Enable UEFI Secure Boot Using OP-TEE as Secure Partition

Sahil Malhotra (sahil.malhotra@linaro.org)
Ilias Apalodimas (ilias.apalodimas@linaro.org)
Agenda

- UEFI Overview
- UEFI Secure Boot Overview
- Authenticated Variables in UEFI Secure Boot
- Authenticated Variables Requirement
- Management Mode (MM)
- Secure Partition on ARM v8 systems
- Secure Storage Service in UEFI
- Problem in the current implementation
- Solution: Secure Partition in OP-TEE OS
The **Unified Extensible Firmware Interface (UEFI)** is a specification that defines a software interface between an operating system and platform firmware.

UEFI replaces the legacy Basic Input/Output System (BIOS) firmware interface.

Intel developed the original **Extensible Firmware Interface (EFI)** specifications.

In 2005, UEFI deprecated EFI 1.10 (final release of EFI).

The latest UEFI specification, version 2.8, was approved in March 2019.
UEFI Secure Boot Overview

- UEFI Secure boot is a verification mechanism for ensuring that code launched by the device firmware is trusted and that each efi payload loaded is validated.

- What is covered by UEFI Secure Boot:
  - Next boot stage (kernel, GRUB etc).
  - Authenticated Variable Store (Which stores all keys, certificates, hashes for authenticating the EFI payloads).

= Signed Image
Authenticated Variables in UEFI Secure Boot

- According to the UEFI Specs, Authenticated Variables are one of the building blocks of UEFI Secure Boot.
- The keys, certificates, hashes that are required for UEFI Secure Boot are stored as the Authenticated variables on the secure flash device.
- There are set of predefined authenticated variables that need to be programmed for enablement and working of UEFI secure boot.

**Secure Flash**

- **Platform Key (PK)**: Establishes trust relationship between Platform Owner (PO) and Platform Firmware (PF).
  - PK: PKpriv owned by PO
  - PKpub enrolled into PF

- **Key Exchange Key (KEK)**: Establishes trust relationship between Software Vendor (SV) & Platform Firmware (PF).
  - Different KEKpriv for each SV
  - Each SV enrolls KEKpub into PF

- **db**:
  - db will contain a list of public keys that are from sources that are considered to be secure.
  - db can also hold the hash of binaries that are considered to be secure.

- **dbx**:
  - These are basically anti-db keys. dbx holds the key sources and hashes that are blacklisted from being secure or marked as malware.
Authenticated Variables Requirement

● An authenticated variable implementation requires an isolated execution environment to do the authentication and update variables.

● If there is no isolated environment, malicious code can bypass authentication check and update variable region directly.

● Now a days to fulfill this requirement, most CPU vendors are providing a kind of hardware partition which provides an isolated execution environment for high security sensitive use cases.
Management Mode (MM)

- Management Mode (MM) is a generic term used to describe a secure isolated execution environment provided by the CPU.
  - For x86 systems, this can be implemented with System Management Mode (SMM).
  - For ARM systems, this can be implemented with TrustZone (TZ).

- Two Types of MM supported by UEFI PI Specification
  - Traditional Mode - MM execution environment is setup during DXE phase
  - Standalone Mode - MM execution environment can be setup during or prior to SEC phase. An MM standalone mode driver can only run in the MM environment which provides an isolated environment.

- Standalone MM is ideal for securing EFI variables required for UEFI secure boot
- Allow access to the secure non-volatile storage only via code executing in MM mode
Secure Partition on ARM v8 systems

- A Secure Partition is an unprivileged software sandbox environment running in the Secure World, under the control of privileged software.
- SPM currently implemented in TF-A at EL3
- Shim layer to relay SVC from S-EL0 into SMC to EL3
- Enable early development of SPM code before S-EL2 availability in ARM v8.4 architecture.
- It can be used to instantiate PI Standalone Management Mode, in order to execute MM (secure) services.
Secure Storage service in UEFI

Normal World

EL0
- App1
- App2

EL1
- OS Kernel

EL2
- UEFI Firmware
  - VariableSmmRuntimeDxe
    - Get/SetVariable()
- MM Interface

Secure World

Secure Partition Manager
- Trusted Firmware - A

Secure Partition Manager (SP)
- StMM
- VariableSmm
- FlashDriver
- AuthVariableLib

UEFI Variables in Secure Storage
- Keys
- SignatureDB

UEFI Variables in Secure Storage
- PK
- KEK
- db/dbx

UEFI Variables in Secure Storage
- S-EL0
- S-EL1
- S-EL2
- S-EL3

R/W

- Get/SetVariable()
Problem in current implementation

- **SPD or SPM:**
  - SPD and SPM are mutually exclusive
  - SPD is required for communication between Normal world and Secure OS.
  - SPM instead takes directly care of all lifecycle of SPs (at EL3 today, potentially at S-EL2 in future evolutions)

- Secure OS in server market had limited usage up to now, but starting with secure device onboarding, it has great usage on IoT/Edge devices.
Solution: Secure Partition in OP-TEE

Normal World

- EL0
  - App1
  - App2
- EL1
  - OS Kernel
- EL2
  - UEFI Firmware
    - Get/SetVariable()
  - MM Interface
  - OP-TEE Interface

Secure World

- Trusted Application
- StMM
- OP-TEE OS
- Secure Partition Dispatcher
- Trusted Firmware - A

UEFI Variables in Secure Storage:
- Keys
- SignatureDB
What’s done and next steps

- This setup is tested on NXP LX2160A-RDB board.
- Currently support for Secure Partition with StandAlone MM is being upstreamed in the OP-TEE.
  - [https://github.com/OP-TEE/optee_os/pull/3973](https://github.com/OP-TEE/optee_os/pull/3973)
- We created a new Dxe driver MmcommunicationOpteeDxe which is using the OpteeLib
- We have added support for saving the variables on I2C EEPROM and eMMC RPMB in StandAlone MM.
- We are in the process of upstreaming all code changes to EDK2 also and patches will be available soon.
Thank you

Accelerating deployment in the Arm Ecosystem
References

- [https://www.slideshare.net/linaroorg/lca14-105-uefisecureboot](https://www.slideshare.net/linaroorg/lca14-105-uefisecureboot)