presented by

ORACLE

TrenchBoot and GRUB – A Quick Introduction

UEFI 2020 Virtual Plugfest

June 16, 2020

Daniel Kiper, Oracle & Daniel P. Smith, Apertus Solutions

Meet the Presenter

Daniel Kiper
Software Engineer
Member Company: Oracle

Agenda

- TrenchBoot – What is it?
- TrenchBoot and UEFI Secure Boot
- TrenchBoot and GRUB – Why?
- GRUB - Current State and Challenges
- Questions?
- Documentation

TrenchBoot

- TrenchBoot is a cross-community integration project focused on launch integrity
  - This means there is no “one thing” that is TrenchBoot
  - The name was a play off of dealing with the muddy mess of trying to find a way to unify boot integrity
  - The purpose is to develop a common, unified approach to building trust in the platform through launch integrity
  - And to work with existing Open Source ecosystem to integrate the approach into their respective projects
    - The intention here is to have a unified Dynamic Launch approach between Xen, KVM, Linux, BSD(s), and potentially proprietary kernels

Motivation

- The idea for TrenchBoot originated in 2014 dealing with the limitations of using tboot to launch Xen for the OpenXT project
  - Access to the TXT TPM event log is blocked
  - Conflict over access to the UEFI boot services
  - Can only measure Multiboot modules that were loaded into memory by the bootloader
  - Supports only one attestation action: predetermined the PCR manifest verification
  - Only supports the Intel TXT, no love for AMD's Secure Startup and other architectures and platforms

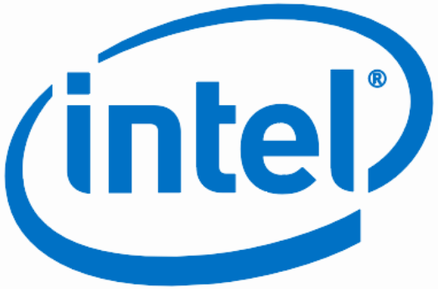
Motivation – Continuation

- Launch integrity is the foundation for platform security
  - It deserves the attention needed to get it right and well integrated with Open Source
- In the past Dynamic Launch was under utilized
  - It can in fact be initiated many times between power-on and power-off
  - Each Dynamic Launch is an opportunity to establish the current integrity of the platform

Secure Launch for Linux

- TrenchBoot Secure Launch for Linux provides for different strategies to build trust in the platform
  - First Launch – Establishing hardware rooted integrity during platform boot
  - Runtime Launch – Establishing hardware rooted integrity during platform runtime, e.g.
    - Secure Launch a kernel upgrade
    - Secure Launch Integrity Kernel for runtime verification
      - Integrity verification before executing a privileged operation
      - Re-establishing platform state after sleep or hibernate
    - Secure Launch Update/Shutdown kernel
      - Reviewing platform state before platform reboot/shutdown
      - Checking integrity before persisting state to disk

Who Contributes to TrenchBoot?

