Arm Morello Evaluation Platform - Validating CHERI-based Security in a High-performance System

Richard Grisenthwaite
SVP Chief Architect and Fellow, Arm
Richard.Grisenthwaite@arm.com
Acknowledgements

This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237 (“CTSRD”), with additional support from FA8750-11-C-0249 (“MRC2”), HR0011-18-C-0016 (“ECATS”), and FA8650-18-C-7809 (“CIFV”) as part of the DARPA CRASH, MRC, and SSITH research programs. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

This work was supported in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694.

We also acknowledge the EPSRC REMS Programme Grant (EP/K008528/1), the ERC ELVER Advanced Grant (789108), the Isaac Newton Trust, the UK Higher Education Innovation Fund (HEIF), Thales E-Security, Microsoft Research Cambridge, Arm Limited, Google, Google DeepMind, HP Enterprise, and the Gates Cambridge Trust.

Security is the greatest challenge computing needs to address to meet its full potential
Memory (Un)safety issues remain major source of CVEs

• Matt Miller (BlueHat 2019) : Microsoft around 70% of CVEs are memory unsafety issues
  • #1 Heap out-of-bounds
  • #2 use-after-free
  • #3 type confusion
  • #4 uninitialized use

• Chromium reports similar issues:
  • Memory safety - The Chromium Projects
    “70% of our serious security bugs are memory safety problems.”

• Been around for a very long time
  • Morris Worm 1988 usually credited as the first buffer overflow attack on the internet
  • C/C++ is not going away any time soon in the world’s software
CHERI architecture in one slide

- CPU architecture adds 128-bit “capabilities” in the register file plus a tag bit
  - Capability contains the address, bounds information, permission information etc
  - The tag bit is metadata that distinguishes a capability from normal data
    - The tag bit prevents “forging” of a capability
    - This functionality gives strong provenance of capabilities
  - Architecture has the ability to “seal” capabilities as well as part of compartmentalisation

- Loads/stores using capabilities as addresses are checked to be legal
  - Within address range and matching the supplied permissions

- Data processing on capabilities has rules to limit operations
  - Bounds cannot be arbitrarily increased, permissions cannot be relaxed etc

- PC is converted to a capability called the PCC to place bounds on the PC
  - Direct Branches will be within the PCC;
  - indirect branches (including returns) can change PCC

- Capability is used in place of a normal pointer in some or all situations
  - Exactly how when this happens is part of the software usage case
Two key applications of the CHERI primitives

1. **Efficient, fine-grained memory protection for C/C++**
   - Strong source-level compatibility, but requires recompilation and minor source-code changes
   - Deterministic and secret-free referential, spatial, and temporal memory safety
   - Retrospective studies estimate $\frac{2}{3}$ of memory-safety vulnerabilities mitigated
   - Generally modest overhead (0%-5%, some pointer-dense workloads higher)

2. **Scalable software compartmentalization**
   - Multiple software operational models from objects to processes
   - Increases exploit chain length: Attackers must find and exploit more vulnerabilities
   - Orders-of-magnitude performance improvement over MMU-based techniques
     (<90% reduction in IPC overhead in early FPGA-based benchmarks)
Microsoft Security Research Center (MSRC) study analyzed all 2019 Microsoft critical memory-safety security vulnerabilities

- Metric: “Poses a risk to customers → requires a software update”
- Vulnerability mitigated if no security update required

Blog post and 42-page report
- Concrete vulnerability analysis for spatial safety
- Abstract analysis of the impact of temporal safety
- Red teaming of specific artifacts to gain experience

CHERI, “in its current state, and combined with other mitigations, it would have deterministically mitigated at least two thirds of all those issues”

https://msrc-blog.microsoft.com/2020/10/14/security-analysis-of-cheri-isa/
Morello Prototype system: What Has Arm Produced?

https://www.arm.com/morello

- Morello prototype architecture
- Morello Platform Model (FVP) — a software model of the Morello platform
- Linaro hosted OSS
- Memory Model Tools
- Toolchain
- Partner Forum including FAQ
- Technical reference manual
- Morello test chip and board
- Morello Overview Guide
- Morello Development Platform and Software Stack User Guide
- Future – how-to video
TSMC N7 Process

2.5GHZ CPU

109.9mm²
Extending Structures and Memory to support capabilities

- Increase register file to support 129-bits
  - Area, power, and other register file optimizations need to be considered
  - Could be implemented as separate register file or unified register file

- Requires additional storage at all levels of memory hierarchy (1-bit per 16B of data)
  - Includes caches, buffers, and other microarchitecture structures
  - May widen existing structure or store in separate structures

- System buses need to transport tag information
  - Use existing signals where possible to decrease protocol changes

- Forwarding networks and internal data buses may need to increase

- Decode complexity and area (new instructions, modes, system registers)
  - Strains on decode space availability may require extra execution units or other changes
Memory checks and Load Store extensions

