Arm CCA open-source enablement update

Dan Handley

Linaro Connect
MADRID 2024 | MAY 14-17 2024

Agenda

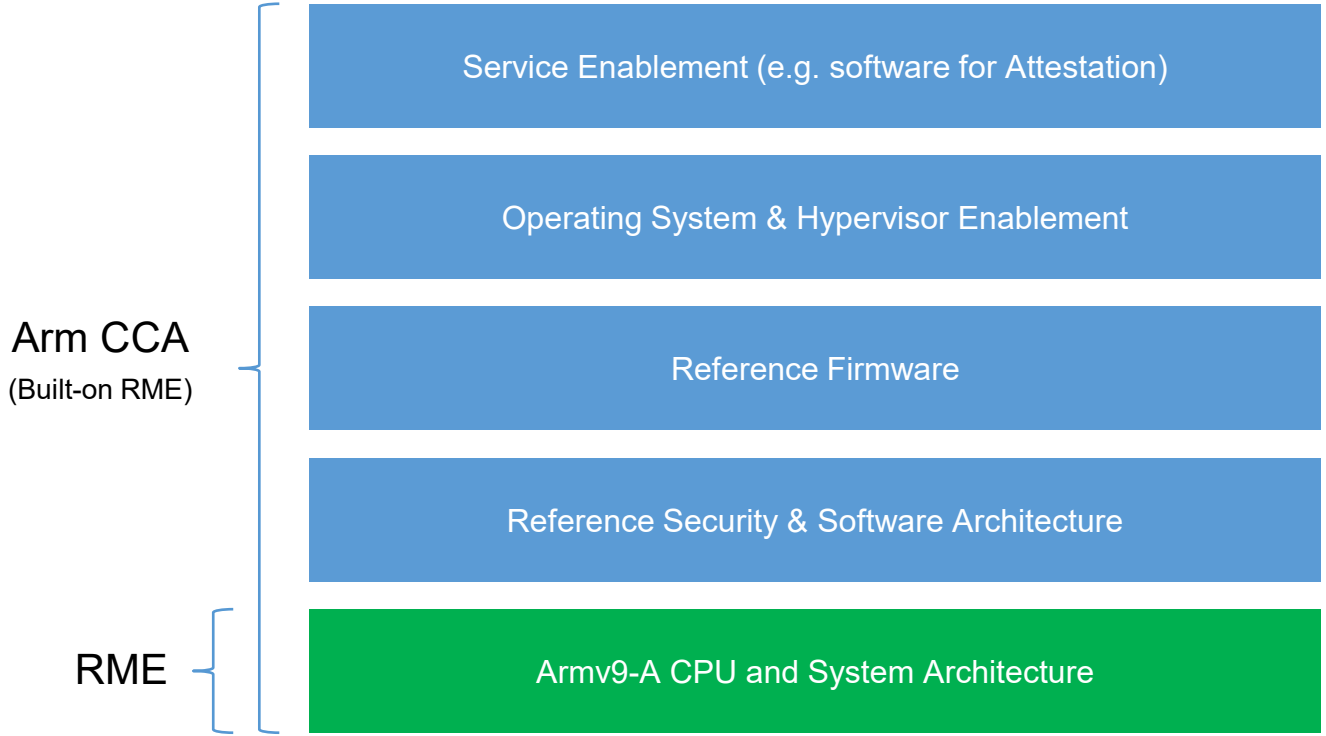
- Arm CCA recap
- Arm CCA 1.0 status and getting started
- Arm CCA 1.1 features
- What to expect next

