LINUX PLUMBERS CONFERENCE September 20-24, 2021

DOE / CMA / SPDM

Jonathan Cameron / Dan Williams

Why discuss this?

DOE necessary for CXL 2.0 enablement (CDAT).

CMA likely to start appearing on devices soon.

PCIe / CXL IDE likely to appear on hosts and devices soon.

Good to prove out the DOE implementation using several protocols.

Turned out there were lots of unknowns!

Scope

Discussion is on a stack of elements.

Questions at all layers. Each section has a ‘questions slide’

Not sure how far we will get! (hopefully beyond slide 8…)

Address low level first so we can make progress on implementation.

Try to avoid too much spec / code detail (only have 45 mins!)

Use cases also interesting… (do we want ALL of this in kernel?)

The Stack

Data Object Exchange (DOE) used as transport for…

(CDAT and) **Component Measurement and Authentication (CMA)** provides DOE protocol to transport…

Security Protocol and Data Model (SPDM)

(Also: **Integrity and Data Encryption (IDE)*)**

*Anyone interested in kernel managed IDE?

Topic 1: DOE

PCI-SIG ECN defines:

- A mailbox interface in an Extended Cap / PCI Config Space
  - In-FIFO / out-FIFO
  - Go signal / Reset Signal
  - Interrupt (response ready, error, no longer busy)
- Protocol format including discovery protocol.
  - Multiple protocols on a single DOE (protocol specs define restrictions)
  - Protocol defined by **Vendor ID** and Vendor controlled **Data Object Type**.
- Other published specs define protocols: CMA, IDE, CXL Table Access (CDAT)

DOE Issue – no mediation

- No standard way to support safe concurrent access.
- DOE instances may be used by:

  Kernel / Userspace / Hypervisor / TEE / Firmware / Other…

- Can’t base decision on what the DOE supports, as discovery protocol can break communications!
  - By the time we’ve queried what a DOE supports, it may be too late.

DOE Userspace Access

Option 1: Kernel implements protocol support with a per protocol user space interface (where appropriate)

Option 2: Kernel also provide generic access for some protocols?

Option 3: Kernel ‘may’ optionally bind to a given DOE. Do we need to prevent user space access? Do we want to support the model of Userspace access when kernel has not bound?

Protocols may be defined by Vendors.

How helpful should we be about supporting Vendor Protocols?

DOE Kernel Access: Why?

Places we want it today include:

- drivers/cxl/pci.c: retrieve CDAT once at driver init, and after set-partition commands
- drivers/pci/pci-driver.c: perform CMA for probe authorization *
- drivers/pci/pcie/ide.c & drivers/cxl/bus.c: establish IDE between capable ports *
- drivers/*/: marshal device-specific secure messages over SPDM including measurement *

(* One potential model)

DOE Kernel Access

Some protocols used by firmware / TEE etc (Firmware interfaces out of scope)

Protocols resilient to security issues, but not denial of service (accidental…).

Need a '**you may use it**' signal.

I only care about ACPI systems

Per EP _DSM doesn't work for hot plug (where do you put it?)

- Hot plug may be entirely OS native (or at least it might look like that to the OS)

Note we need to have a solution in place before broad DOE enablement:

- Allow list / kernel parameters in the meantime?

DOE Kernel Access

Proposal 1: System wide _DSM.

0: Firmware provides mediated interface for all PF DOEs.

1: Firmware uses no DOEs once system booted.

Big hammer. Won't work in all cases!

Proposal 2: ACPI table with buffer based update

Similar to _FIT for hot plug case.

Various approaches to uniquely identify device even when hot plugged.

Non trivial spec change: Code first submission possible.

More details in 'backup material'.

Topic 2: CMA / **SPDM**

Security Protocol and Data Model

Standard messages and protocol for:

- Device Authentication / Mutual Authentication
- Secure Channels
- Integrity Measurement – Firmware changed? State changed?
- (Key management etc)

Used over various transports to establish trust between hosts and connected devices.

