Image Source: Gemini

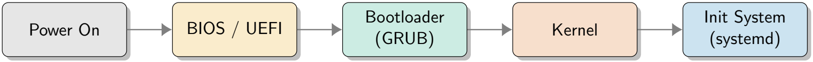
CSL 301

OPERATING SYSTEMS

Lecture 30
(Secure) Boot Process

Instructor
Dr. Dhiman Saha

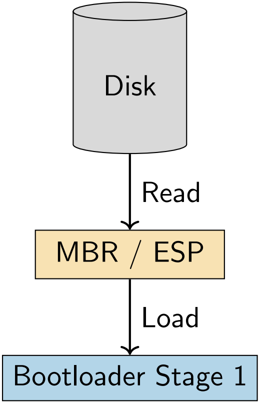
Overview: The Journey of Booting



- **Bootstrap**: Pulling oneself up by one's bootstraps.
- A chain reaction of handing over control to more complex software.

Stage 1: Power On & BIOS/UEFI

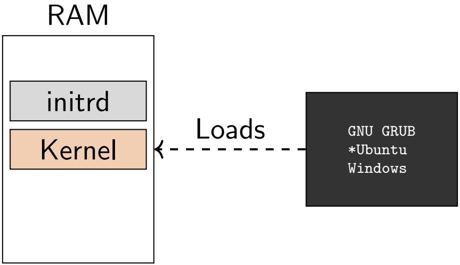
1. **Power On Self Test (POST)**
    - CPU starts, fetches instruction from ROM (Reset Vector).
    - Checks RAM, GPU, Keyboard.
2. **Finding the Boot Device**
    - BIOS: Looks for MBR (Master Boot Record) in first sector (512 bytes).
    - UEFI: Looks for EFI System Partition (ESP) containing ‘.efi‘ executables.



Stage 2: The Bootloader (GRUB)

Grand Unified Bootloader (GRUB)

- **Stage 1**: Tiny, lives in MBR. Loads Stage 1.5/2.
- **Stage 2**: Lives in ‘/boot‘ filesystem.
- Presents a menu to the user.
- Loads the **Kernel** (‘vmlinuz‘) and **Initial RAM Disk** (‘initrd‘/‘initramfs‘) into memory.



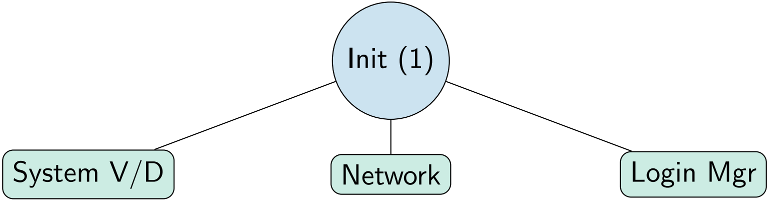
Stage 3: The Kernel

- The "Brain" of the OS takes over.
- **Hardware Detection**: Probes bus (PCIe, USB) and loads drivers.
- **Mounting Root**:
  1. Mounts temporary filesystem ('initramfs') to load necessary modules (drivers for disk, filesystem).
  2. Mounts the real Root Filesystem ('/') as read-only initially, then read-write.
- Executes the first user-space program: `/sbin/init` (PID 1).

Stage 4: Init System (systemd)

PID 1: The Mother of All Processes

- Responsible for bringing the system to a usable state.
- Starts background services (daemons): Network, Audio, Logging, Cron.
- Follows a dependency tree (Targets in systemd).



Stage 5: User Space

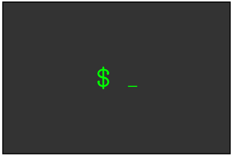
Getty & Login

- `getty`: Manages physical/virtual terminals (TTY).
- `login`: Authenticates user.

Display Manager (GUI)

- Starts X server or Wayland.
- Shows graphical login screen (GDM, SDDM).

Shell



```
$ _
```

System is Ready!