Arm CCA recap

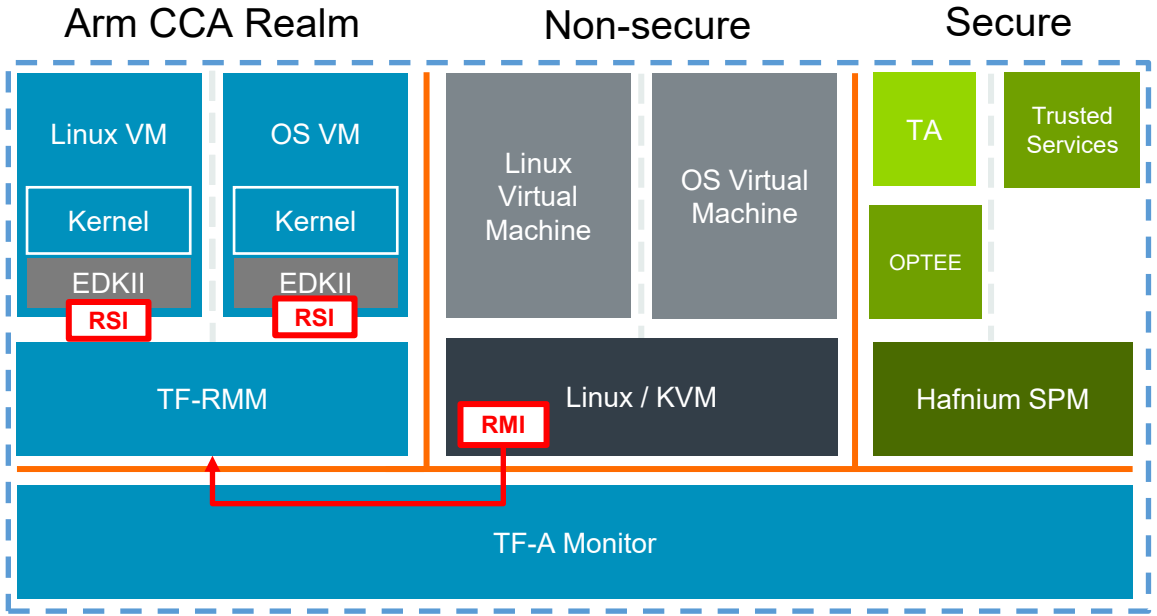
- Realm Management Extensions (RME):
  Armv9-A CPU and system hardware architecture, enabling the creation of isolated, dynamic, attestable, and trustworthy execution environments
  - Suitable for securing mainstream compute workloads such as containers or virtual-machines

- Arm Confidential Compute Architecture (Arm CCA):
  Builds on RME by providing a reference security and software architecture
  - Used to implement a hypervisor-based platform supporting confidential computing
  - Open architecture, assessed using formal methods
  - Supported by popular open-source projects including Trusted Firmware and Linux / KVM, reducing time to market and avoiding the expense of proprietary solutions

Arm CCA stack



Arm CCA open-source reference components



RSI: Realm Service Interface

RMI: Realm Management Interface

KVM

tianocore

RSE (Runtime Security Engine), TF-M Arm reference platform

TrustedFirmware.org

Arm CCA 1.0 (stack for RMM spec 1.0)

- Enables protection of CPU state and memory contents owned by a realm
  - Minimum Viable Product
- Final RMM 1.0 spec (EAC5) released in Oct 2023
- TF-A / TF-RMM support upstream since Jan 2024
  - Public Linux / KVM / EDK2 branches available at same time on https://gitlab.arm.com/
- Latest Linux / KVM patches based on v6.9-rc1 on list since Apr 2024
  - kvm-unit-test patches too
- Latest EDK2 patches (realm guest firmware) on list since Apr 2024

Getting started

- Shrinkwrap tool supports CCA configs for latest component tree builds
  - Standard configurations available using public branches, for example cca-3world.yaml
  - Can use overlay configurations
    - for example, to build a custom kernel against the latest stable versions of other components
  - Runs on Base Fixed Virtual Platform (FVP)

- Integrated stack also available in Base FVP and Fremont solution releases
  - Latter runs on Neoverse V3 Infrastructure FVP

- Arm learning paths help accelerate application development with Arm CCA
  - Get Started with RME
  - Create a VM in a realm using Arm CCA
  - Coming soon: Run an application in a realm using Arm CCA

CCA 1.1 features – needed for initial deployments

Further strengthen the security guarantees provided to end users (Realm owners)

- Memory Encryption Contexts (MEC)
  - Physical memory contents of each Realm protected using a unique key or tweak
- Multiple signers
  - Require firmware image to be endorsed by multiple authorities, for example vendor plus a trusted auditor

Enable migration of workloads from non-secure VM to Realm, by providing feature parity

- Planes
  - Multiple privilege levels within a Realm, orthogonal to traditional kernel / user-space split
- Device Assignment (DA)
  - Enable trusted device functions to be admitted into a Realm's TCB, and granted DMA
- Host Debug of Realm
  - In a controlled environment, enable host to debug a realm (bypassing CCA security guarantees)

