



Booting Android with Linux EFI Boot Stub

Yi-Yo Chiang <yochiang@google.com>

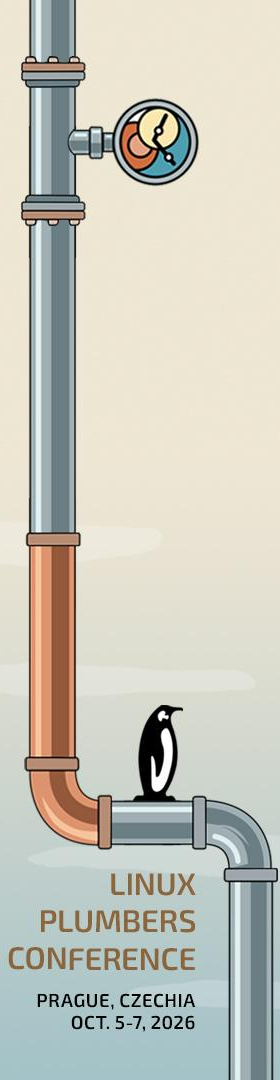


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026

GBL Recap

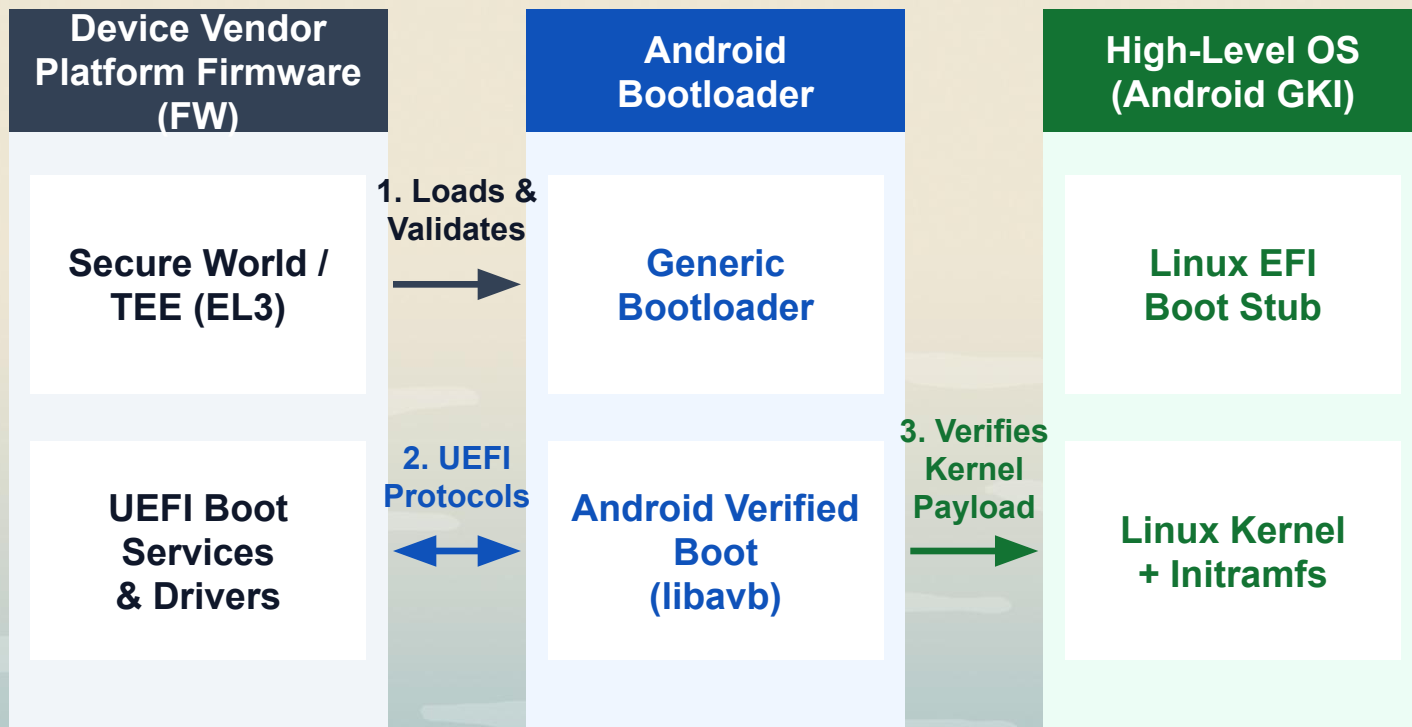
- **Generic Bootloader / Android Bootloader**
 - Generic and updatable UEFI application
 - Bare-metal Rust
 - UEFI interfaces FW ↔ Bootloader
- **Reduce vendor forks (LK, U-Boot, EDK2, etc) and integration efforts**
- **Provide common Android boot logic**
 - Android Verified Boot (AVB), DTB/DTBO assembly, DT & Bootconfig fixup etc



