SPDM/CMA

Linaro-open-discussions

1st September 2021 – Jonathan Cameron (Huawei Tech R&D UK)

Change log

2021/09/01: Added diagram for certificate management after LOD call.

Background (why are we talking about this?)

- It was on the TODO list and we needed more evidence to back DOE implementation choices.
- Raised lots of questions: Hence Plumbers microconf proposal…
- Kernel:
  - https://lore.kernel.org/linux-pci/20210831135517.0000716f@Huawei.com/#t
- QEMU emulation (thanks to Avery Design)
  - https://lore.kernel.org/qemu-devel/1624665723-5169-1-git-send-email-cbrowy@avery-design.com/
- SPDM reference implementation https://www.github.com/dmtf

DMTF - Security Protocol and Data Model

- Based on USB security model + now used over various transports
- Provides:
  - Asymmetric Crypto / x509 Certificate based device (or mutual) authentication.
  - Signed measurements of device state (typically firmware running etc)
  - Can establish a secure channel (symmetric crypto key exchange etc)

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

PCI / Component Measurement and Authentication (CMA)

- PCI ECN https://pcisig.com/specifications/pciexpress
- Data Object Exchange (DOE) protocol for SPDM via PCIe config space.
  - Any function – including VFs
- Also basis of Integrity and Data Encryption (IDE) ECN
  - May well be to hide that from the OS (DOE topic at plumbers)
  - No hardware mediation of access, so software handling needed…
- CMA may also be in control of lower level software (TEE / other firmware) – but… There are use cases in kernel (Strong enough?)
- If in lower levels, firmware interface will be needed (separate topic!)

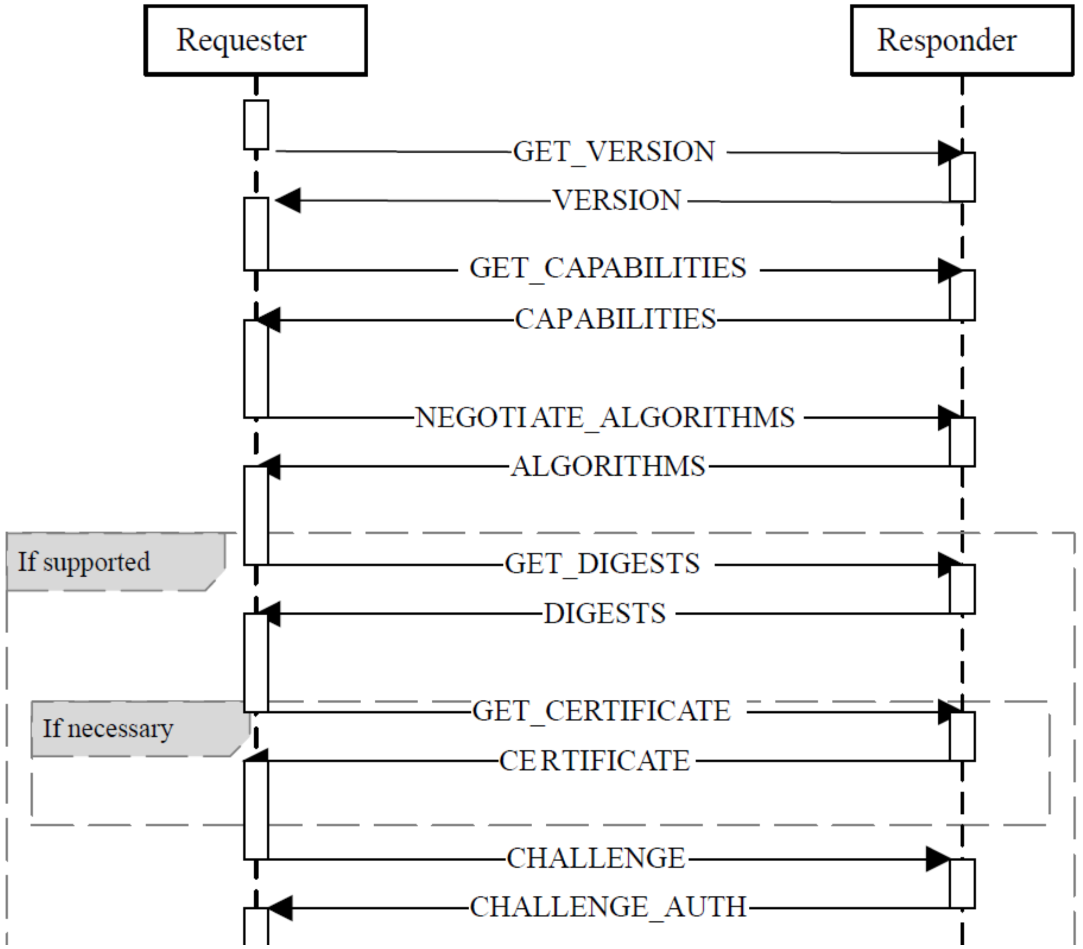
What’s it for?