Summary

1. **BIOS/UEFI**: Hardware check, find bootloader.
2. **Bootloader**: Load Kernel + Initrd.
3. **Kernel**: Initialize hardware, mount root.
4. **Init**: Start services (PID 1).
5. **User Space**: Login → Shell/Desktop.

The Secure Boot Process

Secure Boot: The Core Concept

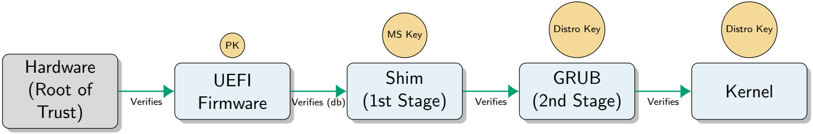
- **Goal**: Prevent malicious code (rootkits/bootkits) from loading during the boot process.
- **Mechanism**: Cryptographic Signatures.
- **Chain of Trust**:
  - Each component verifies the digital signature of the *next* component before executing it.
  - If verification fails, the boot process stops.
- **Root of Trust**: The foundation (usually the Hardware/Firmware) that is implicitly trusted.

Real World: UEFI Secure Boot Architecture

The Key Hierarchy (Stored in NVRAM)

- **Platform Key (PK)**: The "Master Key". Owned by the hardware vendor (OEM).
- **Key Exchange Key (KEK)**: Keys trusted to update the database. (Often includes Microsoft's key).
- **Allow Database (db)**: Public keys/hashes of authorized bootloaders (e.g., Microsoft Boot Manager, Linux Shim).
- **Forbidden Database (dbx)**: "Blacklist" of revoked keys/hashes (e.g., for binaries with known vulnerabilities).

The Chain of Trust in Action



- **Shim**: A small bootloader signed by Microsoft (trusted by UEFI). It contains the distro's public key to verify GRUB.

Handling Third-Party Code: Machine Owner Key (MOK)

- **Problem**: What if you want to install a custom kernel module (e.g., Nvidia driver, VirtualBox) or a self-compiled kernel?
- **Solution**: The **Shim** bootloader allows the user to enroll their own keys.
- **MOK Manager**:
  - A special UEFI program launched by Shim if verification fails but a key is pending.
  - Requires physical presence (user must press a key) to prove authorization.
  - Adds the user's key to the "MOK List" (stored in NVRAM), which Shim trusts.

Secure Boot vs. Measured Boot

Secure Boot (Enforcement)

- **Action**: Stop!
- **Mechanism**: Verify signature against a trusted key.
- **Outcome**: Prevents execution of untrusted code.

Measured Boot (Attestation)

- **Action**: Record and Continue.
- **Mechanism**: Hash the component and extend a PCR (Platform Configuration Register) in the **TPM** (Trusted Platform Module).
- **Outcome**: Proves to a remote server (Remote Attestation) or local secret (BitLocker) that the boot state is valid.

Limitations and Controversy

- **"Restricted Boot"**:
  - Concern: If users cannot disable Secure Boot or add their own keys, the device is locked to the vendor's OS.
  - *Reality*: On x86 (PC), Microsoft requires disabling to be possible. On ARM (Mobile/Tablets), it is often mandatory (locked bootloaders).
- **Complexity**:
  - Managing keys (signing modules) can be difficult for average users.
  - "BootHole" vulnerability: A bug in GRUB allowed bypassing Secure Boot, requiring massive revocation (dbx updates).

Summary: Secure Boot

1. **Chain of Trust**: Every boot component verifies the next one using digital signatures.
2. **UEFI Keys**:
   - **PK**: Platform Key (Root).
   - **KEK**: Key Exchange Key (Updates).
   - **db**: Allowed signatures (Whitelist).
   - **dbx**: Revoked signatures (Blacklist).
3. **Shim & MOK**: Bridges the gap between UEFI and Distros; allows users to enroll custom keys.
4. **Secure vs. Measured**: Secure Boot *stops* bad code; Measured Boot *records* what ran (for TPM/Attestation).