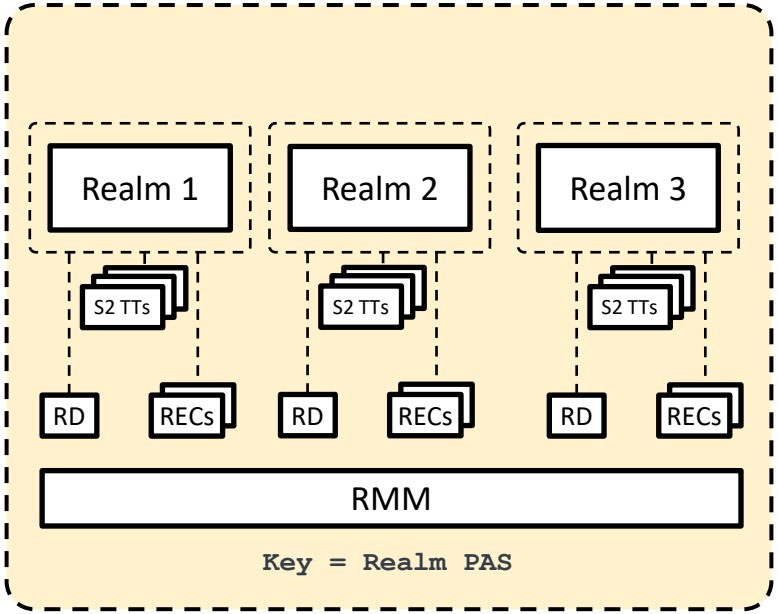
Allow platform owners additional flexibility, in deploying and updating firmware

- Live firmware activation
  - Update firmware image(s) while workloads continue to run, with minimal loss of availability
  - Replace platform firmware (for example, RMM) with an image supplied by the non-secure host

