Securing TPM Secrets with TXT and Kernel Signatures

Paul Moore
paul@paul-moore.com
November 2019

Disclaimers

This talk is not about secure boot, although I will talk about the related technologies a lot.

This talk is about protecting secrets stored in the TPM NVRAM, not restricting what can be booted on the system.

This talk focuses on Intel's TXT, AMD's SVM should provide similar capabilities.

The Problem

Problem Statement

- Store secrets in the TPM
- Restrict access to the secrets to authorized kernels
- Work on legacy BIOS as well as UEFI based systems
- Easy to manage, handle updates gracefully

Protecting TPM Secrets

- "Seal" data to a set of PCRs
  - A specific set of PCR values are used as a key to lock/unlock TPM secrets
- TPM protects PCRs from tampering

Setting TPM PCRs During Early Boot

- Secure boot measures system state into the PCRs
  - firmware / config
  - bootloader, etc.
  - kernel
- When the components change, the PCR values change

Detour: Secure Boot

UEFI Secure Boot

- UEFI verifies signature of everything it executes
  - static root of trust
  - public key embedded in firmware
  - Microsoft controls master keys
- PCR 7 measures the kernel's signing authority
  - stable across multiple kernels with the same signer

Intel Trusted Execution Technology (TXT)

- Hardware and firmware creates a dynamic root of trust
  - "SINIT ACM"
- TXT "measured launch environment" verifies and bootstraps the kernel
  - tboot bootloader
- PCRs measuring the OS are based on hashes
  - not stable for TPM sealing

Solving The Problem

Applying Secure Boot to the Problem

- UEFI Secure Boot
  - stable PCR
  - only works on modern UEFI systems
- Intel TXT
  - unstable PCRs
  - works on legacy BIOS and all UEFI systems

