TrenchBoot

How to nicely boot system with Intel TXT and AMD SVM

Daniel Kiper, *Oracle, Software Developer, GRUB upstream maintainer*
Daniel P. Smith, *Apertus Solutions, Chief Technologist*

Linux Security Summit North America 2019, August 19-21, 2019

ORACLE

Program Agenda

1. Introduction to Integrity and Trust
2. Introduction to Dynamic Launch
3. Theory of TrenchBoot
4. TrenchBoot and Upstream
5. Questions and Answers

Introduction to Integrity and Trust

Daniel P. Smith

Trustworthiness of Booting

- A frequent question is, “Why is UEFI secure boot not sufficient?”
  - To answer that, we will first provide an understanding of Trust and then show a comparison of the Trust constructs of UEFI secure boot and Dynamic Launch
- An understanding of Trust was well established decades ago yet even today the concept is often conflated
  - To understand Trust, in particular System Trust, one must understand the impersonal structure “Strength of Mechanism” is and how it relates to Roots of Trust

The Meanings of Trust: System Trust

- System Trust addresses the extent to which one believes that the proper impersonal structures are in place to enable one to anticipate a successful future endeavor [1].
  - Two types of impersonal structures can be differentiated: (a) structural assurances, and (b) situational normality.
- A Root of Trust is a System Trust whereby the “Strength of Mechanism” is a structural assurance to enable one to believe in its operation.

Understanding Roots of Trust

- Definition: (Trusted Computing Group) A component that must always behave in an expected manner because its misbehavior cannot be detected [2][4].
- What does that mean and how does that relate to “Strength of Mechanism”?
  - “Strength of Mechanism” is the behavior of a component subject to stresses and strains[3]
    - Of concern is how resilient is a component’s behavior to being affected by these stresses and strains, i.e. how mutable is the component
    - It follows then that to drive up the Strength one must increase immutability

Roots of Trust and Integrity

- Since we “Trust” the RoT’s operation to be correct and that operation is used to make an assessment of another component, it then can be used to establish the integrity of that component and thereby impart a degree of Trust to that component
  - This is what the TCG refers to as a Transitive Trust [4]
- This process can be performed recursively to produce what is colloquially referred to as a trust chain that is anchored to the RoT
  - This is analogy serves very well because just like in life,
    - If anchor has a weak strength then the chain is easier to become dislodged under stress
    - If the chain becomes long, the probability raises that a link in that chain could fail
- It is through this trust chain that assertions to the integrity of a system may be made

Load Time vs. Run Time Integrity

- When making assertions about the integrity of a system, it is also important to consider what is being asserted, in particular whether the correct code started was started and/or if the correct code is currently running.
  - This distinction is important when deciding to trust the assertions being made by a system

Load Time vs. Run Time Integrity

Continuation

- Load time integrity is when correctness is established after loading but before execution
  - UEFI secure boot
  - TCG Dynamic Launch
  - Linux IMA
- Run time integrity is when correctness can be (re-)established any time during the lifecycle of the component
  - Kernel Integrity Monitoring
  - Process Integrity Monitoring