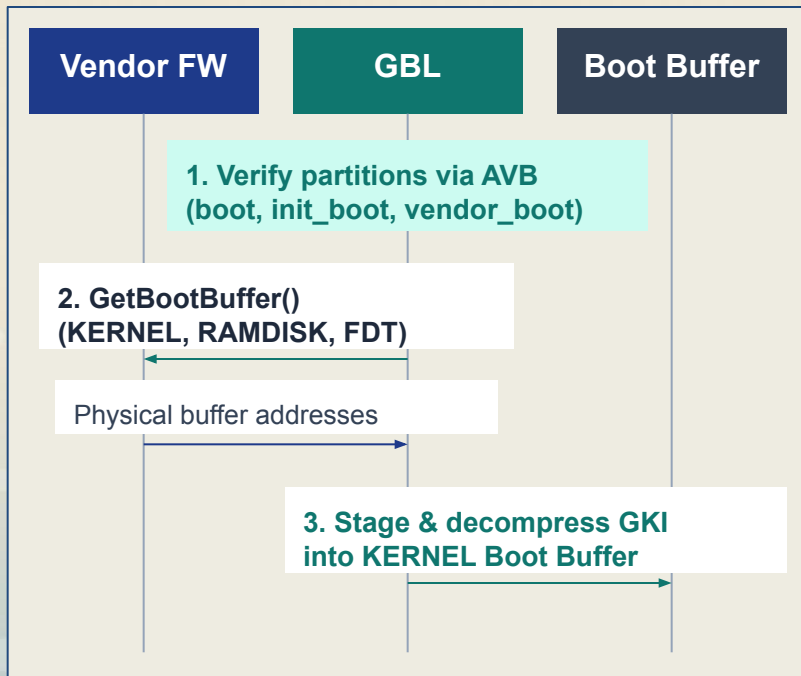
LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

GBL Architecture & Trust Boundary



Physical Memory Placement



- **Embedded SoC Constraints:**
 - Strict address requirements
 - Hypervisor carve outs
- **GBL_EFI_BOOT_MEMORY_PROTOCOL**
GetBootBuffer() provides buffers:
 - **KERNEL:** 2MB-aligned buffer
 - **RAMDISK, FDT, etc**



How GBL Boots Linux: Direct Boot vs. EFI Stub

Current GBL Direct Boot

1. Stage verified kernel, ramdisk & FDT in boot buffers
2. Populate FDT /chosen & apply DT fixups
3. Optional vendor FW handoff (could contain SMC calls), OR:
4. GBL calls **ExitBootServices()**, **disables MMU** & jumps to raw kernel

