ORACLE

TrenchBoot

GRUB changes and Intel TXT implementation

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

FOSDEM 2020, February 1st, 2020

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.

GRUB

- 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
- Summary of changes
  - Expand the Linux loader to support the Secure Launch kernel_info structure (kernel_info structure itself landed in Linux kernel 5.5)
  - Add the additional commands to identify Secure Launch and load the DCE module, Intel ACM or AMD Secure Loader (under development but there is a chance that this thing will be automated)
  - Add DL Event relocators, one for Intel SENTER and one for AMD SKINIT (under development)
- The AMD SKINIT and Intel TXT implementations are developed separately and need synchronization at some point in time
- The Intel TXT RFC will be posted probably at the end of February or beginning of March
- The AMD SKINIT code will be rebased on top of it

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 the SMI and NMI
  - Waking up the remaining APs and restoring the 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 the 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 the values like the boot params or the command line are used. This code measures 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 do not need to be done very early in the compressed kernel are 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:
  - Verifying the platform and whether a measured launch was done via Secure Launch
  - Verifying the VTd PMR registers
  - Reserving certain regions like the TXT heap and TXT registers in the E820 map
  - Validating the E820 map against the MDR (memory descriptor records) passed from the TXT to the MLE
  - Making the TXT provided copy of the ACPI DMAR table available to the Intel IOMMU driver

TrenchBoot
SMP Bringup

While in the 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 the TXT specific routines when it detects a secure launch is in progress.

The APs are basically waiting in a halt state. 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
kexec

During the execution of kexec, the kernel detects if it is in SMX mode. This happens very late after kexec has disabled almost everything and halted all the APs (which follows what the spec says it expects). Then the SEXIT code does some final TXT register operations. After that the kernel invokes the GETSEC[SEXIT] instruction. This causes the BSP to leave SMX mode and later new OS kernel is started.

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.

ORACLE