STAR LAB
A WIND RIVER COMPANY
SOFTWARE

WHITEPAPER

Titanium Secure Boot vs. UEFI Secure Boot

Titanium Secure Boot (a form of Measured Boot) and UEFI Secure Boot are similar in that they verify the authenticity of boot-time components; however, they vary greatly in terms of how verification is performed and to what level of granularity.

Both UEFI Secure Boot and Titanium Secure Boot start from the foundation of the Intel hardware and BIOS working together to measure the integrity of the early boot components / firmware.

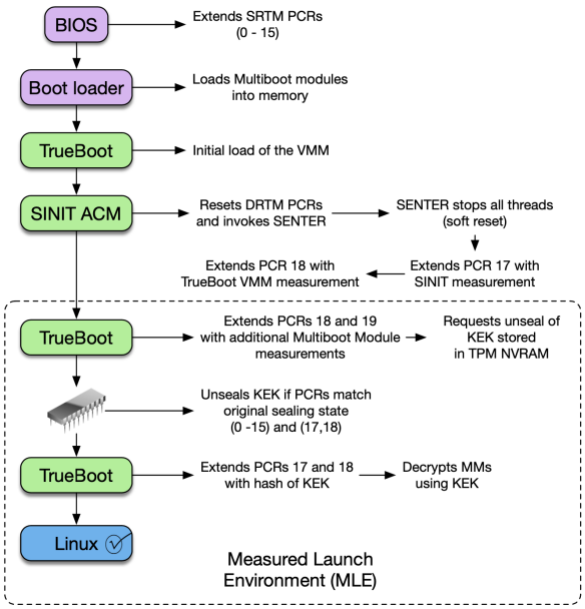
UEFI Secure Boot

UEFI Secure Boot proceeds through the boot process by verifying that the bootloader, Linux kernel (if chain loaded in EFI mode), and (optionally) kernel modules are individually signed and authenticated with enrolled keys. Each component is independently verified in an unbroken process that takes the Intel CPU through execution modes to get to a booted system. The system BIOS, bootloader, and Linux kernel must all be UEFI compatible and signed using private keys and verified with the aforementioned enrolled public keys. The system administrator can enroll new keys as necessary; however, that also means an attacker with sufficient access can load their own keys which would allow booting components signed with an attacker's key.

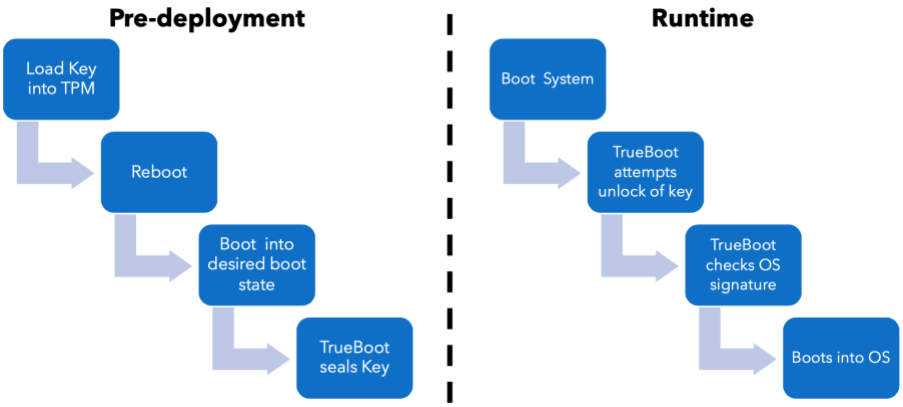
Titanium True Boot

Titanium Secure Boot starts much the same way that UEFI Secure Boot does – verifying BIOS firmware; however, the target system is instructed to store measurements of these firmware level components in an off-CPU TPM. Once the bootloader launches the Titanium Secure Boot module, a signed Intel code module is measured by Titanium Secure Boot and then loaded into the boot process to measure Titanium Secure Boot and clear the CPU state. Titanium Secure Boot then regains control and authenticates the rest of the boot components and the boot-time command-line arguments. This an important distinction.

UEFI Secure Boot does not measure the initramfs or command line parameters. This means that, without these measurements, an attacker would be able to subvert or interpose late-load security components by modifying early boot components within the initramfs or disable important security features (such as intel_iommu=on). With Titanium Secure Boot, measurements (stored in the TPM Platform Configuration Registers or PCRs) are combined to unlock non-extractable key material in the TPM. The unlock attempt succeeds only if the sequence of measurements exactly match a prior trusted state's measurements. *See the diagram at the right.*



Since Titanium Secure Boot is independent from the kernel or other boot components, Titanium Secure Boot works with most any distribution of Linux (e.g., RHEL, RedHawk, CentOS) on a host using legacy BIOS or UEFI boot. Using the Titanium Secure Boot tooling, the Linux kernel and initramfs are protected and authenticated by customer-controlled key material sealed within the TPM, meaning that they can be updated without reprovisioning the TPM. Also, note that Titanium Secure Boot and the signed Intel code module only execute in a special state that resets CPU state, protects memory regions from probing, and invokes the IOMMU to protect memory regions even after booting the kernel. Here's a look at the Titanium Secure Boot Workflow:



In summary, Titanium Secure Boot provides a stronger level of boot-time authentication/trust than UEFI Secure Boot, all the while being more flexible in terms of target Linux distributions and BIOS variants. Furthermore, TrueBoot leverages a TPM to supplement its attestation, removing the sole verification burden from boot-time software that must trust itself. For more information, refer to the Intel TXT Development Guide and the Intel Trusted Execution Technology for Server Platforms.