+ Address generation usually a critical path in load store designs
  - Compartmentalizing legacy code may add an offset to address generation
  - Capabilities require new bounds checks on those addresses

+ New faults need to be detected and reported to control and track capabilities in a system
  - For protection (compartmentalization) or performance (revocation)
  - Adds dependency between stored capability and the location to which being stored that did not exist before (which may cause delays if implementation stalls address until data available)
  - MMU Access faults and PTE updates (capability write permission/dirty bit) dependent on store tag

+ Fault Address Register (FAR) captures full address for all faults (including late detected, precise, data dependent faults)
  - May require additional storage to propagate full address throughout pipeline

+ Capability instruction implementation must maintain atomicity
  - Makes cracking instructions more difficult, especially for atomic instructions such as CAS, ST{L}XP
Extending data processing - Bounds Checking

- Upper and Lower Bounds information is compressed into 64 bits
- When bounds checks are needed, this has to be decompressed
  - Needed for all loads/store operations and branches – done in parallel with address generation
  - Also needed to cover advanced capability operations in the integer unit
Extending data processing - GetBounds logic

- Decoding of the compressed CHERI format for bounds
  - One bounds check requires two shifters, one adder, two short comparators (for TopAdj) and one wide comparator (to compare decompressed Bound against address)
Extending the data processing - Representability

Representability checks needed to make sure capability modifications are valid

- Representability is an artifact of compressing two 64-bit bounds and a 64-bit pointer into 128 bits
- This affects arithmetic operations since taking the pointer too far out of bounds isn't representable
- Representability is designed to be fast for the common case (add/subtract) but other cases require full decompression (absolute value)
Instruction fetch and control flow prediction for capabilities

- PC is extended to 129-bits (Program Counter Capability - PCC)
- Branches need representable checks, bounds checks, and new fault handling
  - Includes new and legacy branches
  - Indirect legacy branches need expensive representable check that may add cycles
- How to handle indirect branches to a capability (might change PCC):
  - Extend branch predictor to hold PCC (simple 1-bit direction prediction or more complex and larger predictions)
  - Statically predict PCC does not change and take branch mispredict penalty if wrong
  - Stall PCC dependencies until PCC known
- Stalling PCC option reduces concerns in modifying/growing existing branch unit but introduces performance costs in delaying instructions dependent on new PCC
Holding the CHERI capability validity tag in the memory

- Considerations should be made for tags in DRAM in terms of area and performance (similar to MTE)
- Add tags as separate carve out space in DRAM or as part of data within (such as ECC)
- Carve out space for tags reduces effective size of DRAM
- Increased latency to read both tags and data (two accesses)
  - Can be mitigated with other structures (tag caches, compression caches, etc.)
- If using existing ECC bits for tag may decrease resolution of single bit errors
Demonstration Morello open-source desktop software stack

- **CHERI / Morello LLVM toolchains**
  - Implements sub-language and C/C++ language pointers as capabilities

- **CheriBSD adapted version of FreeBSD**
  - Complete open-source operating system
  - Memory-safe kernel and userspace
  - Supports legacy aarch64 binaries
  - Multiple compartmentalization models, including fast IPC and sandboxed libraries

- **X11, KDE-based desktop environment**
  - 6MLoC adapted by 1 FTE in 3 months
  - 0.026%LoC changes for memory safety
  - 73.8% assessed vulnerability mitigation rate
Formal proving of security properties of Morello

- Fundamental security property of CHERI architectures is Reachable Capability Monotonicity:
  - normal code execution, of arbitrary code, cannot increase the set of available capabilities
- Conventional testing cannot provide high assurance - but formal methods can prove it holds in general
  - Applied at the architecture specification language level
  - Arm ASL (62K lines of code) -> SAIL -> Isabelle Proof Assistant
  - 3 security issues found before tapeout
- SAIL ISA specification also used for ISA test generation
  - Stressing capability corner cases based on strong ISA understanding
- Machine-checked mathematical proofs of whole-ISA security properties of a full-scale industry ISA

What Feedback Do We Want To Get from Morello?

- Answers to the performance questions for a wide range of different usage models
- Compelling examples of Capabilities offering security/performance improvements
  - Backed up by “Red-teams” having attacked the system and demonstrated security of the system
  - Compelling in comparison with existing deployed state of the art approaches
- Better Understanding of
  - Different languages and run-times can use capabilities, not only C and C++, but also Javascript, Java
  - Fine-grained compartmentalisation can be used
  - Answers to the performance questions for a wide range of different usage models
- A showcase to encourage other architectures to adopt capabilities
- Experience of what the right SoC hardware is for building capabilities
- An architectural approach with formally proven security properties

What to put into the future Arm architecture for an industrial deployment
Questions?
Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
ありがとう
Asante
Merci
감사합니다
धन्यवाद
شكرًا
ধন্যবাদ
תודה