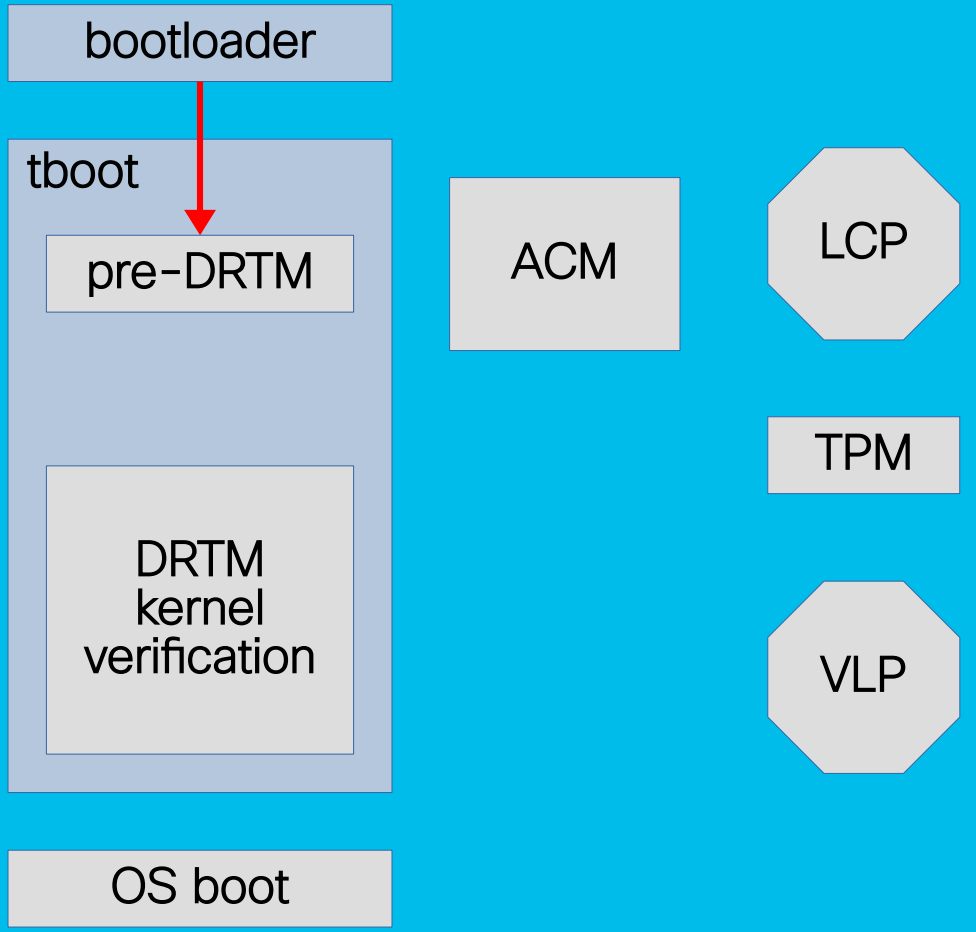
Lessons Learned

- Signature based PCRs are stable assuming the same signing authority
- TXT works regardless of the firmware type (legacy/UEFI)
- TXT verification of the kernel/initrd/cmdline happens in the tboot bootloader

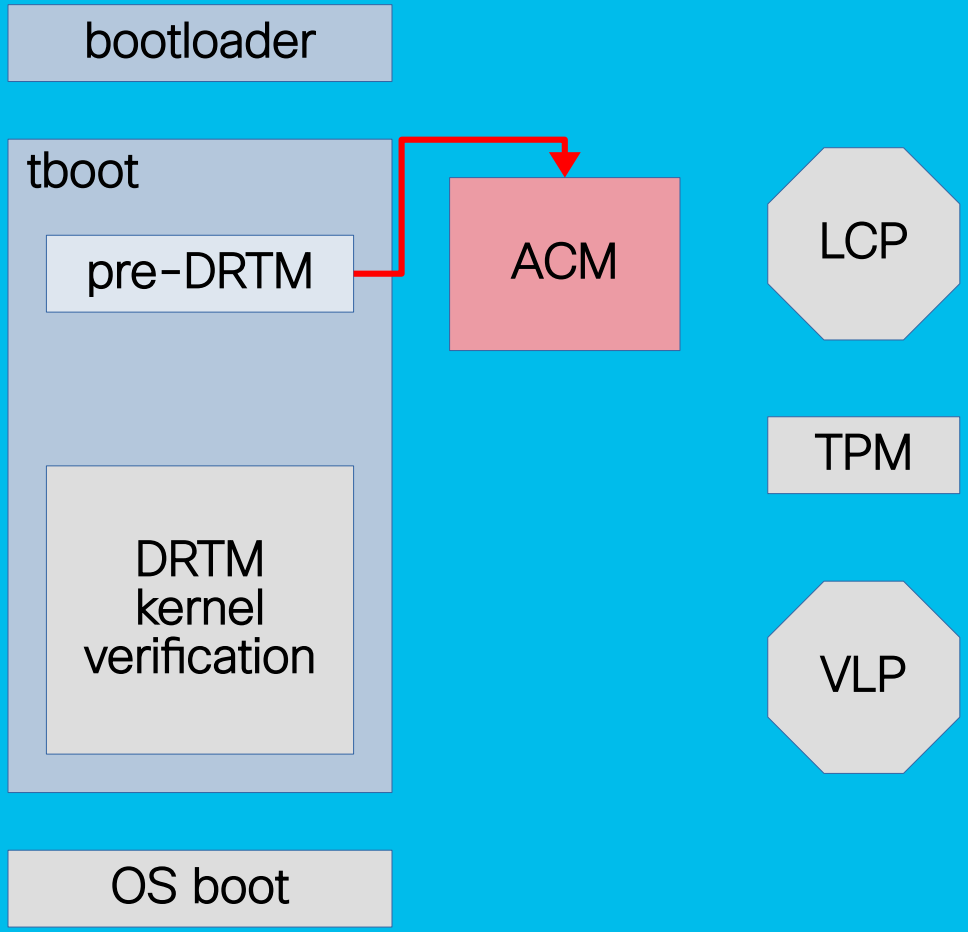
Proposed Solution (TXT/sig)

