Passing and retrieving information from bootloader and firmware

Linux Plumbers Conference 2020

Michał Żygowski

3MDEB

Agenda

- Problem statement
- Specifications confusion
- TPM event log example
- DRTM event log

Introduction

Michał Żygowski
Firmware Engineer

- @miczyg
- michal.zygowski@3mdeb.com
- linkedin.com/in/miczyg
- facebook.com/miczyg1395

- Braswell SoC, PC Engines and Protectli maintainer in coreboot
- interested in:
  - advanced hardware and firmware features
  - coreboot
  - security solutions

Problem statement

Linux has many booting methods: kexec, EFI stub, legacy boot

Each can pass different information in various ways

In terms of capabilities and features firmwares are similar: UEFI, coreboot, LinuxBoot. They typically pass/expose same information for operating system. Yet there are still problems with how to retrieve information from firmware and bootloader.

Specifications confusion

There are `specifications` and ***specifications***. For example retrieving the TPM 2.0 event log:

- Using UEFI Boot Services on Exit Boot Services
- Using TPM2 ACPI table pointed address

Apparently the latter option is not valid [1]:

```
Event log consumers may now retrieve the event log via the
TCG2 EFI protocol GetEventLog API (No longer retrievable
directly from the ACPI tables)
```

1) https://www.uefi.org/sites/default/files/resources/Phoenix_Plugfest_TPM2_March_2016.pdf

Specifications confusion

- On one side of the medal we have the TCG defined ACPI table with event log area
- On the other side we have UEFI specifications which invalidate the event log from APCI table
- UEFI everywhere. What if we have no UEFI? (kexec or legacy boot?)
- The UEFI way is more troublesome than needed[2]
- Using ACPI way leaves open door for custom event logs and parsers to fit everyone's needs (TCG TPM2 event log format is very UEFI-oriented)

```
UEFI provided event log has a few somewhat weird quirks.

Before calling ExitBootServices() Linux EFI stub copies the event log to
a custom configuration table defined by the stub itself. Unfortunately, the
events generated by ExitBootServices() don’t end up in the table.
```

2) https://www.kernel.org/doc/html/latest/security/tpm/tpm_event_log.html

TPM event log example

Just recently, the support for TPM 2.0 event log retrieving from ACPI landed in Linux[3]:

```
commit 85467f63a05c43364ba0b90d0c05bb89191543fa
Author: Stefan Berger <stefanb@linux.ibm.com>
Date:   Mon Jul 6 19:58:07 2020 -0400

    tpm: Add support for event log pointer found in TPM2 ACPI table

    In case a TPM2 is attached, search for a TPM2 ACPI table when trying
    to get the event log from ACPI. If one is found, use it to get the
    start and length of the log area. This allows non-UEFI systems, such
    as SeaBIOS, to pass an event log when using a TPM2.

    Cc: Peter Huewe <peterhuewe@gmx.de>
    Cc: Jason Gunthorpe <jgg@ziepe.ca>
    Signed-off-by: Stefan Berger <stefanb@linux.ibm.com>
    Reviewed-by: Jerry Snitselaar <jsnitsel@redhat.com>
    Signed-off-by: Jarkko Sakkinen <jarkko.sakkinen@linux.intel.com>
```

3) https://lkml.org/lkml/2020/7/7/15

DRTM Event Log

- TCG D-RTM Architecture specification[4] is there but...
- Intel goes their own way with tboot
- D-RTM event log should be separate from firmware TPM event log
- Intel TXT and tboot saves the events in the TXT heap (not generic)
- Lack of unified approach to store and retrieve DRTM event log

Solution?

4) https://trustedcomputinggroup.org/wp-content/uploads/TCG_D-RTM_Architecture_v1-0_Published_06172013.pdf

DRTM Event Log

ACPI is the way to go. The D-RTM Architecture specification defines a DRTM ACPI table.

- holds various information about DRTM capabilities (platform-agnostic)
- can point to the DRTM event log area
- retrieve it in the same manner as TPM2 event log and populate in sysfs

What do you think?

Thank you