- USB chargers verifying a phone can cope with higher currents (SPDM based on USB security model)
- Preventing attacks like Thunderclap in which a device is ‘faked’ and used to attack a driver.
  - “The case for memory segregation” Alex Markuze https://linuxplumbersconf.org/event/11/contributions/902/
- Attested device management at runtime.

Topic 2: **CMA** / SPDM

PCI-SIG Component Measurement and Authentication ECN defines:

- DOE protocol to carry SPDM messages.
  - Other transports allowed (e.g. MCTP), but probably not relevant to Linux today.
- Subset of SPDM that ‘must’ be supported (protocols etc)
- Leaf Certificate must contain information to match to device (prevent some impersonation attacks)

Why does Linux Care? Use cases!

When is not firmware's problem?

- Maybe we are the "firmware"? Linux boot or embedded
- Runtime hardware state changes.
  - Hot plug?
  - Resets
  - Out of Band firmware updates.
- Virtual machines. A VF can have it's own CMA DOE instance.

Strong enough to justify kernel support? – Yes

Virtual Machine Use cases?

1. Simple: Per VF CMA instance provide standard way to verify we trust the hardware + state.
2. PV case: Emulate CMA to provide measurements to the VF. Lots of other ways we could do this (but why not use the standard?)
3. Verifying Emulated Devices: Complex model needed, but ‘might’ provide a means of verifying a device is emulated by someone we trust.

What do we do if we fail?

No firm proposals for this yet!

- Policy controls needed.
  - What granularity?
- Somewhat similar to policy for ‘external’ buses.
- Might even be used to relax security measures such as Bounce Buffers for DMA.
- “A Firewall for Device Drivers” https://lwn.net/Articles/865918/

What do we do when?

- No CMA capable DOE instance available? Classes may support them, but not require (CXL type 3)
  - System policy. May block probe.
- No appropriate Root Cert on host?
  - System policy. May block probe.
- Verification fails?
  - Easy: SCREAM and block probe.

Certificate Management

SPDM authentication based on Asymmetric Crypto (RSA / ECC):

- Uses x509 certificates requested from device. Verify against root cert.
- Can use kernel key retention service.
- _CMA keyring: root certificates added via initrd - modelled on other keyrings in security/integrity/digsig.c (TODO: support other sources of certificate)
- Use keyctl / evmctl to do this.

Q: Single _CMA keyring or root certificates for particular devices only?

Per SPDM instance Keyring used to allow reuse of existing certificate verification.

- Abuse of interface as it is a chain, not a set of keys.
- But – free userspace interface which is useful for debugging…
- Need to figure out how to remove a keyring on hot unplug or how to reuse the infrastructure without using a keyring.

Measurements!

Form of measurements very flexible (can be raw binary).

- May need per device driver handling to know what matters…

Information available on ‘when’ they are allowed to change.

- E.g. Static until cold reset (no point in rereading otherwise)

If they were hashes would look like IMA (integrity management for files etc).

- Nice to avoid reinventing the wheel.
- We could just hash raw values to hammer them into the hole.

When to measure?

- Boot / driver probe or before (kind of obvious)
- Reset
- Polled / on driver driven events?

References - code

DOE Kernel**: [PATCH v4 0/5] PCI Data Object Exchange support + CXL CDAT**

https://lore.kernel.org/linux-pci/20210524133938.2815206-1-Jonathan.Cameron@huawei.com/

SPDM Kernel: **[RFC PATCH 0/4] PCI/CMA and SPDM library**

https://lore.kernel.org/linux-pci/20210831135517.0000716f@Huawei.com/#t

SPDM QEMU:

https://lore.kernel.org/qemu-devel/1624665723-5169-1-git-send-email-cbrowy@avery-design.com/

SPDM reference implementation (used by QEMU)

https://www.github.com/dmtf

Kernel tree: https://github.com/hisilicon/kernel-dev/tree/doe-spdm-v1

Qemu tree: https://github.com/hisilicon/qemu/tree/cxl-hacks

References - Specifications

PCI/CMA: https://members.pcisig.com/wg/PCI-SIG/document/14236

PCI/DOE: https://members.pcisig.com/wg/PCI-SIG/document/14143

PCI/IDE: https://members.pcisig.com/wg/PCI-SIG/document/15149

DMTF/SPDM 1.1:
https://www.dmtf.org/sites/default/files/standards/documents/DSP0274_1.1.0c.pdf

References : Papers

Thunderclap Paper: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_05A-1_Markettos_paper.pdf

Backup

Done / Todo?