- Extend tboot to support signature verification using the PECOFF format
  - same format as UEFI Secure Boot
  - add the signing authority to the tboot policy
- No changes to SINIT ACM required

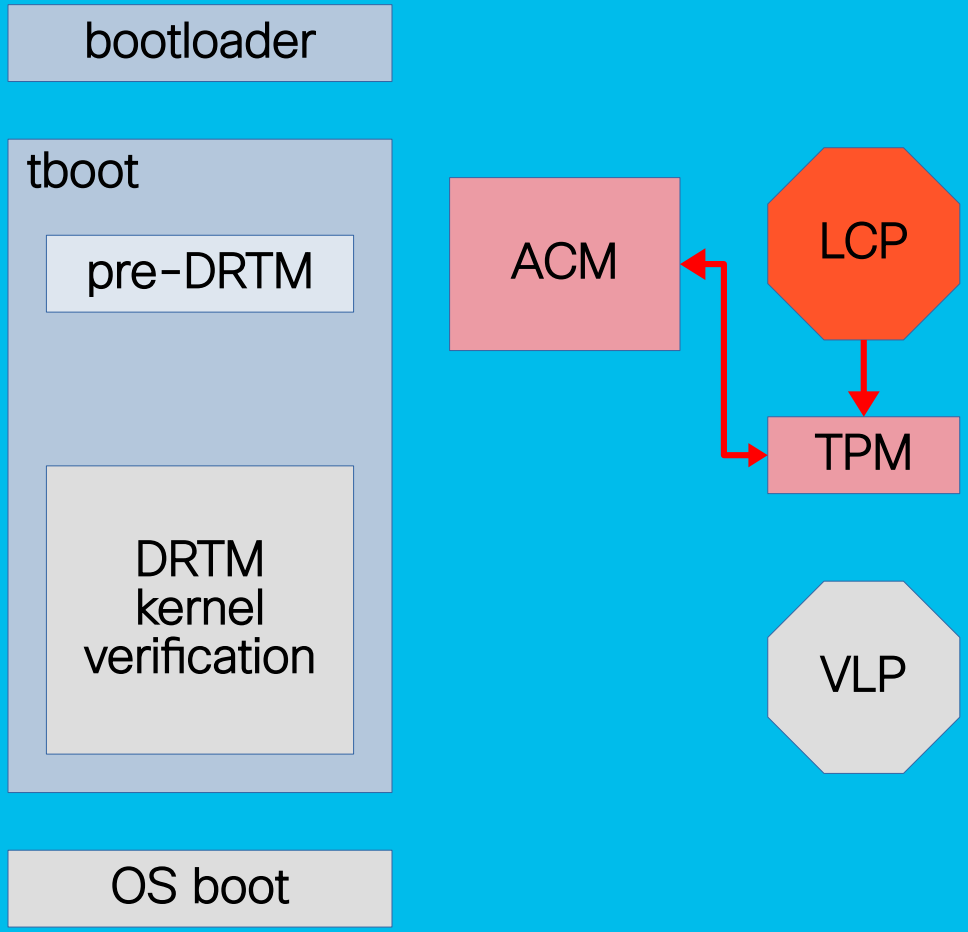
TXT/sig Boot Process