- Verification that the PCI EP is what you think it is.
  - PCI Devices can be external (fake device attacks)
    - May allow relaxation of security protections (bounce buffer, driver auditing reqs)
  - Servers support hotplug and security model may include ‘within the case’ protections.
  - VMs want to be sure devices assigned to them are what they think they are.
  - Secure VMs may need to be sure PCI functions are what they think they are.
    - May be firmware mediated…
- Verification that it is in the state you expect.
  - Firmware changed – perhaps malicious?
  - PCI device can support out of band firmware update (PLDM etc) Not always ‘visible’ to the host.

Flows – Authentication:

Assuming everyone knows basic 101 crypto..?

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

Authentication Subtleties

Done in RFC

Future Improvements

Authentication flow requires a ‘lot’ of state in host software

- One call to do whole flow
- State machine to allow for ‘short cuts’.

Algorithms not known until negotiated + EP must support at least one of several options.

- Challenge needs a digest (hash) of all of prior messages.
- Initially hash type not known.
- Current implementation caches message until hash known, then does running calculation

Certificates known?

- Either pre arranged availability
- Root known by host, chain from that downloaded from EP
- Digest may allow use of cached certs
  - Whilst quite big current usecases aren’t typically time critical so perhaps we won’t bother.

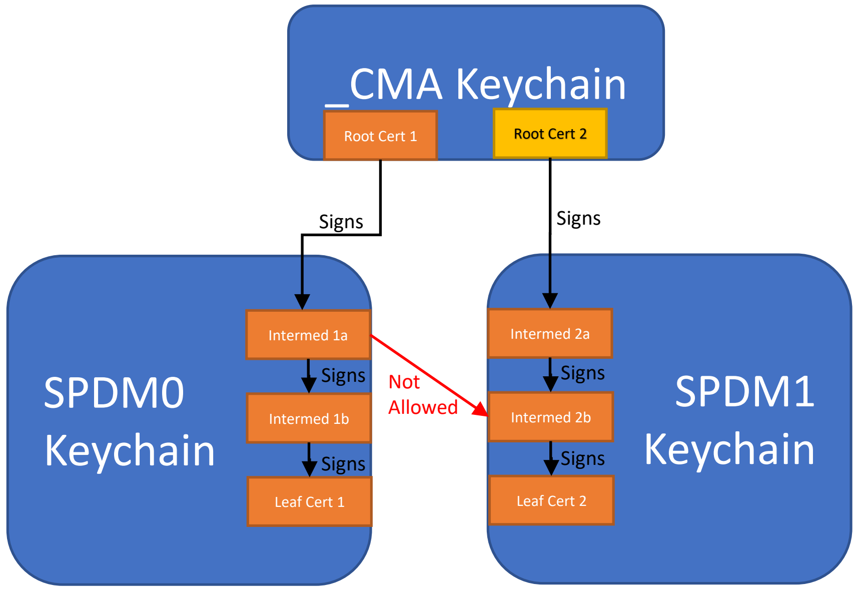
Certificate management

- Kernel already has key management framework (reuse!)
- Good for root certs and can be reused to check the chain from device.
- Oddity in SPDM – ECC signatures raw format, X509 typically ASN1
  - Hack: Encode the Raw signature into ASN1 to pass to standard core code.

Questions:

- Probably safe to have single keyring for root certificates (_cma) – separate from other use cases (evm etc)
- Do we trust valid certs from other devices? Seems inconsistent if a device is only 'good' because it's root cert came from a different device – load order might matter?
  - Proposal – per SPDM instance keychain (may need missing release infrastructure)
  - Valid to use keychains for this? Otherwise probably major refactor to expose keychain like functionality.

Chains considered when checking signatures



1. Only allow certificates from an SPDM instance to be used verify certificates and signatures from that instance.
2. Only leaf certificate used for challenge_auth verification.

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?

Questions

- What did I forget?
- Inputs needed from security folks…

- Question on layering and combinations of DOE to be addressed in plumbers DOE topic.