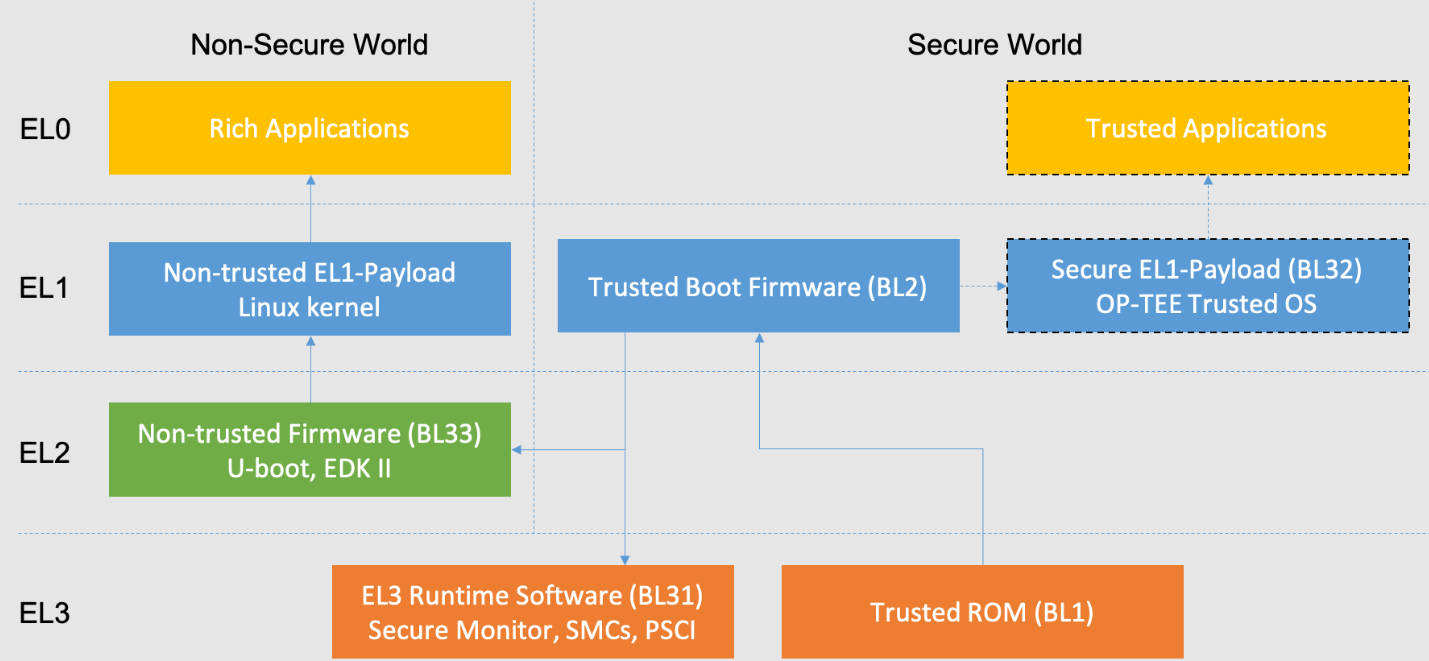
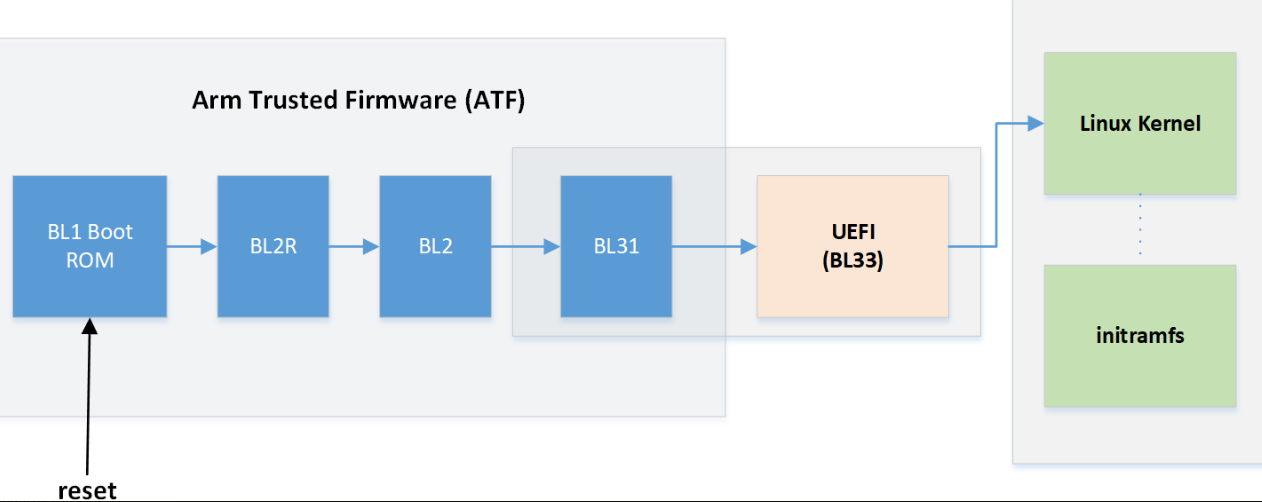
**Boot Stages and Execution Levels**:

* The boot process involves multiple stages, each executing at different privilege levels:
* **EL3 (Exception Level 3)**: The highest privilege level, where ATF (ARM Trusted Firmware) operates.
* **EL2 (Exception Level 2)**: Used for virtualization (optional).
* **EL1 (Exception Level 1)**: Kernel mode.
* **EL0 (Exception Level 0)**: User mode.