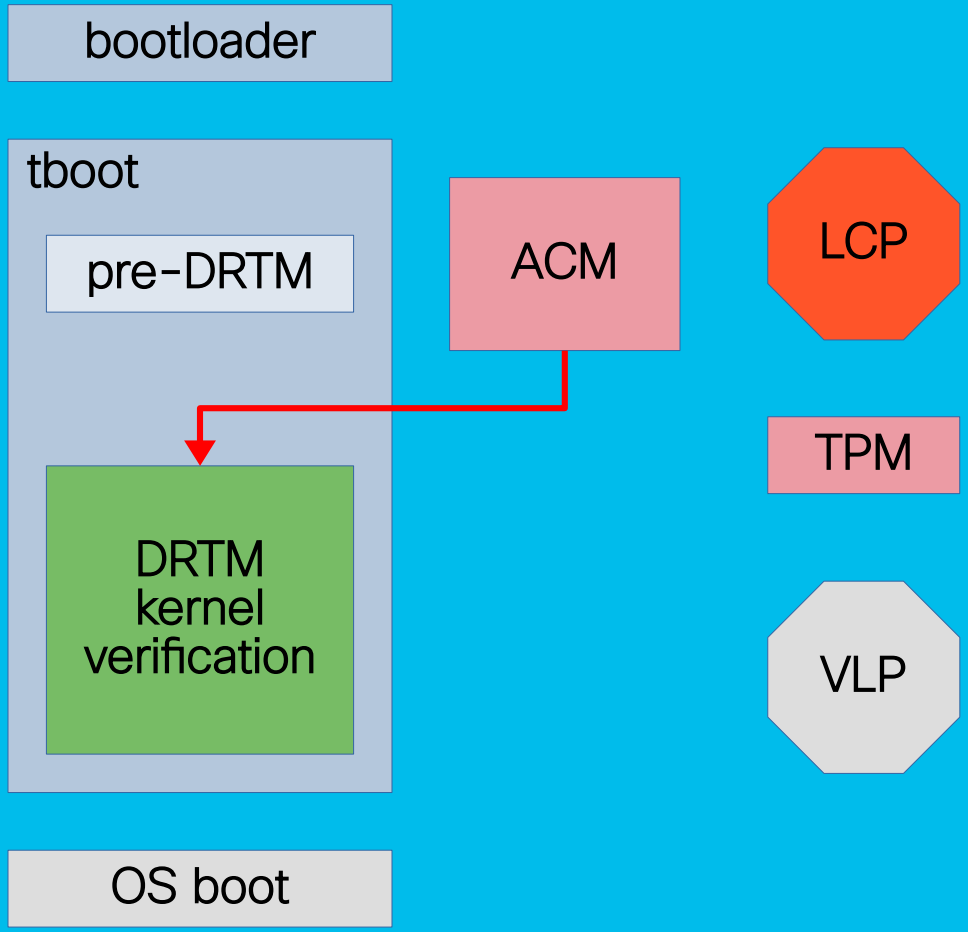
- bootloader boots tboot
- tboot performs TXT sanity checks



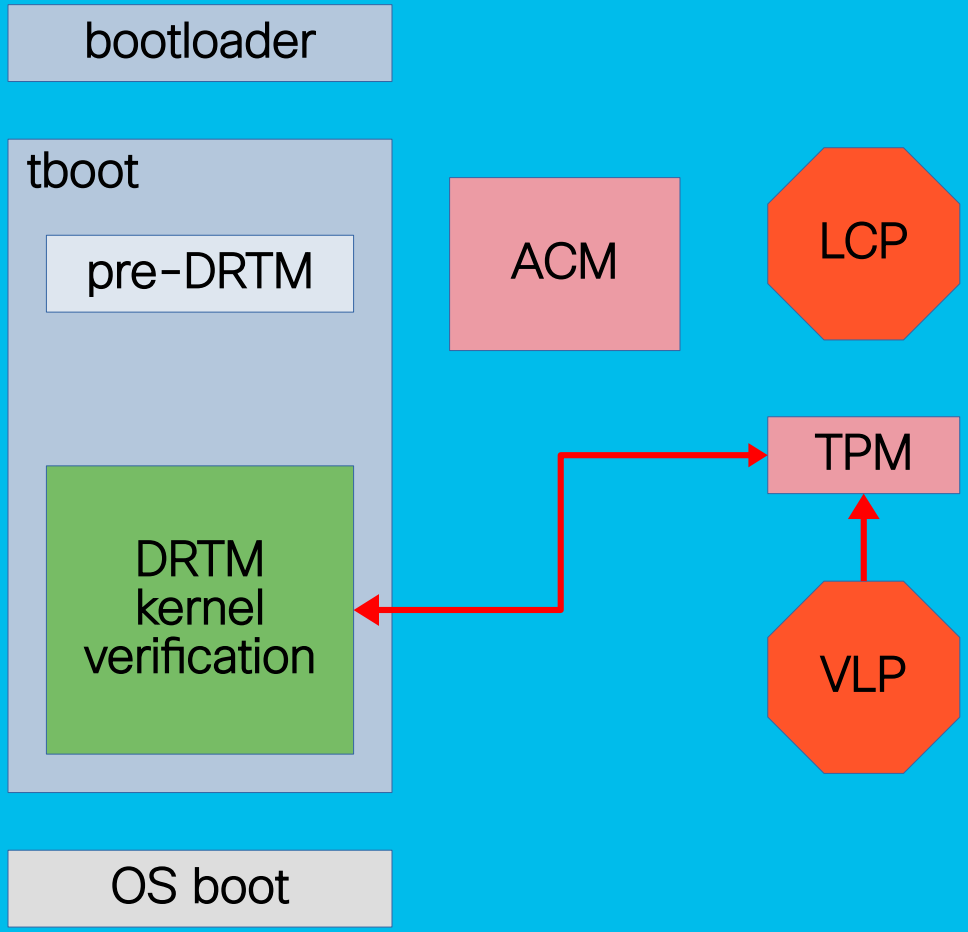
- tboot issues special CPU instruction to start TXT process
- SINIT Authenticated Code Module (ACM) establishes a dynamic root of trust