Introduction to Dynamic Launch

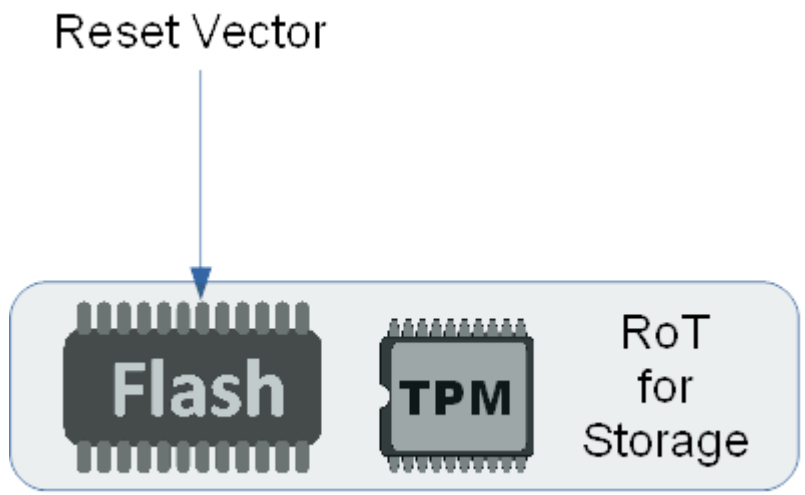
Daniel Kiper

Terminology

Mapping concepts to TCG specification [4] and vendor terms

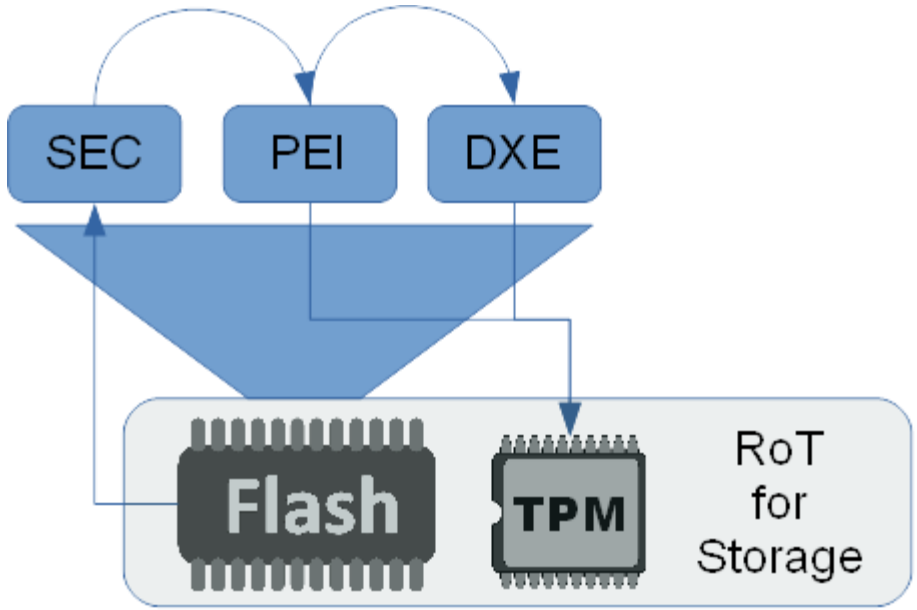
Description	TCG	Intel TXT	AMD-V
Process of starting a software environment at an arbitrary time in the runtime of a system	Dynamic Launch (DL)	Late Launch	Secure Startup
Platform dependent event that triggers the DL	DL Event	GETSEC[SENTER]	SKINIT
Performs initial configuration actions that are platform 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



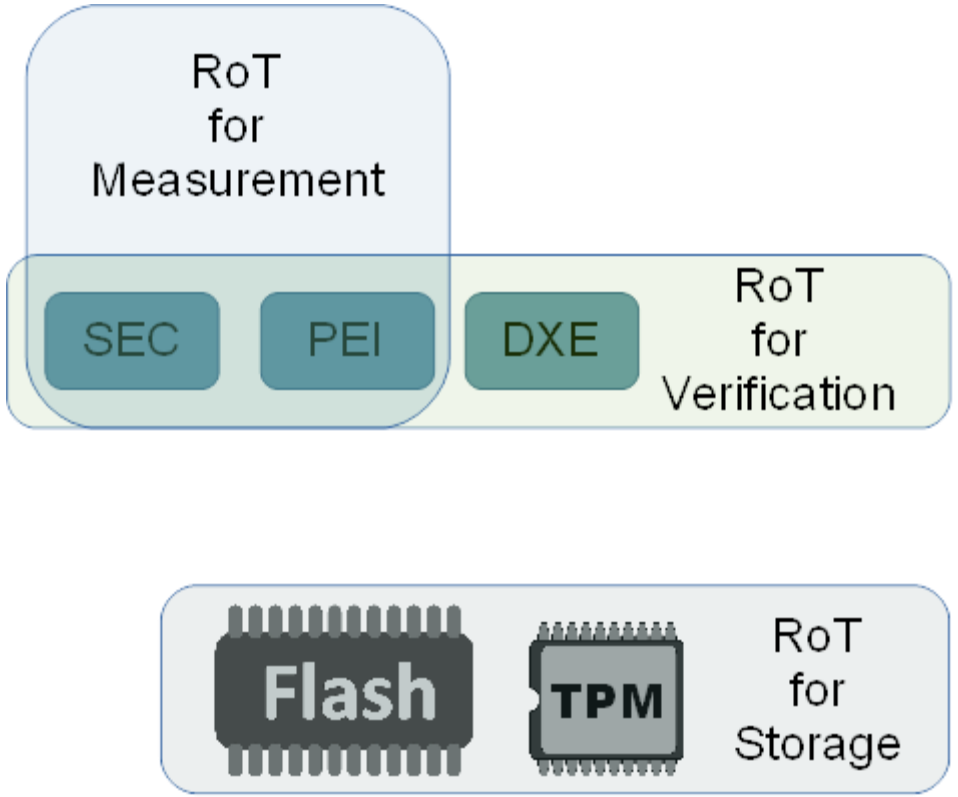
- The CRTM is not measured until during PEI
  - Therefore SEC and PEI must be trusted
  - CRTM is of SEC and PEI, thus it is self referential
  - Relies on integrity of Boot Flash
  - Relies on TPM to protect measurements
- The DXE phase enforces UEFI secure boot verification