* ARM has developed software to standardize interactions between the non-secure and the secure world and made it open source.

The components part of the ATF are BL1, BL2 and BL31.

**BL1** initializes the primary CPU, **BL2** loads BL31 (EL3 Runtime Software), and **BL31** provides essential runtime services

A typical system will consist of five components:

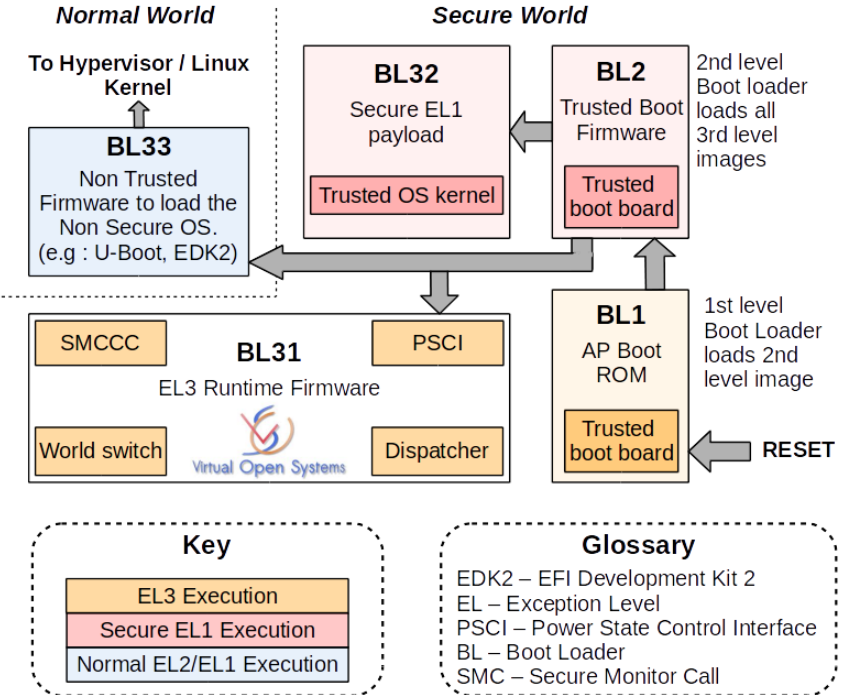
1. Boot Loader stage 1 (**BL1**) AP Trusted ROM
2. Boot Loader stage 2 (**BL2**) Trusted Boot Firmware
3. Boot Loader stage 3-1 (**BL31**) EL3 Runtime Software
4. Boot Loader stage 3-2 (**BL32**) Secure-EL1 Payload (optional) … ex: OPTEE-OS , ProvenCore-OS
5. Boot Loader stage 3-3 (**BL33**) Non-trusted Firmware

**Boot chain procedure**

* ARM has introduced four exception levels for their 64 bit v8 cores with privilege levels ranked as:

EL0 < EL1 < EL2 <EL3

* EL0 runs at the lowest exception level EL3 runs at the highest exception level



**FLOW :**

The first component to begin execution during a cold boot is the BL1 which runs in EL3 .

* BL1 **loads** BL2 which runs in secure EL1.
* BL2 **performs loading** of BL31, BL32 and BL33 and hands the load address of BL31 to BL1.
* Since BL31 runs in EL3 it must be invoked by another EL3 component like BL1.
* BL31 then **initializes** the BL32 which is the OPTEE OS in this case and runs in secure EL1.

It also **triggers execution** of BL33 which can be an OS or hypervisor.

BL33 runs in EL2 in case of a hypervisor and EL1 in case of a Normal OS like Linux.

BL31 is responsible for handling secure monitor call for interactions between secure and non-secure world.

It also handles PSCI (Power System Control Interface) operations.

**Cold Boot Path**

* The cold boot path is the sequence of steps that occur when a platform is **physically powered on**.