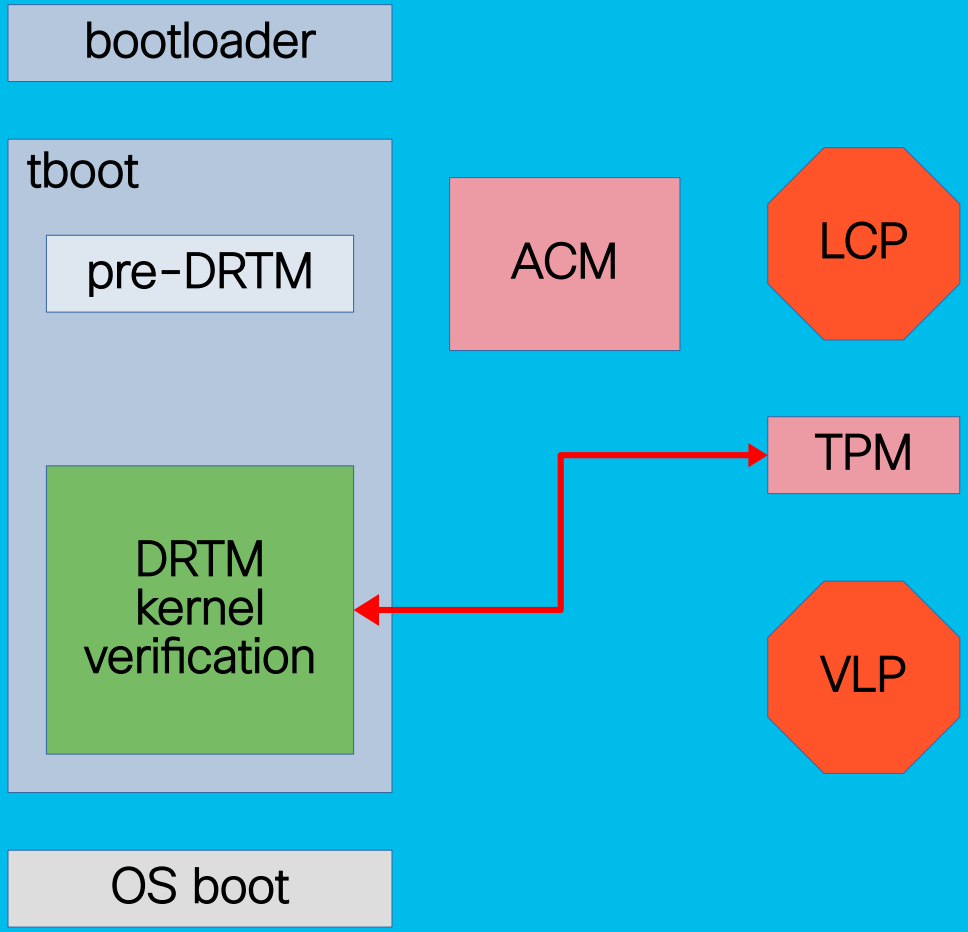
- ACM examines the Launch Control Policy (LCP) rooted in the TPM
- ACM enforces the LCP
  - validates firmware
  - validates tboot



