Deriving and proving security guarantees of Arm CCA

Ben Fiedler
ETH Zurich
Zurich, Switzerland

David Cock
ETH Zurich
Zurich, Switzerland

Timothy Roscoe
ETH Zurich
Zurich, Switzerland

The Arm Confidential Compute Architecture (CCA) allows running applications in the cloud, while providing strong guarantees about data integrity and confidentiality to lower-privileged software.

CCA-compliant systems create and manage four *worlds*, which form a partition of the system address space: realm, root, secure, and non-secure. Worlds are a *dynamic* partition of the system address space: trusted software running in the *root* world is responsible for safely transferring memory between worlds upon request.

Confidential computing takes place entirely in the realm world, in so-called *realms*. Realms are isolated from each other by the realm management monitor (RMM), which is privileged software running in the realm world. The RMM provides a strict subset of hypervisor functionality to realms, such as pausing/resuming execution and nested translation, leaving resource allocation and scheduling decisions up to non-secure world.

CCA's security guarantees have not been formalized, or even exhaustively described informally. Allowing creation, modification, and destruction of realms at runtime makes precisely expressing the necessary security properties difficult: the security state of such a system changes over time, and has to be correctly maintained even during security state transitions. A formal treatment of CCA's security properties is essential to make precise and justified statements about the actual guarantees provided to realms.

However, modern hardware systems are enormously complicated, and thus difficult to reason about. Many components concurrently interact and interleave, with only few coordination primitives. Thus, these platforms can deliver great performance, at the cost of obscuring the exact state of the underlying system, as in-flight operations may have significant impact on other in-flight operations.

Furthermore, the translation and protection structures in modern systems are self-referential: they alter semantics of memory accesses, but are themselves stored within the same address spaces that they configure and protect. Not only does this apply to the configurations themselves, but the hardware mechanisms that interact with these components also interact with the same system as the software running on top. MMUs issue memory operations which are interleaved with other, regular operations of the system. Aggressive translation and protection caching within the MMU TLBs can also leak stale protection state, and thus have to be carefully managed to ensure no unwanted behaviour is exhibited.

Previous work by Li et al. has verified the implementation of CCA firmware as a mixture of C and assembly code running on relaxed-memory hardware, assuming architectural correctness of the underlying platform. Our work complements the existing approach by focusing not on CCA's architectural description, but on the interactions within the underlying memory subsystem, and how it affects the derivable security properties.

We built a machine model that captures the highly concurrent, distributed nature of modern memory subsystems. Component interactions are modelled via message passing, and messages represent memory operations, such as reads, writes, cache/TLB maintenance, and barriers. Message handlers are expressed in a small-step semantics to allow expressing concurrent interleaving during operation handling.

To tackle the self-referential nature of memory protection and translation structures, we leverage an intermediate representation based on decoding nets. Thus we decouple the semantics of memory protection and translation structures from their hardware representation, separating the low-level semantics of memory operations from their effects on the translation and protection state.

Overall, we model a platform from the perspective of the memory subsystem, considering the interactions and interleaving of hardware page table walkers and coherent caches with software running on top. This way we can even account for the effects that potential architectural or implementation bugs have on the platform, which have been shown to occur in the past.

We present early stage work, done in collaboration with and funded by Arm. Currently, we are deriving the security properties provided by Arm CCA's hardware memory protection extensions and proving they are preserved when moving memory between worlds. After establishing security guarantees on the realm address space, we can turn towards verifying the behaviour of the RMM and its interactions with the non-secure world in safely creating, executing, and tearing down realms. Once we have covered the behaviour of CCA-compliant cores, we can move on to tackle the behaviour of DMA devices, and how they could be safely assigned to individual realms. Ultimately, we hope extend our formal modelling to derive and prove guarantees for a whole platform, potentially integrating with other (formal) hardware modelling efforts.

Eurosys'23, May 8–12 May 2023, Rome, Italy
2023.