UEFI Secure Boot Trust



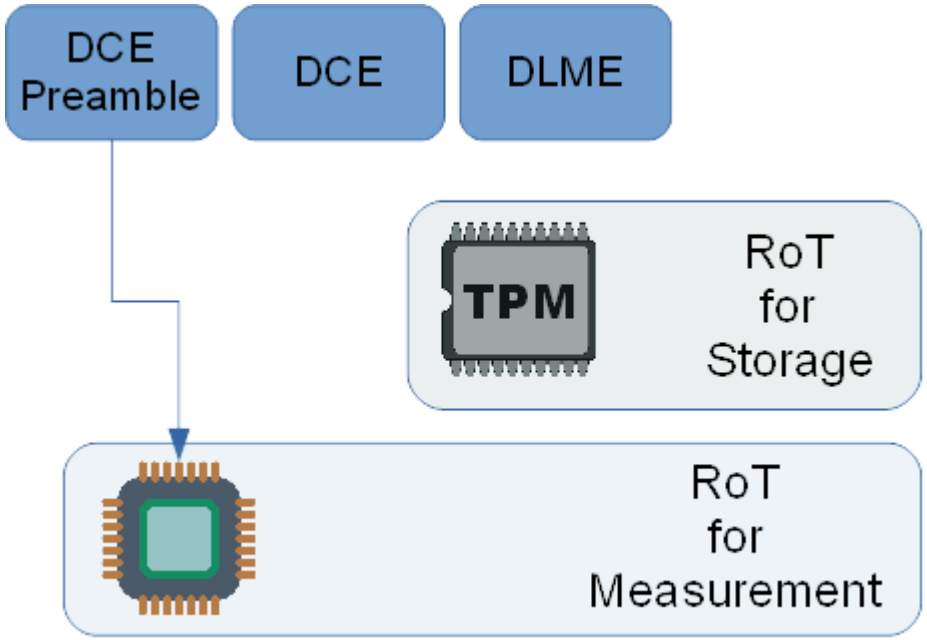
- The CRTM is not measured until during PEI
  - Therefore SEC and PEI must be trusted
  - CRTM is of SEC and PEI, thus it is self referential
  - Relies on integrity of Boot Flash
  - Relies on TPM to protect measurements
- The DXE phase enforces UEFI secure boot verification