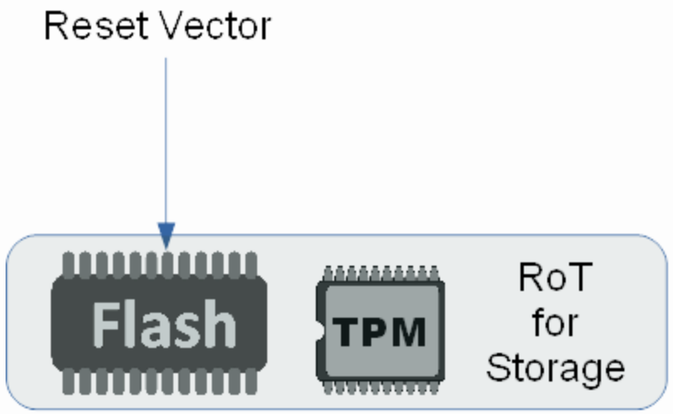


Terminology

Mapping concepts to TCG specification and vendor terms

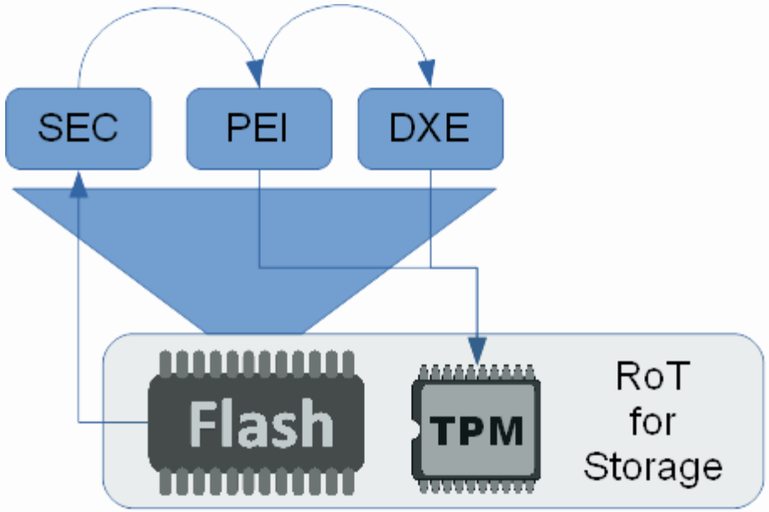
Description	TCG	Intel TXT	AMD-V
Process of starting a software environment at an arbitrary time in the runtime of a system	Dynamic Launch (DL)	Late Launch	Secure Startup
Platform dependent event that triggers the DL	DL Event	GETSEC[SENTER]	SKINIT
Performs initial configuration actions that are platform specific before invoking DL Event	D-RTM Configuration Environment (DCE) Preamble		
The Core Root of Trust for the DL environment that is initiated by a DL event and represented by the initial measurement	D-CRTM		
Software/firmware that executes from the instantiation of the DL Event to the transfer of control to the DLME	D-RTM Configuration Environment (DCE)	Authenticated Code Module (ACM)	Secure Loader (SL)
Software executed after the DCE instantiated TCB is established	Dynamically Launched Measured Environment (DLME)	Measured Launch Environment (MLE)	Security Kernel (SK)

UEFI Secure Boot Trust



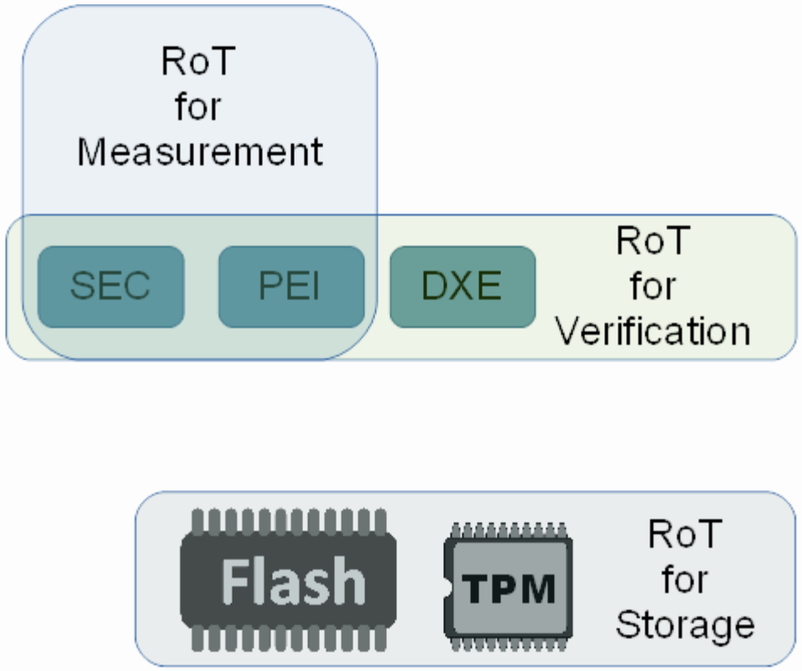
- The CRTM is not measured until during PEI
  - Therefore SEC and PEI must be trusted
  - CRTM is of SEC and PEI, thus it is self referential
  - Relies on integrity of Boot Flash
  - Relies on TPM to protect measurements
- The DXE phase enforces UEFI secure boot verification

