CTSRD

CRASH-worthy Trustworthy Systems Research and Development

Technology Transition for CHERI Opportunities for Research

Robert N. M. Watson, Simon W. Moore, Peter Sewell, Peter G. Neumann

Hesham Almatary, Jonathan Anderson, John Baldwin, Hadrien Barrel, Ruslan Bukin, David Chisnall, James Clarke, Nirav Dave, Brooks Davis, Lawrence Esswood, Nathaniel W. Filardo, Khilan Gudka, Alexandre Joannou, Robert Kovacsics, Ben Laurie, A. Theo Markettos, J. Edward Maste, Alfredo Mazzinghi, Alan Mujumdar, Prashanth Mundkur, Steven J. Murdoch, Edward Napierala, Robert Norton-Wright, Philip Paeps, Lucian Paul-Trifu, Alex Richardson, Michael Roe, Colin Rothwell, Peter Rugg, Hassen Saidi, Peter Sewell, Stacey Son, Domagoj Stolfa, Andrew Turner, Munraj Vadera, Jonathan Woodruff, Hongyan Xia, and Bjoern A. Zeeb

University of Cambridge and SRI International

VeTTS Verified Software Workshop – 24 September 2019

Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

SRI International

UNIVERSITY OF CAMBRIDGE

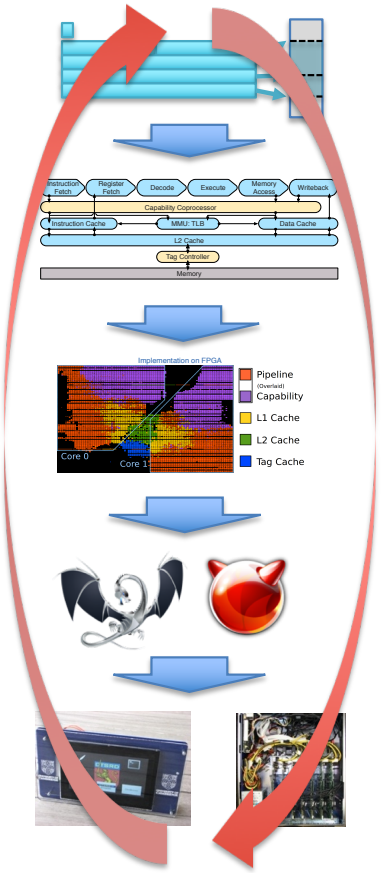
Introduction

- A very brief intro to the CHERI architecture and its use cases
- To learn more about the CHERI architecture and prototypes:

  http://www.cheri-cpu.org/

- Watson, et al. **Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 7)**, Technical Report UCAM-CL-TR-927, Computer Laboratory, June 2019.
- Watson, et al. **Introduction to CHERI**, Technical Report UCAM-CL-TR-941, Computer Laboratory, September 2019.

Hardware-software co-design over 9 years



- SRI + Cambridge over three DARPA programs (~$26M), EPSRC REMS, (£5.6M) Industrial: Google / DeepMind / Arm / HPE / … (~£750K)
- Architectural mitigation for C/C++ TCB vulnerabilities
  - Tagged memory, capability pointer representation
  - Fine-grained pointer and memory protection
  - Highly scalable software compartmentalization
  - Hybrid capability system for incremental adoption
- Least-privilege, capability-oriented design mitigates many known (and unknown future) classes of vulnerabilities + exploit techniques
- Hardware-software-model co-design + concrete prototyping:
  - CHERI model, CHERI-MIPS, CHERI-RISC-V, CHERI-ARM concrete ISAs
  - Formal ISA models, Qemu-CHERI, FPGA prototypes
  - CHERI Clang/LLVM/LLD, CheriBSD, C/C++-language applications
  - Repeated iteration to improve {performance, security, compatibility, ..}
- **New work: CHERI portability – ARMv8-A (w/Arm) and RISC-V**