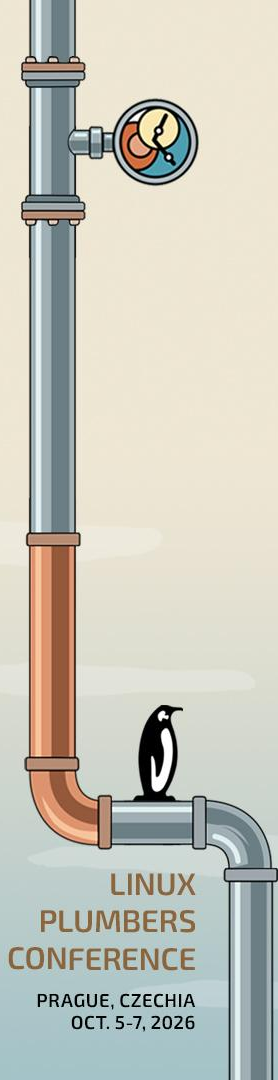
Booting via Linux EFI Boot Stub

1. Stage verified kernel, ramdisk & FDT in boot buffers
2. Install FDT into EFI configuration table
3. Call **BS→LoadImage()** & **BS→StartImage()** → enters **libstub**
4. **libstub** calls **ExitBootServices()**
5. **libstub** jumps to kernel (**efi_enter_kernel()**)



Challenges with Linux EFI Boot Stub

- **Privilege Handover (EL Elevation)**
 - Vendor FW runs untrusted GBL at **Non-Secure EL1**, while **KVM / Type-1 hypervisors** require EL2
 - **libstub** now owns **ExitBootServices()** & kernel jump → GBL never regains control for EL1-to-EL2 elevation
- **Image Loading (BS→LoadImage()) vs. In-memory Execution)**
 - **BS→LoadImage()** copies 30–50 MB out of the in-memory **2MB-aligned KERNEL buffer**
 - Expects **UEFI SecureBoot (PE Authenticode)**