Isolation Boundary

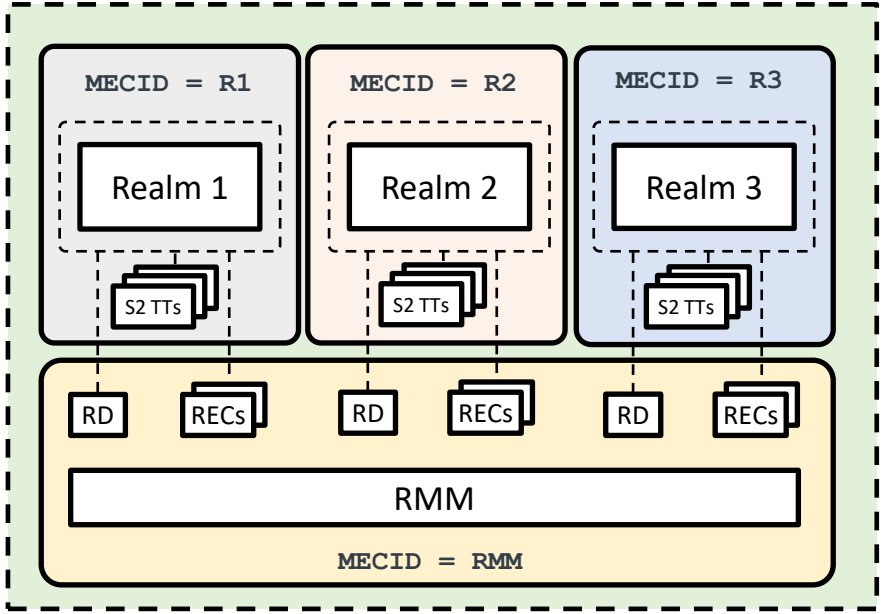
Crypto Boundary

Arm CCA 1.0



- TF-RMM impact: Small
  - Enforce MEC ID uniqueness

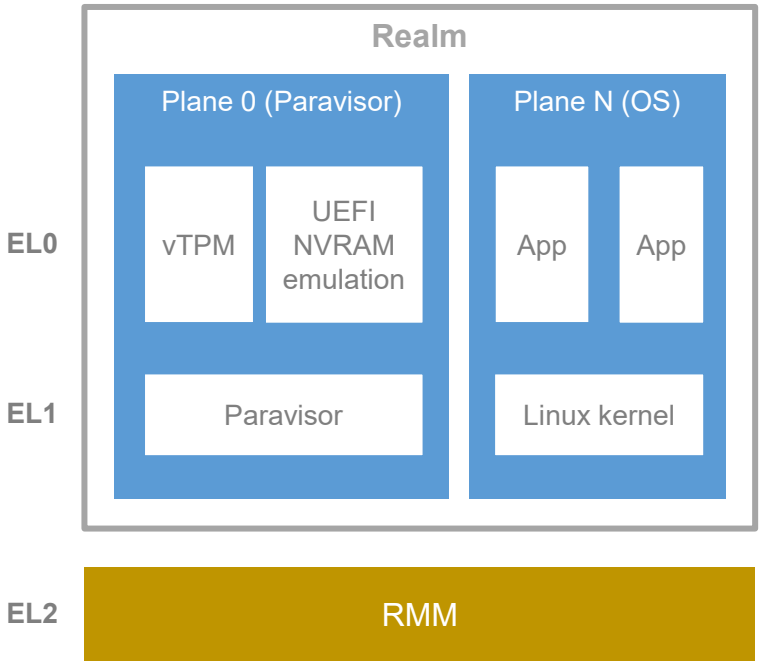
+ MEC



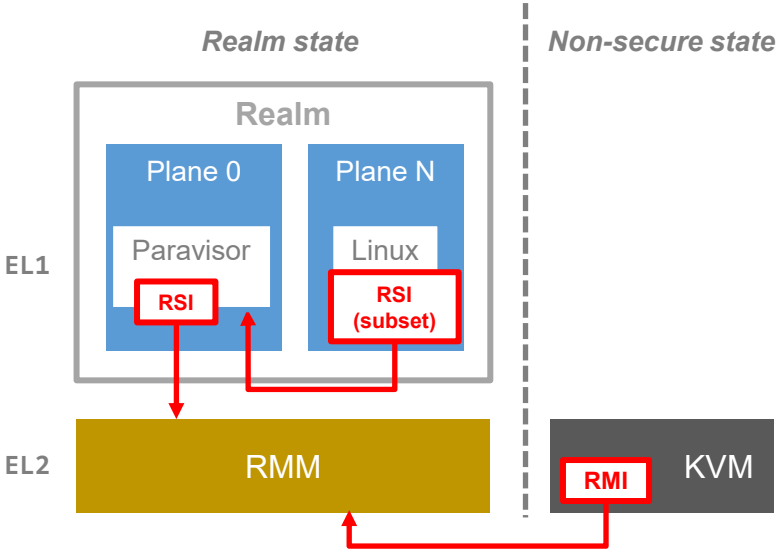
- Realm guest Linux impact: Zero
- Host Linux/KVM impact: X-Small
  - Allocate a MECID for each Realm

Planes requirements

- In addition to the main guest OS and user workload, allow the contents of a Realm to include other software components (Planes)
  - For example, a security service like vTPM
- Provide isolation within a Realm, allowing privilege separation between the Planes
  - All Planes have Same IPA → PA mappings but IPA memory permissions may differ
- Allow the host hypervisor to continue treating the Realm as a single unit, for the purposes of resource allocation, scheduling and migration
- Within the Realm, privileged Plane 0 assigns resources to the other Planes



Planes SW impacts



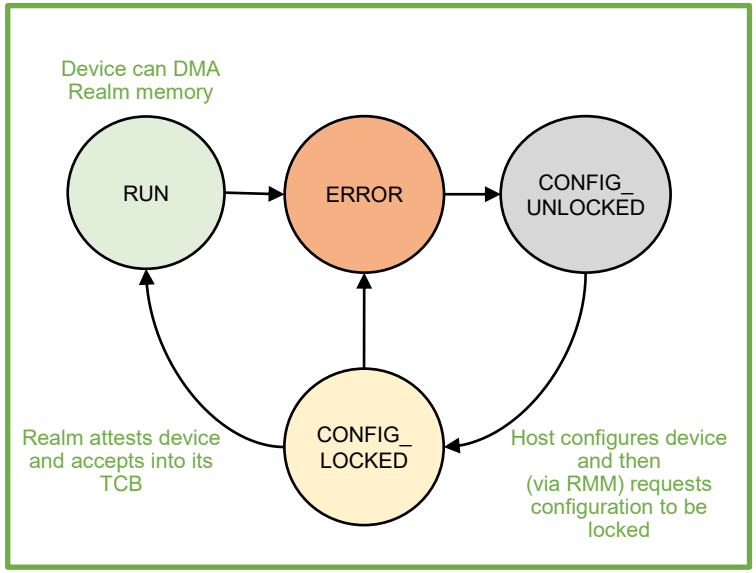
* Need 2 backends to cater for availability of S2PIE/S2POE CPU architecture features