UEFI Secure Boot Trust



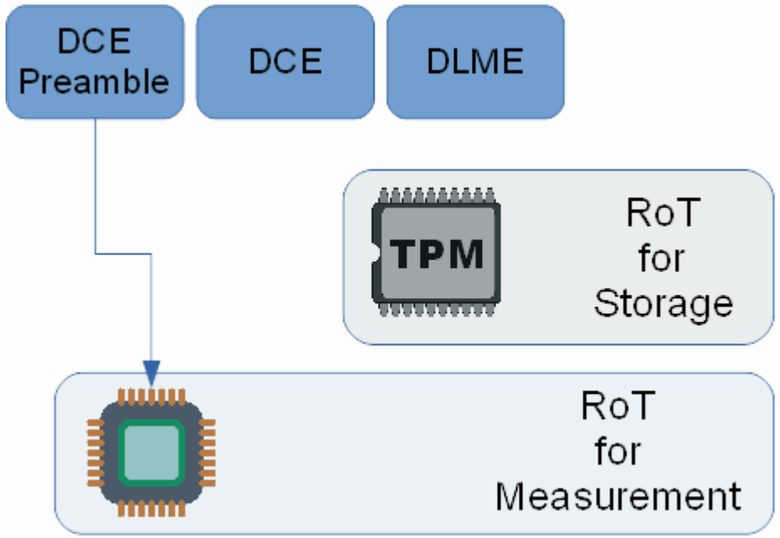
- The CRTM is not measured until during PEI
  - Therefore SEC and PEI must be trusted
  - CRTM is of SEC and PEI, thus it is self referential
  - Relies on integrity of Boot Flash
  - Relies on TPM to protect measurements
- The DXE phase enforces UEFI secure boot verification

UEFI Secure Boot Trust



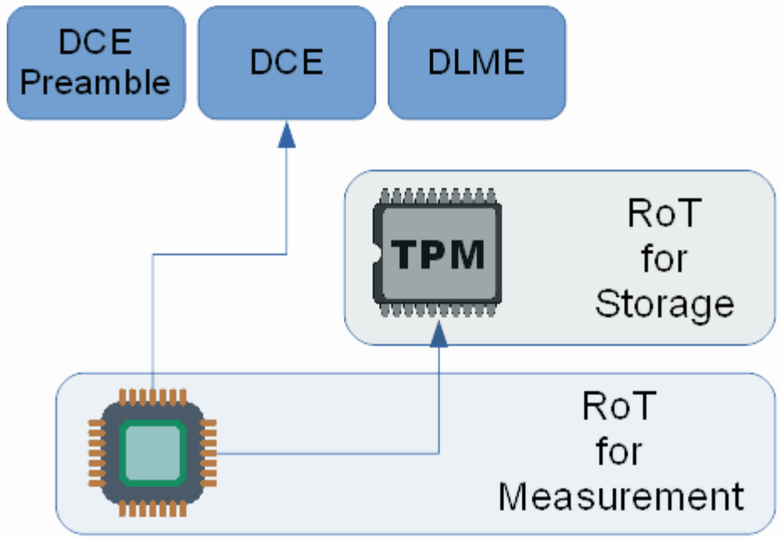
- The CRTM is not measured until during PEI
  - Therefore SEC and PEI must be trusted
  - CRTM is of SEC and PEI, thus it is self referential
  - Relies on integrity of Boot Flash
  - Relies on TPM to protect measurements
- The DXE phase enforces UEFI secure boot verification

Dynamic Launch Trust



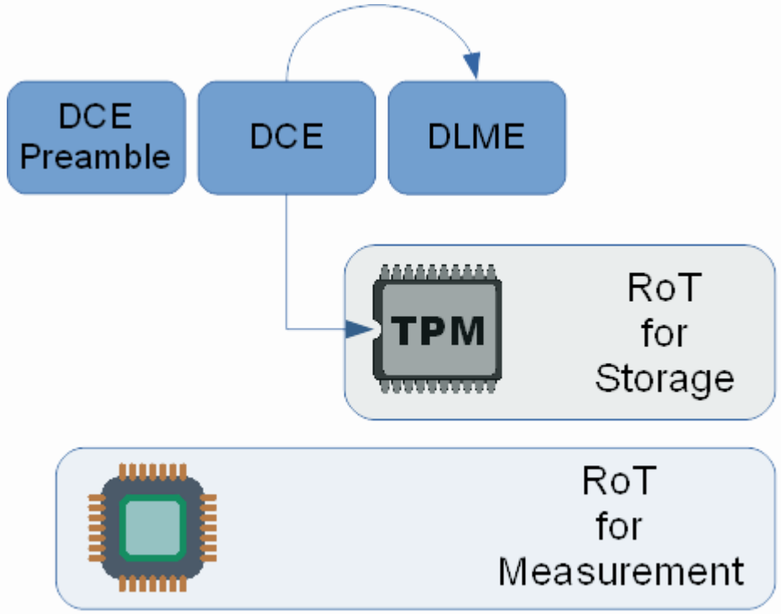
- DCE Preamble may be a bootloader or an executing OS
- The CRTM is taken by the CPU
  - Relies on the TPM to protect measurements
- On Intel there is also an additional authentication protocol between the DCE and CPU