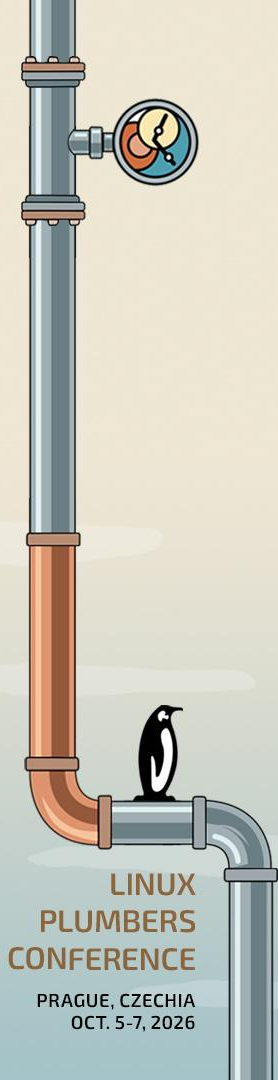


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Problem 1 — EL Elevation

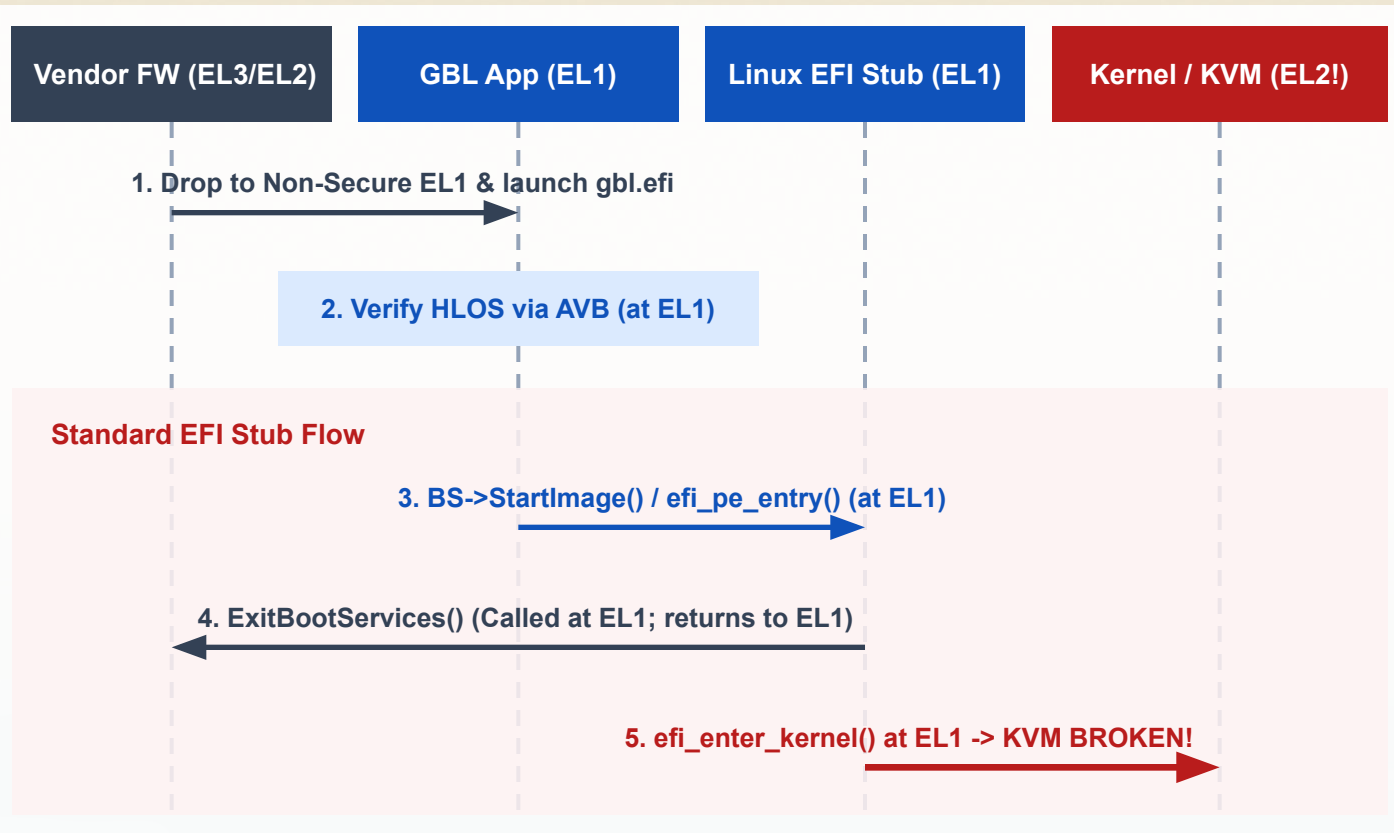
- Vendor FW drops to Non-Secure EL1 before launching untrusted GBL application
- KVM / Type-1 Hypervisors require initialization at EL2
- **Why Linux EFI Stub Breaks Legacy Elevation:**
 - **Legacy Flow:** GBL calls FW handoff hook -> SMC to enter kernel at EL2
 - **EFI Stub Flow:** libstub runs at EL1, calls ExitBootServices(), enters kernel at EL1



