presented by

Google

Creating an EDK2 Firmware Image With an Embedded Application

UEFI Fall 2023 Developers Conference & Plugfest
October 9-12, 2023
Presented by Mikolaj Lisik (Google)

Agenda

- Introduction
- What is AMD SEV-SNP and how Does it Help Attestation
- Embedding a UEFI App Into the Firmware Image
- Booting the UEFI App From ROM
- Questions

Introduction

What is SEV-SNP and How Does it Help Attestation?

What is Confidential Computing

"Confidential computing is a security and privacy-enhancing computational technique focused on protecting data in use"

Source - https://en.wikipedia.org/wiki/Confidential_computing

What is SEV-SNP

- SEV - Secure Encrypted Virtualization - Memory Protection
- SEV-ES - SEV-Encrypted State -
  - Register Protection
- SEV-SNP - SEV-Secure Nested Paging - Integrity Protection

What is Attestation

"The process of validating the integrity of a computing device such as a server needed for trusted computing"

Source - https://en.wikipedia.org/wiki/Attestation

But How Does AMD SEV-SNP Help Attestation?

To support remote attestation, the AMD SNP platform measures the initial workload loaded to memory

Adding a UEFI App to the ROM

Basic File Terminology

DSC - Description File

FDF - Flash Description File

INF - Information File

How an App Gets Added

It needs to get added to the DSC file:

```
  }
    MdeModulePkg/Logo/LogoDxe.inf
    MdeModulePkg/Application/UiApp/UiApp.inf {
      <LibraryClasses>

NULL|MdeModulePkg/Library/DeviceManagerUiLib/DeviceManagerUiLib.inf
        NULL|MdeModulePkg/Library/BootManagerUiLib/BootManagerUiLib.inf

NULL|MdeModulePkg/Library/BootMaintenanceManagerUiLib/BootMaintenance
ManagerUiLib.inf
!ifdef $(CSM_ENABLE)
        NULL|OvmfPkg/Csm/LegacyBootManagerLib/LegacyBootManagerLib.inf
        NULL|OvmfPkg/Csm/LegacyBootMaintUiLib/LegacyBootMaintUiLib.inf
!endif
    }

    OvmfPkg/QemuKernelLoaderFsDxe/QemuKernelLoaderFsDxe.inf {
```

How an App Gets Added

It needs to get added to the FDF file:

```
(...)
INF  MdeModulePkg/Universal/BdsDxe/BdsDxe.inf
INF  MdeModulePkg/Application/UiApp/UiApp.inf
INF
OvmfPkg/QemuKernelLoaderFsDxe/QemuKernelLoaderFsDxe.
inf
(...)
```

What if the Applications Source Code is not Buildable in EDK2?

What if the Applications Source Code is not Buildable in EDK2?

```
[Defines]
  INF_VERSION                    = 0x00000001
  BASE_NAME                      = BuiltinApp
  FILE_GUID                      = 342114AA-
6030-4FFD-A77C-876A414E58F3
  MODULE_TYPE                    =
UEFI_APPLICATION
  VERSION_STRING                 = 1.0

[Binaries.X64]
  PE32|BuiltinApp.efi
```

Creating a Custom FDF File Summary

- Create a custom build rule that takes an application as a parameter
- Build and copy the application as *BuiltinApp.efi* in a predefined location in EDK2
- Trigger the custom EDK2 build rule

Booting the UEFI App from ROM

Booting the UEFI App from ROM - Options

- Creating a filesystem inside the ROM and copying an app there
- Using UEFI variables
- Rewriting the BDS phase

Creating a Filesystem Inside the ROM and Copying an App There

Dropped due to better alternatives

Using UEFI Variables

A UEFI variable can be created in the format of

```
Fv(3CD7F9D4-9667-49E1-B41B-C7CF0C4243D8)/
   FvFile(342114AA-6030-4FFD-A77C-876A414E58F3)
```

The first guid signals the firmware image as the source. It should be taken from the .fdf files:

```
FvNameGuid          = 3CD7F9D4-9667-49E1-B41B-C7CF0C4243D8
```

The second guid is what was assigned to the application.