Dynamic Launch Trust



- DCE Preamble may be a bootloader or an executing OS
- The CRTM is taken by the CPU
  - Relies on the TPM to protect measurements
- On Intel there is also an additional authentication protocol between the DCE and CPU

Dynamic Launch Trust

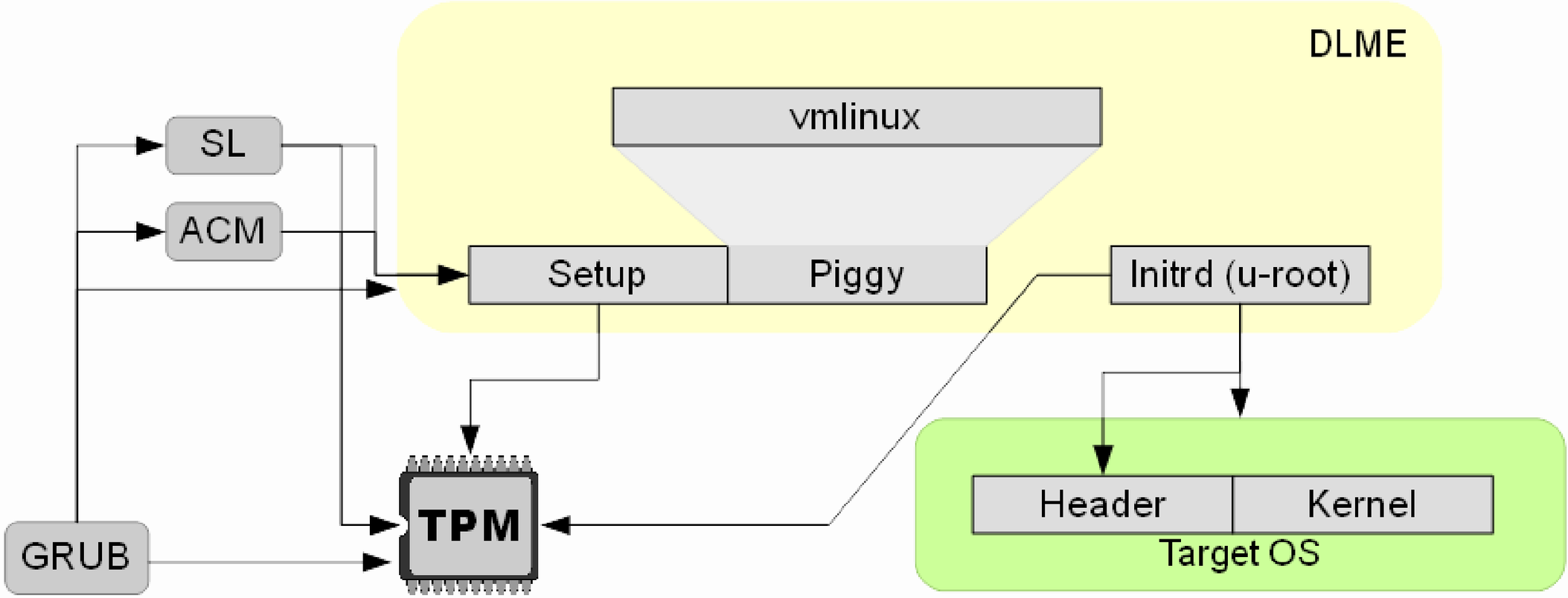


- DCE Preamble may be a bootloader or an executing OS
- The CRTM is taken by the CPU
  - Relies on the TPM to protect measurements
- On Intel there is also an additional authentication protocol between the DCE and CPU