LINUX
PLUMBERS
CONFERENCE

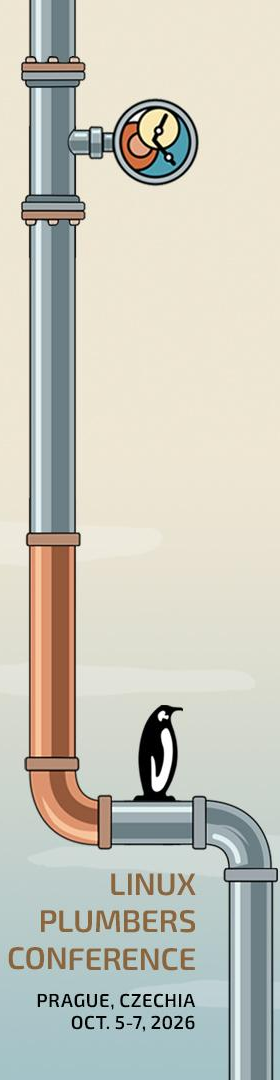
PRAGUE, CZECHIA
OCT. 5-7, 2026

Problem 1 — Standard EFI Stub Flow

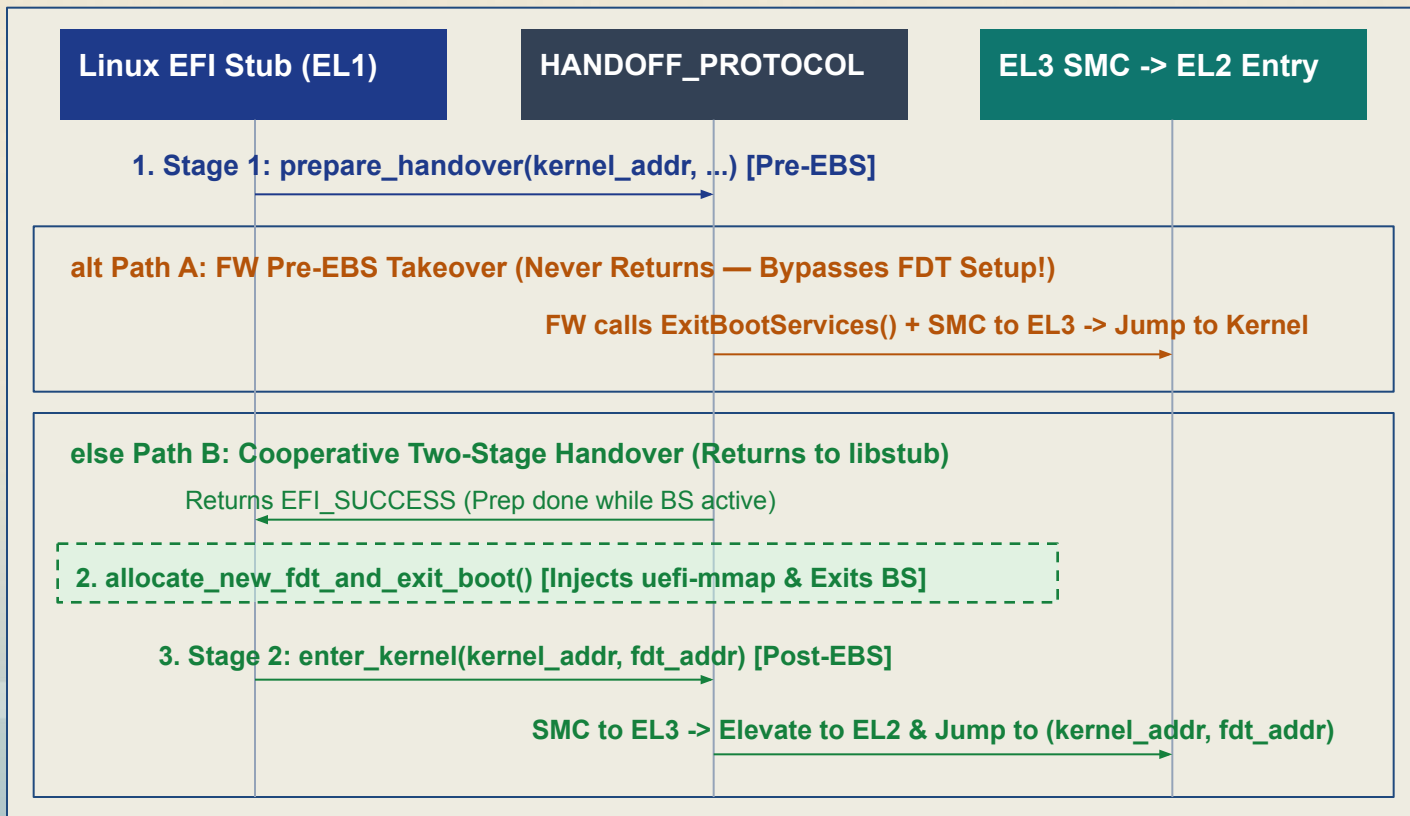


Problem 1 — Proposals

- **Two-stage Handoff Protocol (libstub ↔ FW cooperative flow):**
 - **Stage 1 (prepare_handover — Pre-EBS):** Called by libstub with (kernel_addr, fdt_addr) while Boot Services are active
 - **libstub calls allocate_new_fdt_and_exit_boot():** Injects linux,uefi-mmap-* and finalize FDT, calls ExitBootServices()
 - **Stage 2 (enter_kernel — Post-EBS):** Called by libstub with final (kernel_addr, fdt_addr) to enter kernel at EL2
- **Key Tradeoff — Non-Returning Stage 1 (Pre-EBS Takeover):**
 - Bypasses allocate_new_fdt_and_exit_boot()
 - Forces FW to duplicate linux,uefi-mmap-* setup



Problem 1 — Handoff Protocol



Problem 2 — BS->LoadImage() Flow

Standard BS->LoadImage(SourceBuffer) Flow

GBL KERNEL Buffer
(2MB-aligned)

1. Fails if SecureBoot=1
2. Allocates & copies 30–50 MB

UEFI Heap Copy
(30–50 MB duplicate;
Physical Placement LOST;
64KB-aligned)

Desired In-Place Flow (Zero Extra Copies)

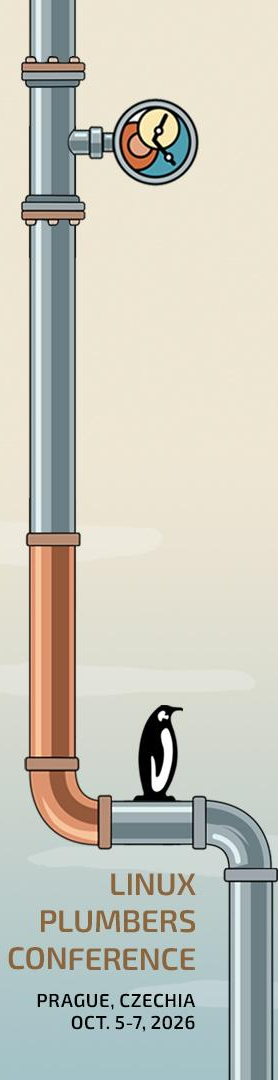
GBL KERNEL Buffer
(2MB-aligned)

(No extra copy or relocation)

Kernel Executes In-Place!
(Zero extra copies,
2MB placement kept, ignores
SecureBoot)

Problem 2 — Why BS->LoadImage() Fails