- TF-RMM impact*: Large
  - Validate RMI Realm Translation Table (RTT) commands and create Stage2 page tables
  - Implement Plane entry/exit
- Realm guest Plane 0 impact: Medium -> Large
  - Create/enlighten Paravisor to manage guest Planes (for example, schedule P_N vCPUs, emulate RSI, …)
  - Solutions under investigation - currently out of scope for reference solution
- Realm guest Plane N Linux impact: X-Small
  - Expect Plane N to use (a subset of) RSI to communicate with Plane 0
  - Note, Planes are optional; CCA continues to support (RSI-enlightened) Linux guests without Planes
- Host Linux/KVM impact*: Medium
  - Use RMI to program RTTs per plane
  - Minimal impact on VMM

Device Assignment (DA) requirements

- Allow hypervisor to assign a PCIe TDISP device to a Realm
  - Also support coherently-attached devices, such as CXL instances*
  - Also support on-chip PCIe devices*
- Allow Realm to attest the identity and configuration of the device function
- Device lifecycle guarantees that
  - DMA is blocked until device has been approved by the Realm
  - Any changes in device configuration cause transition to an error state, which revokes DMA
  - Once removed from a Realm, device guarantees that it will scrub confidential state
- Management of device lifecycle must be standards-based
  - RMM must not require any device-specific knowledge
  - However, RMM will require knowledge of platform topology