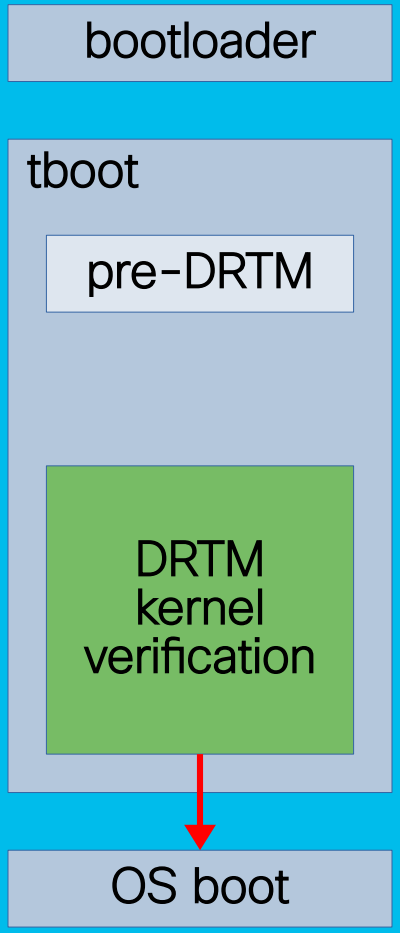
- ACM returns execution to the “measured launch environment” (tboot)
- tboot continues to execute in a protected environment

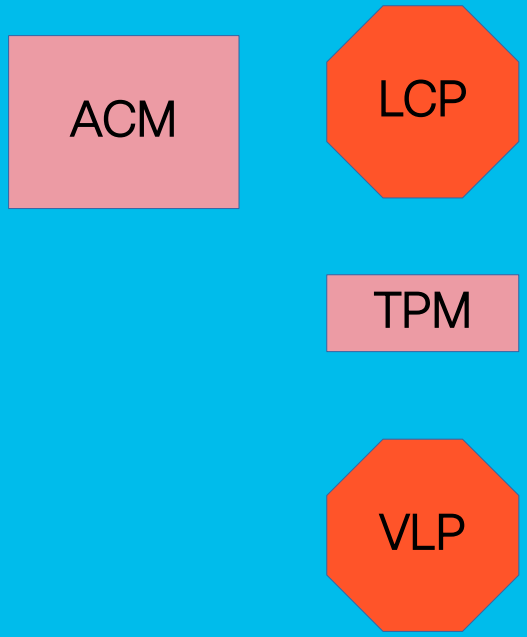


- tboot examines the Verified Launch Policy (VLP) rooted in the TPM
- tboot verifies the kernel, initrd, and cmdline
  - currently using hash values
  - adding support for signature verification



- tboot extends TPM PCRs
  - kernel signing authority certificate digest
  - kernel, initrd, and cmdline digests
- TPM TXT PCRs are protected against tampering outside the dynamic root of trust





- tboot boots the OS using the measured kernel, initrd, cmdline
- sealed TPM secrets are unlocked if the tboot PCR values match the sealing values

Current Status

- Prototype is in progress
  - kernel verification working, certificate chains supported
  - policy and PCR support are works in progress
- Prototype code

  https://github.com/pcmoore/misc-tboot

  (checkout the working-txtsig branch)

Open Issues

- No signature verification of the initrd or command line
  - problem for UEFI too
  - may be able to use UEFI workarounds
  - existing digest verification OK
- Upstream acceptance
  - prototype has been well received

Questions?

- Mail: paul@paul-moore.com
- Twitter: @securepaul