- **Starting State:**
 - 30–50 MB GKI staged in 2MB-aligned KERNEL buffer & verified via AVB (vbmeta)
- **Issues with BS->LoadImage(SourceBuffer):**
 - **Redundant 30–50 MB Copy:** Unconditionally allocates new pages & copies sections
 - **VOIDS Platform Placement:** Moves kernel out of negotiated KERNEL buffer
 - **EFI_SECURITY_VIOLATION (SecureBoot=1):** Enforces UEFI SecureBoot, but Android Kernel (GKI) uses AVB



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Problem 2 — Proposals

A: GBL Starts Kernel In-Place

- GBL jumps to in-memory `efi_pe_entry()` directly
- Never calls `BS->LoadImage()`
- GBL synthesises and installs a `EFI_LOADED_IMAGE_PROTOCOL` for the KERNEL boot buffer (to satisfy the EFI contract ↔ boot stub)
- Shims `BS->Exit()` & `ExitBootServices()`
- **Tradeoff:** Works on most firmwares without modifications. Non-standard.

B: Custom FW Protocol

- GBL-specific FW protocol that loads and starts Linux from the in-memory kernel buffer
- Also replaces `BS->LoadImage()` & `BS->StartImage()`
- Flag to disable SecureBoot
- **Tradeoff:** Requires all firmwares to implement this "In-memory Loading" protocol



LINUX

PLUMBERS CONFERENCE

PRAGUE, CZECHIA / OCT. 5-7, 2026



Thank You

Open discussion / Questions?

Contact: android-gbl@google.com



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026