UEFI Secure Boot Trust



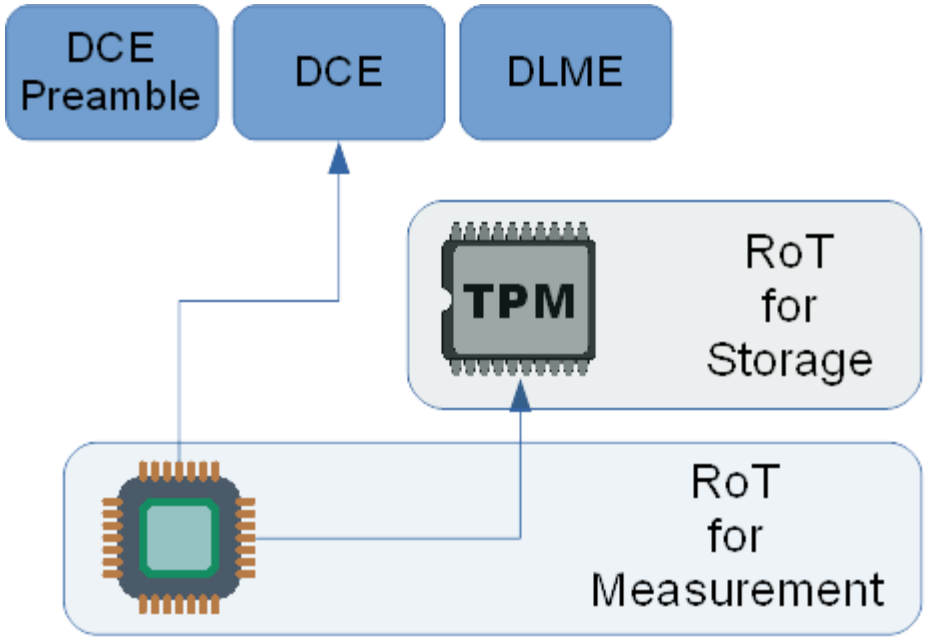
- The CRTM is not measured until during PEI
  - Therefore SEC and PEI must be trusted
  - CRTM is of SEC and PEI, thus it is self referential
  - Relies on integrity of Boot Flash
  - Relies on TPM to protect measurements
- The DXE phase enforces UEFI secure boot verification

Dynamic Launch Trust



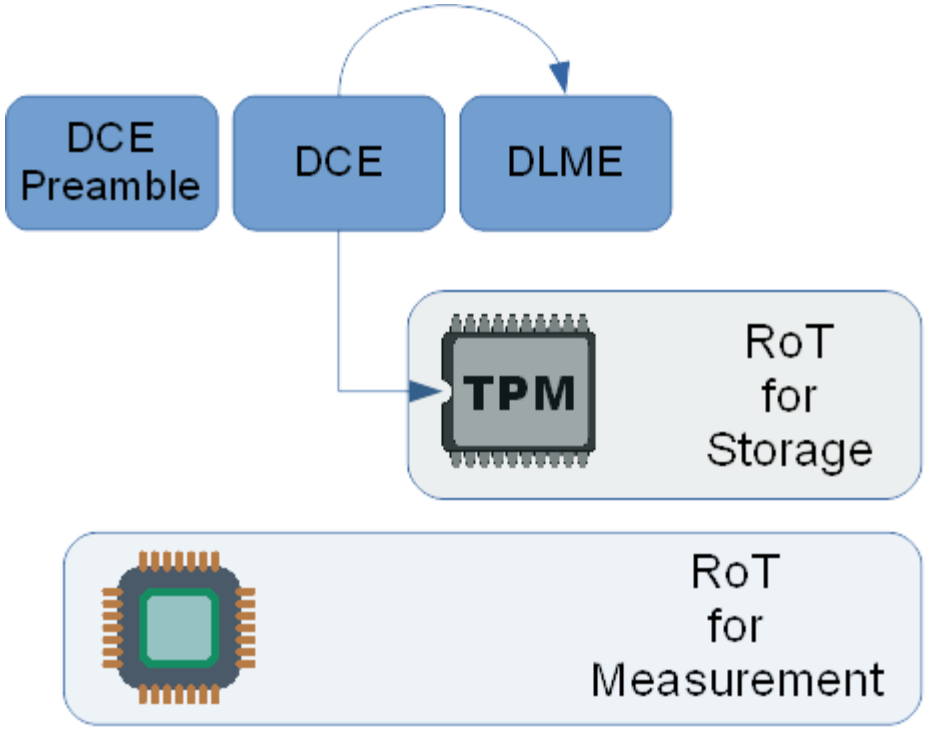
- DCE Preamble may be a bootloader or an executing OS
- The CRTM is taken by the CPU
  - Relies on the TPM to protect measurements
- On Intel there is also an additional authentication protocol between the DCE and CPU