TDISP (Trusted Device Security Interface Protocol) lifecycle

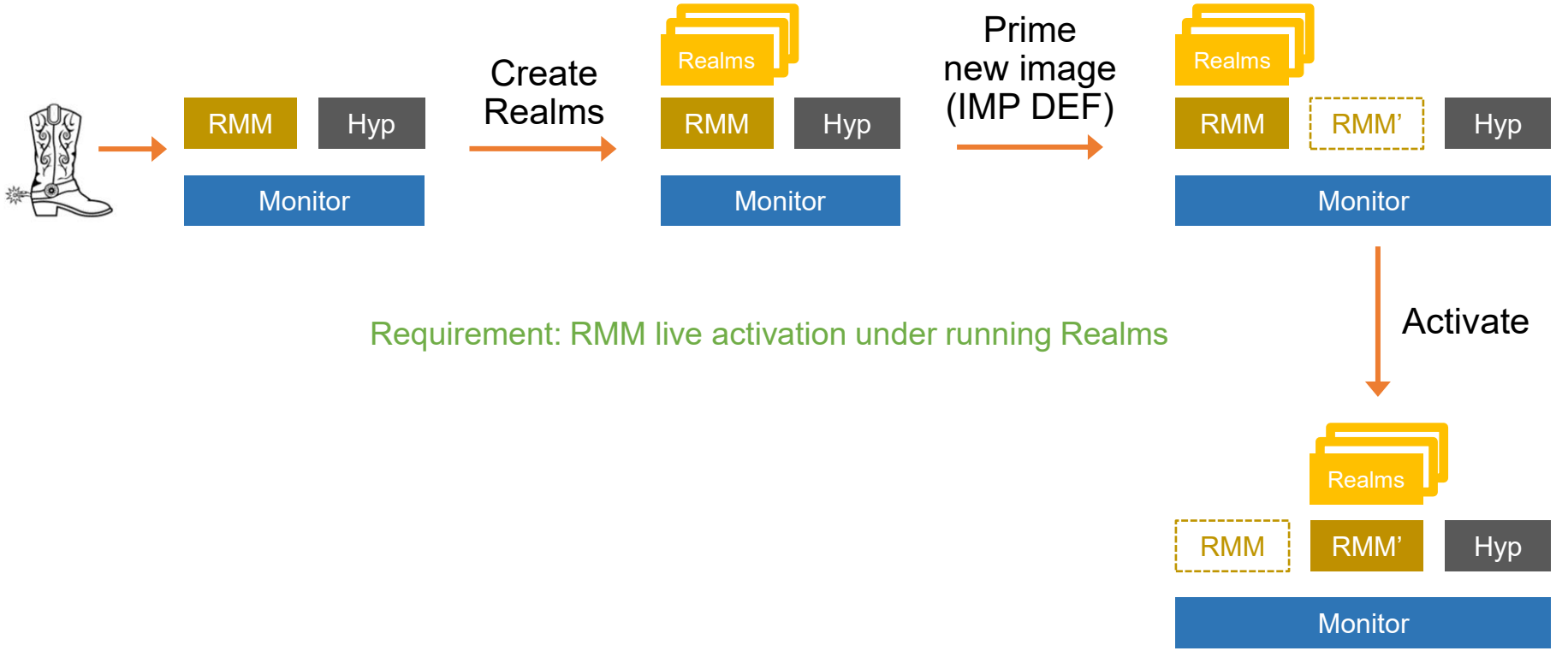


* Note, not supported by any CC architecture yet

DA SW impacts

- Plan to enable foundation PCIe TDISP DA then expand use-cases
- TF-A / TF-RMM impact: Large (foundation) -> X-Large (advanced)
  - Implement DA ABIs and integrate PCIe standard reference libs to enforce device lifecycle
  - SMMU S2 driver + SMMU S1 emulation + PCIe root port programming
- Realm guest Linux impact: Large (foundation) -> X-Large (advanced)
  - Generic PCIe / driver support for trusted devices
    - Irrespective of whether running on Confidential-Compute (CC) or non-CC VMs
  - Community Drive: Linux Foundation CC Consortium (CCC) – Linux Kernel SIG
    - Driven by PCI Dev maintainer
  - Arch specific backends for TDISP will plug in to CCA interface hooks
- Host Linux/KVM impact: Large (foundation) -> X-Large (advanced)
  - DA enlightenment (use new RMIs to control device lifecycle)
  - UABI for VMM to describe DA devices – shared across architectures
  - SMMU stub (for interrupt management) for RMM SMMU Driver

Live firmware activation



Requirement: RMM live activation under running Realms

Live firmware activation

RMM Hyp

Monitor

Prime new image (IMP DEF)

Requirement: RMM installation during hypervisor boot

RMM RMM' Hyp

Monitor

Activate

RMM RMM' Hyp

Monitor

Create Realms

Realms

RMM RMM' Hyp

Monitor

Live firmware activation impacts

- TF-A / TF-RMM impact: Medium -> Large
  - Create staging area for new firmware and transfer live state to new image
  - Live activating an arbitrary firmware version is hard – may need to restrict use-cases initially
    - For example, limit to specific code sections or require new version to be data compatible
    - Will focus on RMM and BL31 (EL3 firmware) live activate initially
    - Live activating the latter is especially hard (for example, may require CPU reset)
  - Can increase use-cases over time (for example, by versioning data structures)
  - Also need hooks to authenticate new firmware and to update firmware measurement log
    - Actual authentication is platform specific
- Realm guest Linux impact: None (hopefully)
- Host Linux/KVM impact: Medium
  - Use new ABIs to provide cycles to prime/activate new firmware
  - May need to quiesce activity and rendezvous CPUs during activation phase

What to expect next

2024

- Continued upstreaming of CCA v1.0 Linux / KVM patches
- Monthly releases of RMM v1.1 spec
  - ALP with early DA / Planes support available now
  - Individual features will reach BET through the year, as they mature
- Collaborative development of CCA v1.1 SW (prototyping in progress)
- Regular public stack drops of CCA v1.1 as spec features reach BET (using Shrinkwrap)

2025

- Final RMM v1.1 spec (EAC)
- Upstreaming of non-DA-related CCA v1.1 features as they mature
  - Much quicker for TF projects than Linux / KVM
- Continued development of CCA v1.1 DA features and start upstreaming foundation support
- Quarterly Arm solution releases of integrated stack with CCA v1.1 features

First Deployments → 2026


LinaroConnect
MADRID 2024 | MAY 12-17 2024
Thank you