RFC

- Generic DOE support + Discovery and CDAT protocols.
- CMA / SPDM – simple case up to CHALLENGE_AUTH

TODO

- Leaf certificate verification against Device ID (VID / serial number etc)
- Measurement management.
- Secure channel setup.
- Mutual Authentication (usecases?)

Test Platform!

No (known) hardware available yet…

Built on the CXL QEMU based emulation:

- Ben's talk tomorrow!

Avery design added DOE emulation:

- CXL Table Discovery Protocol (CDAT)
- CXL Compliance Protocol
- CMA via a proxy to …

- Spdm-responder from the spdm-emu tools from DMTF (using spdmlib reference implementation)

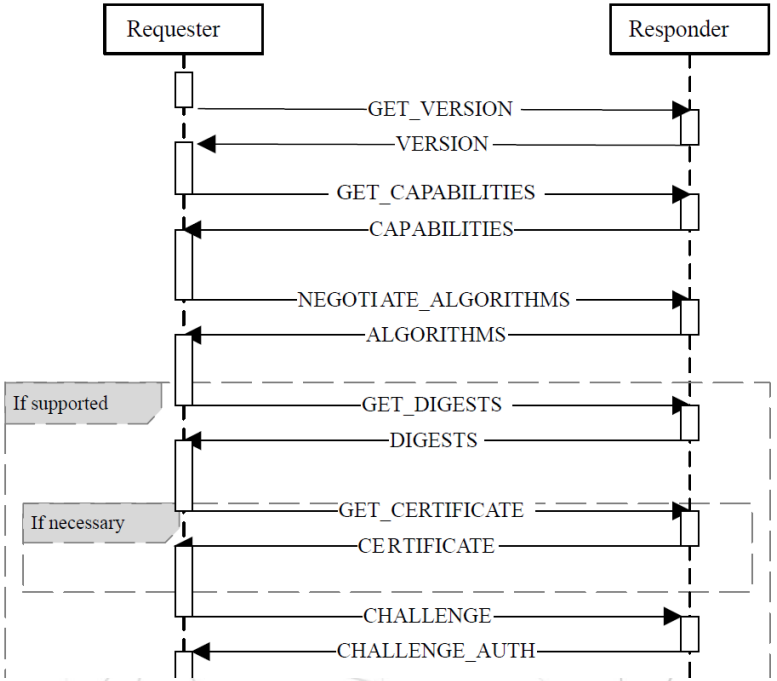
Hotplug DOE 'gate' flow.

- Hot plug of PCIe / CXL devices may be entirely OS managed.
  - It is then we 'could' assume other software entities will not be interested in DOEs. Is this valid? Assume not.
- _DSM entries have to be associated with an entity we know is there in (DSDT firmware node).
- Associate a _DSM with host bridge (PNP0A08) or root port.

1. Device hotplugged.
2. Normal notification flow and enumeration.
3. Device Driver Instantiated – needs to know if any DOEs are for OS use.
4. Query _DSM to return a buffer.
   - Note that system may do magic before returning this, including firmware discovering devices, starting IDE etc.
   - Buffer contains device identification + allow list for DOE instances on that device. (Hierarchical description may be needed to avoid enumeration dependency)
5. Device driver only accesses permitted DOE instances. May fail to probe depending on policy.

SPDM CHALLENGE_AUTH

1. Version negotiation
2. Capability negotiation
   - Lots of optional stuff (measurements, mutual auth etc)
3. Negotiate Hash and Asymmetric Algs
4. Get Hash of Certificates (can avoid redownloading chain)
5. Get Certificate chain.
6. Issue a challenge.



DMTF 1.1 Specification

Assuming everyone knows basic 101 crypto…

Why leaf cert must identify device?

Attack is to get a ‘wrong’ driver to load.
Note that IDE would break this as EP (A) would not be able to get the key so not decrypt the link.
Good to make the attack harder!