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 results is a very high integrity assertion of the DLME
  - Removes boot firmware from the TCB with the exception being the SMI Handler

Theory of TrenchBoot

Daniel P. Smith

Motivation

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

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 has been 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

Launch Integrity Ecosystem

- The hardware categories for Launch Integrity are evolving
  - Root of Trust (discrete TPM, ME PTT, PSP fTPM)
  - Secure Coprocessors (ME, PSP, T2, Nitro)
  - Boot SPI interposers (OpenTitan, Cerberus, SureStart)
  - Hybrid (AzureSphere MCU, Arm Corstone-700 M-class Secure Enclave)
- Vendors are active in incorporating hardware-rooted integrity into their products
  - Google has Shielded VMs
  - Microsoft has System Guard Runtime Attestation
  - VMWare has ESXi Host Attestation Status

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.

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
      - Useful for checking integrity before persisting state to disk

Existing Open Source Dynamic Launch

- XMHF (AMD & Intel)
- OSLO (AMD)
- OpenDTeX Secure Boot (Intel)
  - tboot fork
- uber eXtensible Micro-Hypervisor Framework (AMD, Intel, ARM/Raspi)
  - XMHF fork
- tboot (Intel)

- Their focus is on First Launch use case
- The Intel-based ones are Exokernels that trap ACPI S states
- All have limited or no attestation capabilities

First Launch Use Case

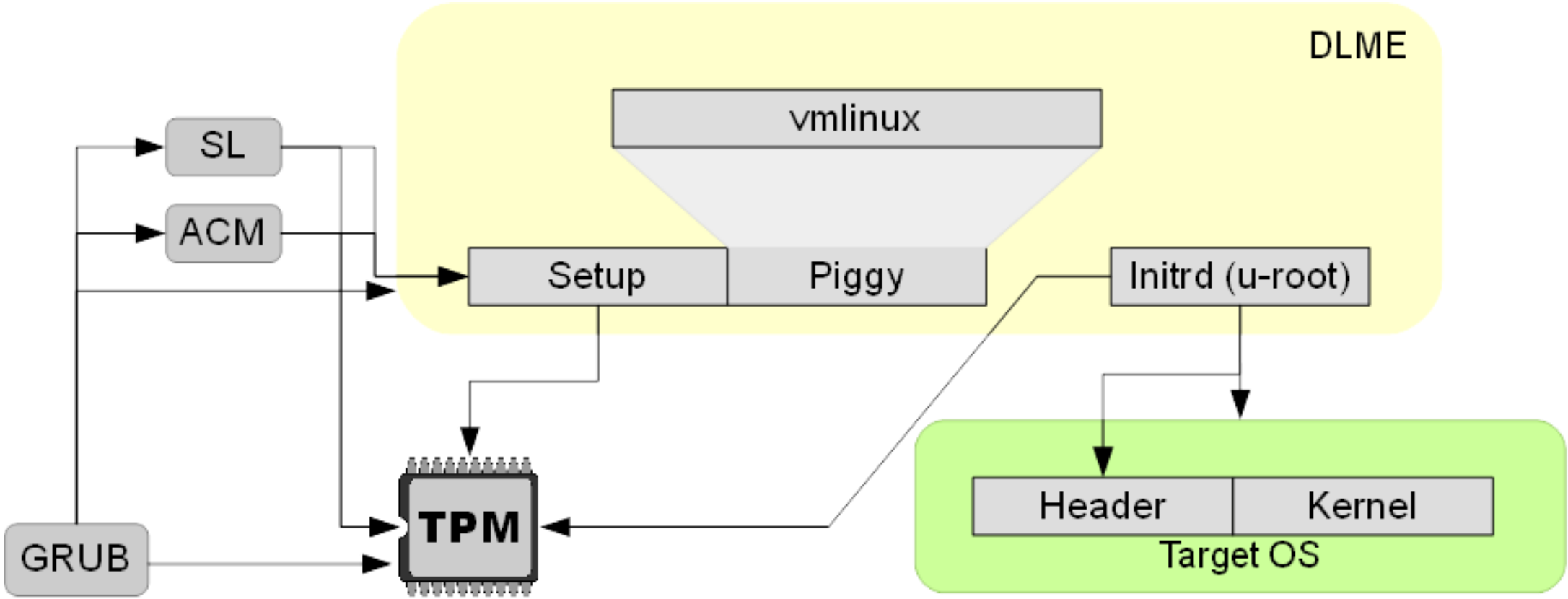
- The initial implementation being worked to demonstrate a common use case
  - This is the traditional approach that uses Dynamic Launch to root the target kernel in hardware
- TrenchBoot approach expands the traditional approach
  - Leverages Linux existing UEFI support to handle EBS hand-off
  - Leveraging u-root to provide a more flexibility means for measuring and attesting the environment
    - Block devices, Individual files
    - SMBUS/DMI information
    - Unseal-based, external device and network based attestation

