FlexCap: Software Compartmentalisation Trade-Offs with Hardware Capabilities

John Alistair Kressel, Hugo Lefeuvre, Pierre Olivier – *The University of Manchester*

The University of Manchester · Innovate UK · Engineering and Physical Sciences Research Council · Digital Security by Design

Motivation & Background

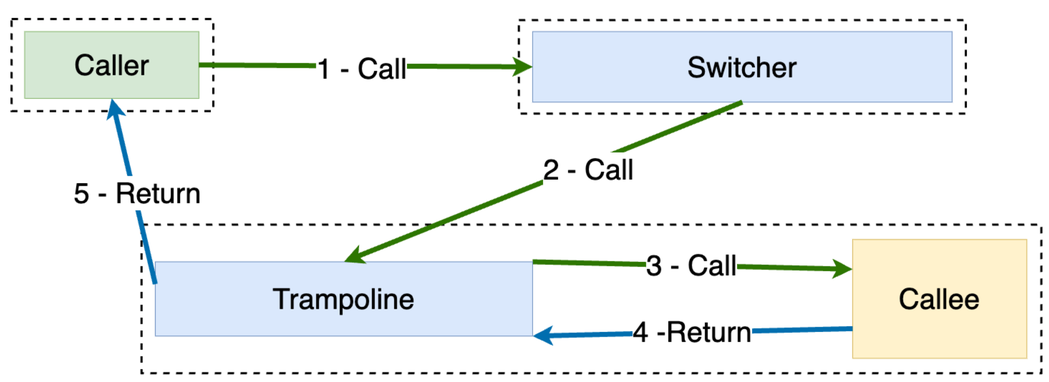
- CHERI brings hardware capabilities to RISC ISAs
- CHERI hardware capabilities add bounds and permissions information to pointers
- CHERI can be used in **hybrid execution mode** – both pointers and capabilities used
- Hybrid mode enables existing software to be used **without major porting**
- How easy is it to apply to compartmentalisation (the isolation of code and data) in hybrid mode to a single address space scenario such as applications?

Problem Statement:

Exploration of intra-address space, hybrid mode compartmentalisation is lacking - we explore the possible compartment models and their performance and engineering costs as well as security and scalability

System Design

- Prototype implemented using compartmentalisation-aware unikernel – FlexOS
  - All components in single address space
  - Unikernels have no memory isolation for performance – FlexOS uses compartmentalisation to restore security
- Compartments defined statically by developer
  - Compartments defined by *PCC* and *DDC* – global architectural capabilities
- Function call 'gates' initiate compartment switches (**Caller**)
- Privileged **switcher** switches compartment capabilities and stacks
- Trampoline is entry point to **callee** and performs capability return to **caller**

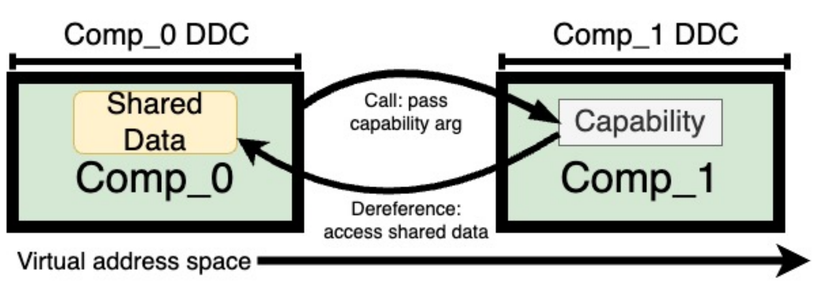


Challenges – Data Sharing

Mixing pointers and capabilities is an engineering burden – two approaches proposed to reduce effort

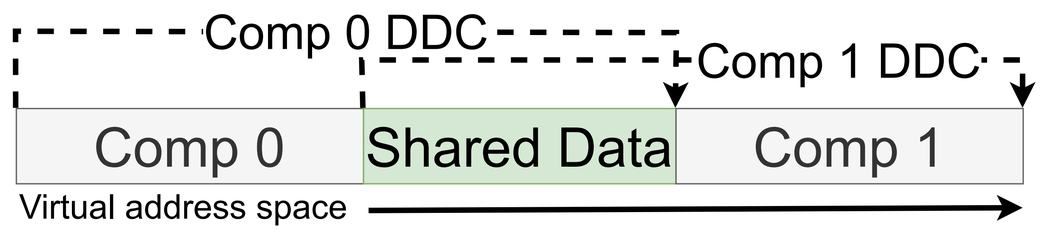
Approach 1: Manual Capability Propagation