SRI International

UNIVERSITY OF CAMBRIDGE

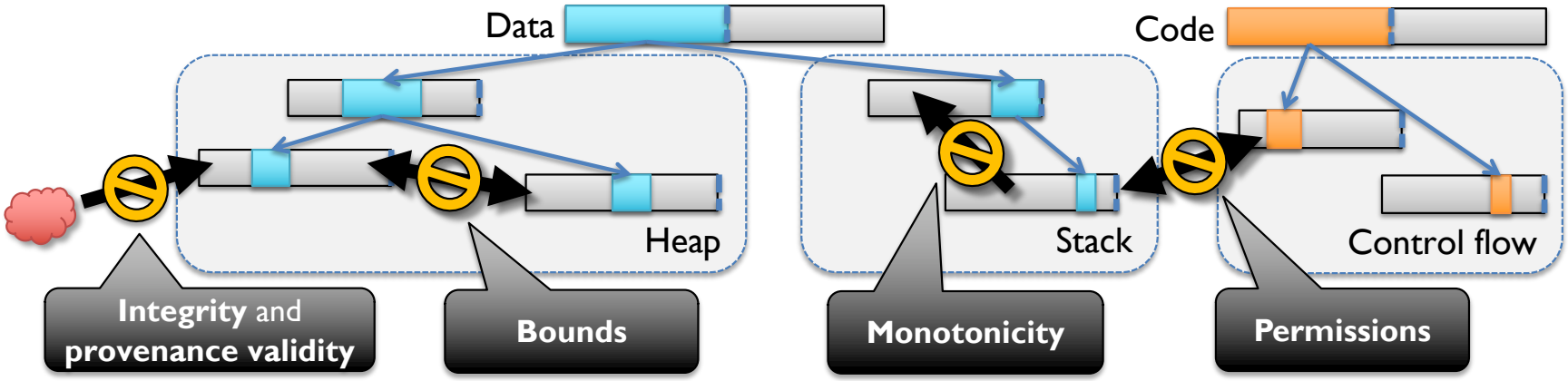
CHERI design goals and approach (1)

- **Architectural security** to mitigate **C/C++ TCB vulnerabilities**
  - Efficient primitives allow software to ubiquitously employ the **principle of least privilege** and **principle of intentional use**
- **De-conflate virtualization and protection**
  - Memory Management Units (MMUs) protect by **location** in memory
  - CHERI protects **references (pointers)** to code, data, objects
  - Capabilities can also be used to describe **scalable isolated compartments with efficient sharing** within address spaces
  - Capabilities add protection properties to **existing indirection** (pointers), avoiding adding new architectural table lookups

CHERI design goals and approach (2)

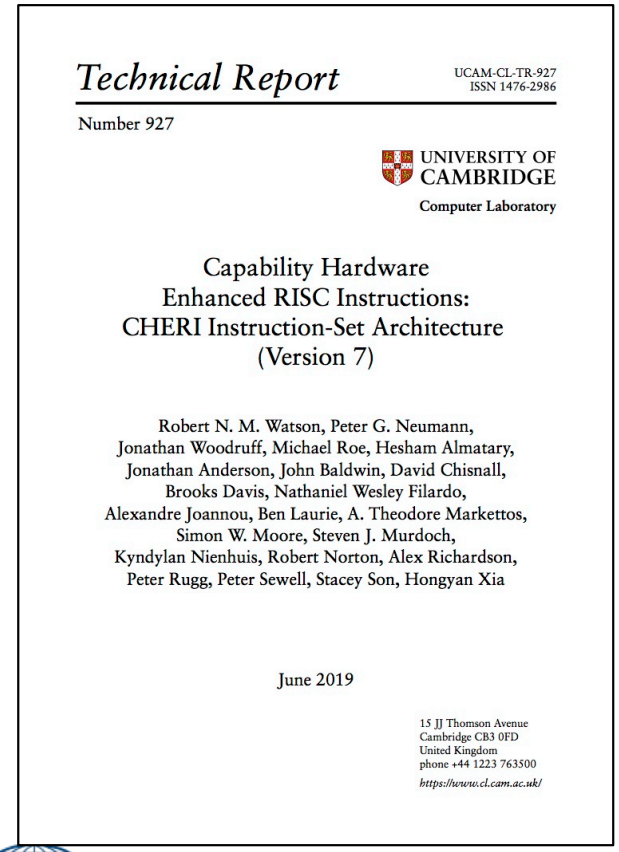
- **Hybrid capability architecture**
  - Model **composes naturally** with RISC ISAs, MMUs, MMU-based systems software, C/C++ languages
  - Capabilities protect resources **within virtual address spaces**
  - Supports **incremental software deployment paths**
- **Architectural mechanism** can enforce various **software policies**
  - **Language-based properties** – e.g., referential, spatial, and temporal integrity (e.g., C/C++ compiler, linkers, OS model, runtime)
  - **New software abstractions** – e.g., software compartmentalization (e.g., confined objects for in-address-space isolation)
- **Portable protection model? MIPS, RISC-V, ARMv8, …**

