TrenchBoot

How to nicely boot system with Intel TXT and AMD SVM
Linux kernel secure launch dive-in

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

Linux Plumbers Conference 2019, September 9-11, 2019

ORACLE

Safe Harbor Statement

The following 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.

Presentation Agenda

1. The TrenchBoot Project - Short Intro
2. The TrenchBoot - Linux Kernel Secure Launch Dive-in
3. Discussion

TrenchBoot

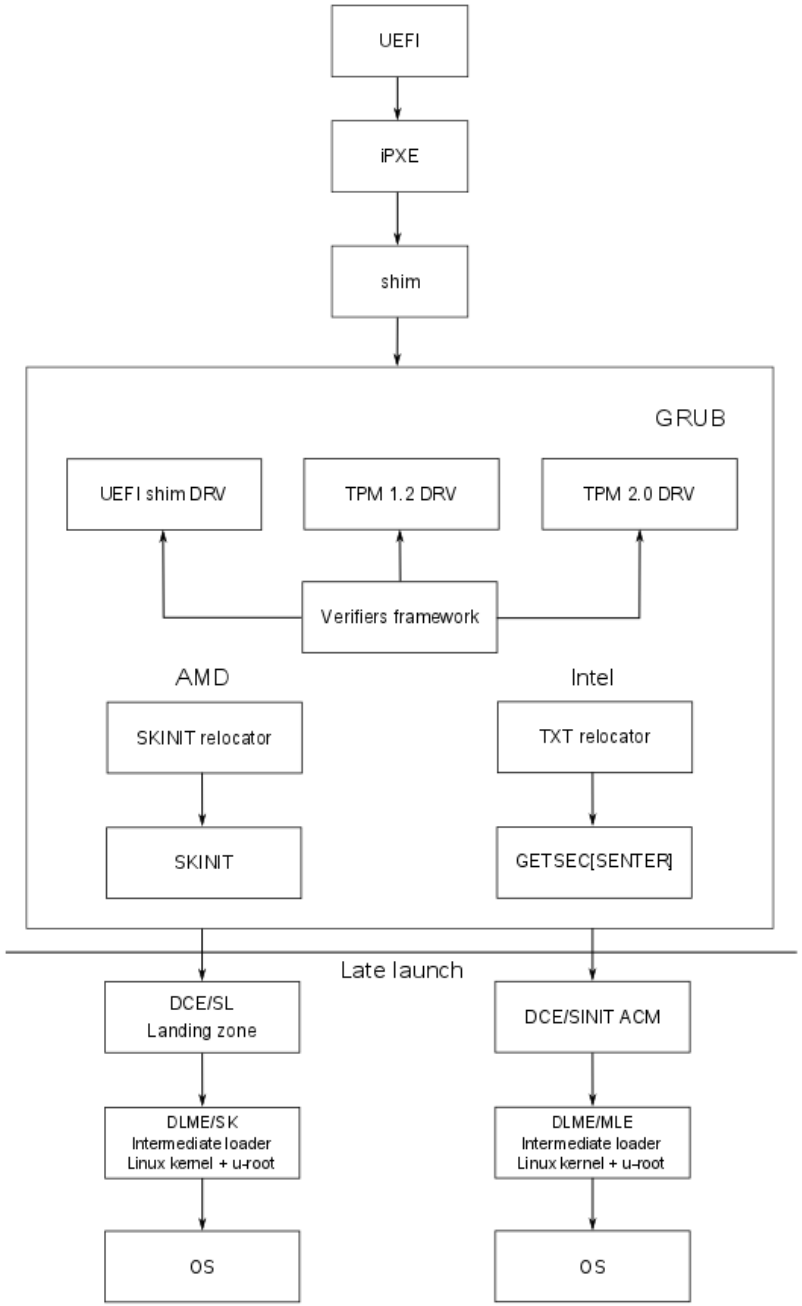
- TrenchBoot is a cross-community integration project focused on launch integrity
  - This means there is no “one thing” that is TrenchBoot
  - The name was a play off of dealing with the muddy mess of trying to find a way to unify boot integrity
  - The purpose is to develop a common, unified approach 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

TrenchBoot

The Boot Flow



TrenchBoot

More About the Project

TrenchBoot - How to nicely boot system with Intel TXT and AMD SVM

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

https://sched.co/RHb0

TrenchBoot

Linux Kernel Secure Launch Dive-in

The Trenchboot Secure Launch feature in the Linux kernel allows the starting of the post launch MLE stage of a secure late launch to establish the DRTM. The feature is present in logic blocks or phases in the kernel as it is booting so it will be described this way. The following slides discusses the Intel TXT implementation only.

TrenchBoot

Entry Point (sl_stub)

The sl_stub entry point is a lot like the EFI stub entry point. The very first bit of this called sl_stub_entry is in the .head.text section. Since this area is tightly organized, this stub jumps directly to the main sl_stub function in the .text section.

TrenchBoot

Entry Point (sl_stub) - Continuation

- The sl_stub function is responsible for handling the state of the BSP CPU upon entry from TXT. The specifics of the environment are covered in the TXT specification. These tasks include:
  - Loading a GDT for use
  - Checking and preparing parts of the TXT heap
  - Re-enabling SMI and NMI
  - Waking up the remaining APs and restoring AP state in the same manner as the BSP and loading an IDT
  - Parking the APs in a safe location (see below for more details)
  - Restoring MTRRs and the Miscellaneous Features MSR which are clobbered during the launch

TrenchBoot

Measurements and Verification (sl_main)

One of the main features of a secure launch is that everything must be measured before it can be used. The function sl_main is also called out of the compressed kernel before decompression as early as possible before values like the boot params or the command line are used. This code measure these values along with other related values, e.g. it will measure any external initramfs present. It also validates that the loaded MTRRs have correct values.

TrenchBoot

Late Verification and Setup (slaunch_setup)

- All verification and setup that does not need to be done very early in the compressed kernel is done in the later secure launch code. The function slaunch_setup is called out of setup_arch in setup.c. It performs a number of tasks:
  - Verify the platform and whether a measured launch was done via Secure Launch
  - Verifies the VTd PMR registers
  - Reserving certain regions like the TXT heap and TXT registers in the E820 map
  - Validate the E820 map against the MDR (memory descriptor records) passed from TXT to the MLE
  - Make the TXT provided copy of the ACPI DMAR table available to the Intel IOMMU driver

TrenchBoot

SMP Bringup

While in SMX mode, #INIT cannot be asserted on the APs and SIPIs cannot be sent to start them. The earlier code above wakes the APs up and eventually parks them in a safe location provided by the boot loader pre-launch code. The smpboot.c code is modified to call TXT specific routines when it detects a secure launch is in progress.

The APs are basically waiting in a busy pause loop. The TXT specific code sends an NMI IPI to the waiting APs and vectors them to a bit of Secure Launch specific code in the rmpiggy that mimics the normal 16b entry for the SIPI. The rest of the AP startup proceeds as usual.

TrenchBoot

Setup securityfs

The TPM log must be made available to the security framework that will run in user mode. This is done by exposing the log via a node in the securityfs.

Discussion

Integrated Cloud

Applications & Platform Services


ORACLE®