The Control of a Dynamic Launch

- Provides a very controlled and protected startup
  - The CPU obtains Locality 4 on the TPM and clears DRTM PCRs (17-22)
  - All CPU interrupts (NMI, SMI, INIT, etc) are disabled
  - The CPU protects the DCE from DMA access
    - Intel uses Cache as RAM (CRAM)
    - AMD uses Device Exclusion Vector (DEV)
  - The DCE is measured by the CPU and stored in PCR 17 of the TPM before execution
    - On Intel the ACM is authenticated before measurement
    - On AMD the Secure Loader is owner provided
  - The DCE ensures the DLME is DMA protected, measures, and then executes
- The result is a very high integrity assertion of the DLME
  - Removes boot firmware from the TCB with the exception being the SMI Handler

Basic Flow of First Launch – Intel TXT



The GRUB History

- The project was initiated by Erich Boleyn in 1995
- It was an attempt to boot the GNU Hurd with the University of Utah's Mach 4 microkernel
- One of the outcomes of this efforts was the Multiboot Specification made by Erich Boleyn and Brian Ford
- Erich tried to implement the Multiboot Specification in FreeBSD boot loader but quickly realized that it was easier to write own bootloader from scratch
- This way the GRUB was born
- In 1999, Gordon Matzigkeit and Yoshinori K. Okuji adopted GRUB as an official GNU package

The GRUB History – Continuation

- Over the next few years, GRUB was extended to meet many needs
- However, it quickly became clear that its design was not keeping up with the extensions being made to it
- Around 2002, Yoshinori K. Okuji started work on PUPA (Preliminary Universal Programming Architecture for GNU GRUB), aiming to rewrite the core of GRUB
- The project PUPA was later renamed to GRUB2 and the original version of GRUB was renamed to GRUB Legacy
- The GRUB Legacy last release (0.97) was made in 2005
- Major GNU/Linux distributions migrated to GRUB2 between 2007 and 2009

Based on https://www.gnu.org/software/grub/manual/grub/grub.html#History

The GRUB – Why is it the Bootloader of Choice?

- The GRUB is the most common boot loader in deployment thus making it the choice initial boot loader to make capable of being a DCE Preamble for DL
- The GRUB is the most feature rich reach bootloader in the wild:
  - It supports at least 24 variants of targets (architectures) including ARM, x86, IA64, MIPS, POWER, RISC-V, SPARC64 and platforms like e.g. UEFI
  - It supports many filesystems including btrfs, ext2, ext3, ext4, F2FS, FAT, HFS, JFS, ReiserFS, SquashFS, romfs, NTFS, XFS, ZFS, LUKS, LUKS2
  - It has many security and crypto features embedded including UEFI Secure Boot via shim_lock and TPM support
  - It can start directly from ROM (coreboot)
  - It supports network boot
  - It has minimal shell capabilities which allow scripting
  - And many more...

The GRUB and UEFI

- The GRUB works on all architectures which are capable of running UEFI
- Most UEFI features are supported by the GRUB
- The GRUB supports the UEFI Secure Boot via shim
- The GRUB supports measurements using the UEFI TPM calls
- The GRUB can load many different OSes which even sometimes do not support the UEFI at all
- The GRUB presents the unified interface to the user regardless of architecture and firmware
- The GRUB supports basic scripting which is very useful for automation
- The GRUB community is pretty active

The GRUB – Current Challenges

- We want to unify the UEFI Linux boot protocol for all targets
- ...and later for other OSes
- The GRUB has some long standing network boot problems on UEFI platforms due to issues with the SNP driver
- The project struggles with a shortage of reviewers from firmware and OSes side

Questions?

Documentation

- https://trustedcomputinggroup.org/wp-content/uploads/TCG_D-RTM_Architecture_v1-0_Published_06172013.pdf
- https://trustedcomputinggroup.org/wp-content/uploads/DRTM-Specification-Overview_June2013.pdf
- https://github.com/TrenchBoot/
- https://www.gnu.org/software/grub/
- https://lists.gnu.org/mailman/listinfo/grub-devel/

Thanks for attending the UEFI 2020 Virtual Plugfest

For more information on UEFI Forum and UEFI Specifications, visit http://www.uefi.org

presented by

ORACLE