# CWE Detail – CWE-1326

## Description

A missing immutable root of trust in the hardware results in the ability to bypass secure boot or execute untrusted or adversarial boot code.

## Extended Description

A System-on-Chip (SoC) implements secure boot by verifying or authenticating signed boot code. The signing of the code is achieved by an entity that the SoC trusts. Before executing the boot code, the SoC verifies that the code or the public key with which the code has been signed has not been tampered with. The other data upon which the SoC depends are system-hardware settings in fuses such as whether "Secure Boot is enabled". These data play a crucial role in establishing a Root of Trust (RoT) to execute secure-boot flows. One of the many ways RoT is achieved is by storing the code and data in memory or fuses. This memory should be immutable, i.e., once the RoT is programmed/provisioned in memory, that memory should be locked and prevented from further programming or writes. If the memory contents (i.e., RoT) are mutable, then an adversary can modify the RoT to execute their choice of code, resulting in a compromised secure boot. Note that, for components like ROM, secure patching/update features should be supported to allow authenticated and authorized updates in the field.

## Threat-Mapped Scoring

Score: 0.0

Priority: Unclassified

## Related Attack Patterns (CAPEC)

* CAPEC-679
* CAPEC-68

## Attack TTPs

**•** T1553.002: Code Signing (Tactics: defense-evasion)

## Modes of Introduction

**•** Architecture and Design: N/A

**•** Implementation: Such issues could be introduced during policy definition, hardware architecture, design, manufacturing, and/or provisioning. They can be identified later during testing or system configuration phases.

## Common Consequences

**•** Impact: Gain Privileges or Assume Identity, Execute Unauthorized Code or Commands, Modify Memory — Notes:

## Potential Mitigations

**•** Architecture and Design: When architecting the system, the RoT should be designated for storage in a memory that does not allow further programming/writes. (Effectiveness: N/A)

**•** Implementation: During implementation and test, the RoT memory location should be demonstrated to not allow further programming/writes. (Effectiveness: N/A)

## Applicable Platforms

**•** None (Class: Not Language-Specific, Prevalence: Undetermined)

## Demonstrative Examples

**•** In general, if the boot code, key materials and data that enable "Secure Boot" are all mutable, the implementation is vulnerable.

**•** It performs security-critical functions such as defining the system's device tree, validating the hardware cryptographic accelerators in the system, etc. Hence, write access to bootrom should be strictly limited to authorized users or removed completely so that bootrom is immutable. In this example (see the vulnerable code source), the boot instructions are stored in bootrom memory, mem. This memory can be read using the read address, addr\_i, but write access should be restricted or removed.