**Here’s how it unfolds:**

* **Boot Loader stage 1 (BL1)**:
* BL1 is responsible for **initializing the primary CPU** and performing essential platform-specific initialization.
* It sets the groundwork for subsequent stages.
* **AP Trusted ROM (BL2):**
* After BL1, BL2 executes.
* BL2’s primary task is to **load BL31 (EL3 Runtime Software)** into trusted SRAM.
* It ensures that the necessary components are ready for secure execution.
* **Trusted Boot Firmware (BL31)** (EL3 Runtime Software):
* BL31 operates at **Exception Level 3 (EL3)**, the highest privilege level in ARM systems.
* Its critical functions include:
  + **Power Management**: BL31 implements the **Power State Coordination Interface (PSCI)**, allowing normal world software to request power management actions (e.g., secondary CPU boot, hotplug, idle)

via **SMC (Secure Monitor Call)** instructions.

* + **Interrupt Management**: BL31 handles interrupts generated in either security state.
  + **Translation Tables**: BL31 sets up and manages translation tables.
  + **Secure Services**: Provides essential runtime services accessible via SMC instructions.
* **Secure-EL1 Payload (BL32)** (optional):
* If needed, BL32 executes as a secure payload.
* It can handle additional secure tasks specific to the platform.
* **Non-trusted Firmware (BL33)**:
* BL33 is the final stage.
* It loads the **non-trusted firmware**, such as the operating system, into memory.

**Boot Loader stage 2 Trusted Boot Firmware (BL2)**

* **BL2 performs the following platform initialization**

-  Initialize the console.

-  Configure any required platform storage to allow loading further boot-loader

   images.

-  Enable the MMU and map the memory it needs to access.

-  Perform platform security setup to allow access to controlled components.

-  Reserve some memory for passing information to the next boot-loader image

   EL3 Runtime Software and populate it.

-  Define the extents of memory available for loading each subsequent

   boot-loader image.

-  If BL1 has passed TB\_FW\_CONFIG dynamic configuration file in ``arg0``,

   then parse it.

* **Image loading in BL2**
* BL2 generic code loads the images based on the list of loadable images provided by the platform.
* BL2 passes the list of executable images provided by the platform to the next handover BL image.
* The list of loadable images provided by the platform may also contain dynamic configuration files.
* The files are loaded and can be parsed as needed in the ``bl2\_plat\_handle\_post\_image\_load()`` function.
* These configuration files can be passed to next Boot Loader stages as arguments by updating the corresponding entry-point information in this function.
* **EL3 Runtime Software image load**
* BL2 **loads** the EL3 Runtime Software image from platform storage into a platform-specific address in trusted SRAM.
* If there is not enough memory to load the image or image is missing it leads to an assertion failure.

**Why ProvenCore OS With Linux**

* ProvenCore OS serves as a Trusted Execution Environment (**TEE**) designed to enhance security in IoT ecosystems by providing a secure operating system that can run critical security services.
* While Linux is a robust and widely-used operating system, ProvenCore OS offers specific advantages in terms of security and trustworthiness.

**Why Use Both?**

**Enhanced Security**:

* Linux OS, while highly versatile and customizable, may not meet the stringent security requirements needed for certain applications, especially those involving sensitive data or critical operations.
* ProvenCore is designed with a strong focus on security, including features like formal verification, which proves its resilience against a wide range of vulnerabilities. By running sensitive tasks in ProvenCore, you significantly reduce the risk of security breaches.

**Isolation**: Using a TEE like ProvenCore alongside Linux allows for a clear separation between the general-purpose operating system (Linux) and the secure OS (ProvenCore). This isolation ensures that even if Linux is compromised, the components running within ProvenCore remain secure, protecting critical functions and data.

**Compliance**: Certain industries and applications require adherence to rigorous security standards and regulations. ProvenCore's design and implementation allow it to meet these high security standards, making it easier for businesses to comply with regulatory requirements.

**Specialized Capabilities**: ProvenCore can offer features and capabilities specifically tailored to security or real-time operations that Linux, being a general-purpose OS, might not provide. This can include secure boot, hardware-based isolation, and secure storage, among others.

**How Can Two OS Coexist?**

In ARM-based systems, this coexistence is facilitated by ARM TrustZone technology, which partitions the hardware into secure and non-secure worlds. Each world can run its own operating system:

* **Non-secure World (Linux OS)**:
  + Runs the general-purpose Linux OS, handling everyday computing tasks, applications, and the user interface. It operates with lower privilege levels, ensuring that any compromise does not affect the secure world.
* **Secure World (ProvenCore)**:
  + Runs ProvenCore, which handles security-sensitive operations like cryptographic processing, secure boot, and secure storage. It operates at a higher privilege level, inaccessible from the non-secure world, thereby providing an additional layer of security.

**Implementation**

* **Secure Monitor Call (SMC)**:
  + A mechanism used to switch between the secure and non-secure worlds. When a secure operation is needed (e.g., a cryptographic verification), the non-secure world makes an SMC to request services from the secure world (ProvenCore), which then performs the operation securely and returns the result.
* **Hardware Support**:
  + Modern processors, especially those based on ARM architectures, support virtualization and TrustZone technology, allowing them to efficiently and securely run multiple operating systems simultaneously.