Contribution ID: **267** Type: **not specified**

PCI Data Object Exchange (DOE), Component Measurement and Authentication (CMA) / SPDM 1.1 - Mediating access and related issues

Thursday, 23 September 2021 07:35 (45 minutes)

DOE (PCI ECN) provides a standard mailbox definition, so far used for query / response type protocols. There can be multiple instances of a DOE on each PCI function, and each instance can support multiple protocols. Currently we have published definitions of the Discovery, CMA, IDE (available from the PCI SIG) and CDAT protocols (available from UEFI forum). Some of these protocols are intended for Linux kernel access (e.g. CDAT), others are less clear but there are possible use cases (CMA, IDE).

Patches to support DOE mailboxes in PCI extended config space have raised questions about how to ensure that these mailboxes, which may be of interest to various software entities (userspace / kernel / firwmare / TEE etc) can be safely used.

The DOE design does not easily allow for concurrent use by different software entities (even if possible, we cannot rely on other software elements doing this safely), so it seems some level of mediation is required. The topics for discussion include:

1. Do we want to enable any direct userspace access to these mailboxes or should we address on a per protocol basis (if at all)?
2. Do we need to 'prevent' userspace being able to access these registers whilst the DOE is in use?
3. How do we know the kernel should not touch a given mailbox (in use by other system software)? Perhaps a code first submission to ACPI to define a mediation mechanism? Is this sufficient for expected use cases? (What other suggestions do people have?)

A very brief overview of DOE and proposed kernel support will be presented to make sure everyone is aware of the background - then straight into the discussion of the above questions.

The PCI ECN defining CMA adds the ability (using a DOE mailbox) to establish the identity and verify the component configuration and firmware / executables.

This is done using the protocols defined in the DMTF SPDM 1.1 specification: https://www.dmtf.org/sites/default/files/standards/documen which is also used for the same purpose on other buses such as USB, but we are not aware of any work to support those buses yet. The design is extensible to other buses with an abstracted transport layer (via a single function pointer).

The CMA use of the SPDM 1.1 protocol defines a certificate based public private key authentication mechanism including signed measurements of PCIe component state (firmware and other implementation defined elements) and setup of secure channels for continuing runtime measurement gathering and for other related PCI features such as Integrity and Data Encryption IDE.

An initial implementation will be posted shortly for review, and there are a number of open questions that may benefit from a discussion in this forum:

1. Is there a sufficiently strong case to support CMA natively in the kernel at all?
   Some approaches might push this facility into a trusted execution environment. VFs can implement CMA however, to provide this level of authentication and measurement, when in use by a VM. It would be useful to understand other use cases as they motivate the software design and testing.
2. Approach to providing authentication of device certificates? SPDM uses x509 certificates and so relies on a chain of trust. What trust model should we apply? Current code assumes a separate keychain dedicated to CMA and root key insertion from userspace (probably initrd).
3. Method of managing / verifying measurements. The nature of the measurements is implementation defined. In some cases they are not expected to change unless the firmware is flashed, but in others they may change with device configuration. Whilst closely related to the challenges of IMA for files, is it appropriate to reuse that subsystem and tooling?
4. As it's related, is there interest in supporting kernel managed IDE (link encryption)?

5. When do we actually want to make these measurements? (On boot, on driver probe, on reset, on first use of a particular feature, on demand from userspace etc?) Currently they are done on driver probe only.

Other, more detailed questions can be addressed as part of normal discussion on list.

References:
https://lore.kernel.org/linux-pci/CAPcyv4i2ukD4ZQ_KfTaKXLyMakpSk=Y3_QJGV2P_PLHHVkPwFw@mail.gmail.com/
https://lore.kernel.org/linux-pci/20210520092205.000044ee@Huawei.com/

I agree to abide by the anti-harassment policy

I agree

Primary authors: CAMERON, Jonathan (Huawei Technologies R&D (UK)); WILLIAMS, Dan (Intel Open Source Technology Center)

Presenters: CAMERON, Jonathan (Huawei Technologies R&D (UK)); WILLIAMS, Dan (Intel Open Source Technology Center)

Session Classification: VFIO/IOMMU/PCI MC

Track Classification: VFIO/IOMMU/PCI MC