CHERI enforces protection semantics for pointers



- **Integrity** and **provenance validity** ensure that valid pointers are derived from other valid pointers via valid transformations; **invalid pointers cannot be used**
  - E.g., Received network data cannot be interpreted as a code or data pointer
- **Bounds** prevent pointers from being manipulated to access the wrong object
  - Bounds can be minimized by software – e.g., stack allocator, heap allocator, linker
- **Monotonicity** prevents pointer privilege escalation – e.g., broadening bounds
- **Permissions** limit unintended use of pointers; e.g., W^X for pointers
- These primitives not only allow us to implement **strong memory protection**, but also higher-level policies such as **scalable software compartmentalization**

Recent CHERI evolution



Technical Report

UCAM-CL-TR-927
ISSN 1476-2986

Number 927

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 7)

Robert N. M. Watson, Peter G. Neumann, Jonathan Woodruff, Michael Roe, Hesham Almatary, Jonathan Anderson, John Baldwin, David Chisnall, Brooks Davis, Nathaniel Wesley Filardo, Alexandre Joannou, Ben Laurie, A. Theodore Markettos, Simon W. Moore, Steven J. Murdoch, Kyndylan Nienhuis, Robert Norton, Alex Richardson, Peter Rugg, Peter Sewell, Stacey Son, Hongyan Xia

June 2019

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500
https://www.cl.cam.ac.uk/

- First release of CHERI ISA specification technical report in two years
- Key features:
  - Architecture-neutral CHERI model
  - Elaborated CHERI-RISC-V ISA
  - CHERI Concentrate capability compression model (**TCS 2019**)
  - Side-channel resistance features
  - Improved C-language compatibility, dynamic linkage, performance optimizations (**ASPLOS 2019**)
  - Experimental features including 64-bit capabilities for 32-bit architectures (**ICCD 2018**), temporal safety (**MICRO 2019**)
  - All instruction pseudocode derived from Sail formal models

Looking Beyond CHERI-MIPS

- 64-bit MIPS for pragmatic reason: needed a 64-bit RISC ISA in late 2010
- Reason for hope: portable virtual-memory semantics and UNIX process model despite (quite) different MMUs across architectures
  - **Architectural abstraction**: Lift CHERI properties above ISA – E.g., tagged capabilities → **architectural-neutral specification**
  - **Architectural localization**: E.g., ISA choices, opcode approaches, exceptions, page tables, … → **architecture-specific specifications**
- Maintain essential CHERI design choices (capabilities, tagged memory, …)
- Validate through full architectural instantiations and software stacks
- Review instantiation choices based on gained experience, evaluation
- Broaden research agenda – managed languages, non-volatile memory, semantics, …

CHERI target architectures

Architecture	Features	CHERI challenges
64-bit MIPS	1990s RISC architecture (CHERI baseline)	Poor code density and addressing modes: harder to differentiate 'essential' CHERI costs; few transition opportunities with MIPS
64-bit ARMv8-A	Mature and widely deployed load-store architecture	Feature-rich; exception-adverse; rich address modes; constrained opcode space; hardware page tables; virtualization features; ecosystem
32-bit and 64-bit RISC-V	Open RISC ISA in active development (MIPS + 10 years?)	Limited addressing modes (expects micro-op fusion); hardware page tables; only partially standardized; features missing (e.g., hypervisor); immature software stack

CHERI-ARM and CHERI-RISC-V

- We are now pursing two DARPA-supported CHERI transition projects:
  - Joint with Arm, an experimental adaptation of 64-bit ARMv8-A to implement the CHERI protection model (since 2014)
  - An experimental adaptation of 32/64-bit RISC-V to implement CHERI protection model (since 2017)
- Complete elaborations of the full hardware-software stack:
  - All aspects of the architectures (e.g., CHERI w/hypervisors, etc.)
  - Formal models, hardware implementations, compilers, OSes, etc.
- Potential for industrial transition through both paths

