SPs) within the Secure World. It is responsible for the lifecycle management, communication, and context management of these partitions. SPM enables more flexible and granular security solutions, allowing for a standardized approach to managing multiple isolated execution environments within the Secure World.

In summary, SPD and SPM are mutually exclusive because they represent different stages in the evolution of ARM's security architecture, and they cannot be used together due to their differing roles and responsibilities in managing secure environments. While SPD focuses on managing a single secure payload, SPM is designed to manage multiple Secure Partitions, offering a more sophisticated and flexible approach to security in ARM-based systems.

You:

What is BL32_AP_MM.fd?

ChatGPT:

BL32_AP_MM.fd is a firmware binary file related to the Standalone Memory Manager (StMM) in the EDK2 (UEFI Development Kit 2) ecosystem. It represents the compiled StMM binary that can be loaded and executed in a secure environment, such as OP-TEE or a Secure Partition.

In the context of the implementation described in the earlier comment, CFG_STMM_PATH is assigned the path of BL32_AP_MM.fd. This configuration allows the OP-TEE environment to load the StMM binary into a secure partition context and execute it.

The Standalone Memory Manager (StMM) provides a minimal execution environment for running security-sensitive applications independent of the traditional UEFI firmware. It offers memory management, security, and communication services to the applications running within it. In the case of secure variable storage, StMM is utilized in cooperation with OP-TEE RPMB APIs to store UEFI variables in secure storage, while leveraging its built-in fault-tolerant write functionality to protect variables against corruption during writing.

You:

Does OP-TEE require UEFI?

ChatGPT:

No, OP-TEE (Open Portable Trusted Execution Environment) does not require UEFI (Unified Extensible Firmware Interface) to function. OP-TEE is an open-source Trusted Execution Environment (TEE) implementation that provides a secure environment for running Trusted Applications (TAs) separate from the normal operating system. It is designed to work with various types of firmware and bootloaders, not limited to UEFI.

UEFI is a modern firmware interface designed to replace the traditional BIOS (Basic Input/Output System) in personal computers and other devices. While OP-TEE can work with UEFI-based systems, it can also function with other firmware and bootloader solutions, such as U-Boot or Coreboot.

In some cases, OP-TEE and UEFI might be used together to provide a more comprehensive security solution. For example, when implementing secure variable storage for UEFI variables, OP-TEE can work alongside EDK2's Standalone Memory Manager (StMM) to store these variables in a secure storage. However, this is a specific use case, and OP-TEE itself does not inherently require UEFI to operate.