- Functions ported to use capabilities
- Effort low in small compartments
- Tightly bounded data accesses
- **Trust model**: sandbox



Approach 2: Overlapping Shared Memory

- Region of shared memory between pairs of compartments
- Compartment data bounds (DDC) extended to overlap shared memory
- Data annotated by developer to relocate to shared data
- Coarse-grained data sharing
- **Trust model**: mutual distrust

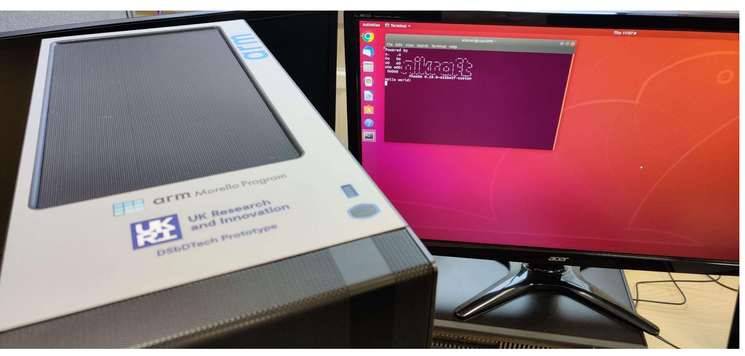


Evaluation On Morello Hardware

Libsodium benchmark used to evaluate **Approach 1** (5 functions), SQLite benchmark used to evaluate **Approach 2** (filesystem isolated) **– All evaluation on bare metal**

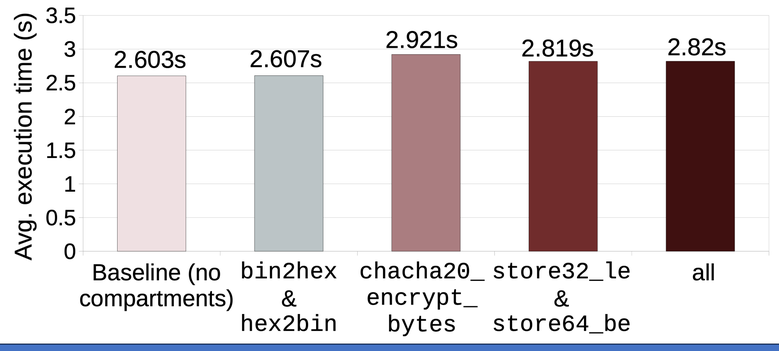
FlexOS on Morello

- FlexOS Unikernel ported to Morello
  - Runs in hybrid mode
  - 2200 LoC needed for port
- Execution bare metal on hardware
  - Majority of existing work evaluated on FPGAs or softcores



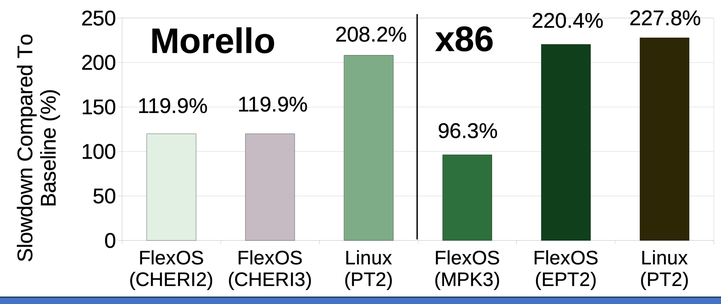
Approach 1: Manual Capability Propagation

- **Performance overhead:**
  - 0.1%-12.2% relative to uncompartmentalised execution on Morello
- **Engineering cost:**
  - max 2 hours and 73 LoC changed (>50%)



Approach 2: Overlapping Shared Memory

- **Performance overhead:**
  - 119.9% relative to uncompartmentalised execution on Morello
  - Comparable to MPK & outperforms EPT compartmentalisation in FlexOS on Intel x86-64
  - Outperforms Linux with SQLite benchmark and similar isolation (filesystem)
- **Engineering cost:**
  - max 2 days and <300 LoC changed in filesystem (~5%)



Conclusions & Future Work

- Performance overhead of hybrid mode compartmentalisation in a single address space is comparable to other intra-address space mechanisms
- Engineering burden of mixing pointers and capabilities requires trade-offs to design to reduce it
- Pure capability compartments within a Unikernel environment will be explored in the future

John Alistair Kressel, Hugo Lefeuvre, and Pierre Olivier. 2023. Software Compartmentalization Trade-Offs with Hardware Capabilities. In Proceedings of the 12th Workshop on Programming Languages and Operating Systems (PLOS '23)

Project Website: https://olivierpierre.github.io/project-flexcap/