Using UEFI Variables

Positives:

- Simple to set up, only a new variable needs to get added

Negatives:

- Less secure, the system can accidentally boot into something it shouldn’t
- Variables in OVMF can be edited by external factors, the solution isn’t entirely self contained

Rewriting the BDS Phase

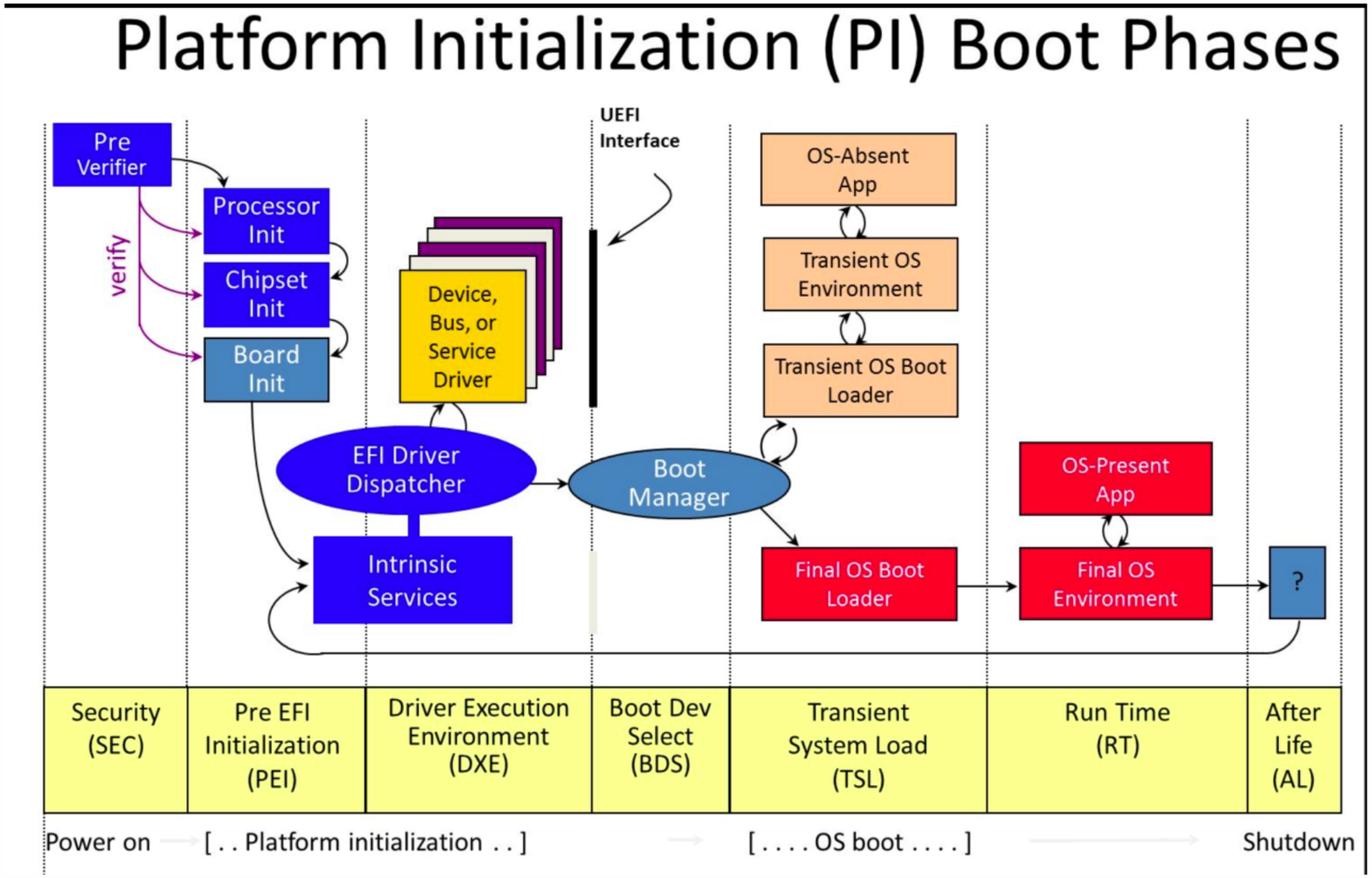

Image Courtesy of UEFI Forum

Rewriting the BDS Phase

We could copy the entire BDS phase. The files that need to be duplicated are located in

OvmfPkg/Library/PlatformBootManagerLib/

Rewriting the BDS Phase

In order to boot directly into the new app located in the formware image the new BDS implementation will simply need to contain a reference to:

```
//
// Register the new app
//
PlatformRegisterFvBootOption (
  &gBuiltinAppGuid, L"Built in App Bootloader",
LOAD_OPTION_ACTIVE
  );
```

Rewriting the BDS Phase - Downsides

Positives:

- Full control over the boot process. Each UEFI workload will always boot only into the specified app
- The solution is entirely self contain and does not depend on any external factors
- Ease of making additional changes (e.g. deletion of the UI app)

Negatives:

- Larger maintenance cost

Questions

Thanks for attending the UEFI Fall 2023 Developers Conference & Plugfest

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by

Google