First Launch Use Case

Continuation

- Leverages Linux kexec interface for launching subsequent kernel
- On Intel platforms, SEXIT is called to close access to DRTM PCRs

Basic Flow of First Launch – Intel TXT



TrenchBoot and Upstream

Daniel Kiper

Who is contributing to TrenchBoot

ORACLE

ORACLE

GRUB

- 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
- Summary of changes
  - Expand the Linux loader to support the Secure Launch kernel_info structure
  - Add additional commands to identify Secure Launch and load the DCE module, Intel ACM or AMD Secure Loader
  - Add DL Event relocators, one for Intel SENTER and one for AMD SKINIT
- AMD development is further along than Intel
- Has not gone upstream yet

SLBOOT

- The initial focus was on making Linux capable of being launched by DL (i.e. Secure Launch), eventually we needed to be able to test and thus required a DCE Preamble capable of loading Linux
- The fastest way was to fork tboot, stripping it down to just its “prelaunch” code
- This works only for Intel TXT platforms and is an interim solution until GRUB work can be completed

The kernel_info patch set

- Secure Launch requires new information to be passed from the kernel to the boot loader
- The kernel's setup_header has been the method to convey this info but has a hard limit on the amount of info it can carry and space has been almost out for a very long time
- Working with H. Peter Anvin the kernel_info structure has been proposed as a way for the boot protocol to be extended regardless of the setup_header limitations

Secure Launch

- Two sibling patch series are in parallel development
  - Intel TXT focused “core” series led by Oracle and Apertus Solutions
  - AMD Secure Startup focused “extension” series led by 3mdeb with support from Oracle and Apertus Solutions
- Summary of changes introduced
  - Compressed Kernel
    - SL entry point: Minimal code to handle entry from DL Event
    - Minimal SHA and TPM logic
      - Parts being reconsidered to reduce code size
  - Setup
    - SL handler: remainder of setup code for processing hand over from DL Event

u-root

- The u-root project is being used as the initramfs for the DLME and referred to as the Security Engine for the First Launch implementations
- Summary of changes
  - A uinit app that functions as the “Core Engine” for processing a policy file that describes what “Evidence Collectors”, any “Attestors”, and what “Launcher” to run
  - Introduces a measurement library that provides a core set of “Evidence Collectors”
  - Will use the existing booting capabilities in u-root to implement a kexec “Launcher”
- Attestation in DLME will be provided in later releases

Future Work

- A TODO list of what is planned
  - Extending kexec with a DCE Preamble implementation
  - Adding DCE Preamble to iPXE
  - Enable Xen to be kexec'ed from DLME
  - Adding DCE Preamble to Xen

Questions and Answers

Documentation & Resources

Subtitle text is Calibri bold 24 pt

- [1] Harrison Mcknight, D., and N. L. Chervany, *The meanings of trust*, Citeseer (1996): 1-86
- [2] *TCG Glossary*, https://trustedcomputinggroup.org/wp-content/uploads/TCG-Glossary-V1.1-Rev-1.0.pdf
- [3] *Trusted Network Interpretation Environments Guideline (TNI)*, NCSC-TG-011, National Computer Security Center
- [4] *TCG D-RTM Architecture*: https://trustedcomputinggroup.org/wp-content/uploads/TCG_D-RTM_Architecture_v1-0_Published_06172013.pdf

Safe Harbor Statement

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Integrated Cloud

Applications & Platform Services


ORACLE®