CHERI ISA comparison

Base ISA	Status	Width	Register file	On fail?	MMU	Cap mode	DDC reloc.	Multibit ASR
MIPS	Full	64-bit 128-bit 256-bit	Split	Exception	SW TLB	No	Yes	No
RISC-V	Experimental	64-bit 128-bit	Both	Exception	HW PT	Yes	Yes	Yes
ARM-A	Experimental	128-bit	Merged	Clear tag	HW PT	Yes	No	Yes
x86-64	Sketch	128-bit	Merged	TBD	HW PT	TBD	Yes	TBD

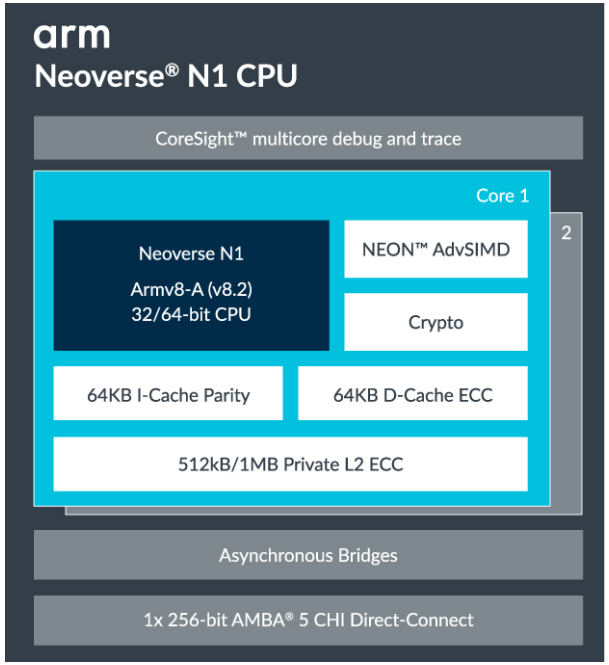
- While software-facing CHERI semantics may be identical, integration with the baseline ISA may differ substantially based on underlying architecture design

CHERI: Portability implications for software

- **CHERI Clang/LLVM**
  - Modest pointer/capability abstraction improvements in front-end, IR
  - Adapt target back-ends to teach them about capability code generation
  - Optimize for architecture-specific code generation
  - Optimize for available microarchitectures
- **CheriBSD**
  - More clear machine-independent / machine-independent split
  - Shift to hybrid capability C in the kernel to improve machine independence
  - Various MD kernel updates: boot code, exceptions, PMAP, …
  - Clean up APIs, header separation, architecture abstraction
  - Various userspace updates: rtld, libcheri, CRT/CSU, …

ISCF: Digital Security by Design (UKRI)

- 5-year **Digital Security by Design** UKRI program: £70M UK gov. funding, £117M UK industrial match, to create CHERI-ARM demonstrator SoC + board with proven ISA
- Leap supply-chain gap that makes adopting new architecture difficult – in particular, validation of concepts in microarchitecture, architecture, and software "at scale"
- Support industrial and academic R&D (EPSRC, InnovateUK)
- Baseline CPU selected; reuses existing SoC/board designs
- Ongoing collaboration reviewing and distilling {essential, desirable, experimental} CHERI features for use in SoC
- Science designed allowed: Support multiple architectural design choices for software-based evaluation once fabricated
- 2020 emulation models; 2021 "Morello" board delivery



ISCF dual-stack software strategy

Open-source application suite

CheriBSD/Arran (DARPA)	**Android** (Arm)
• FreeBSD kernel + userspace • Kernel spatial and referential memory protection • Userspace spatial, referential, and temporal memory protection • Co-process and intra-process compartmentalization	• Linux kernel + Android userspace • Userspace spatial and referential memory protection • Android Runtime Environment (ART)

CHERI Clang/LLVM compiler suite, LLD, LLDB, GDB

CHERI-extended Google Hafnium hypervisor

- Dual-stack strategy: pushes each software stack as far as it can go in the available timeline, provides both BSD- and GPL-licensed variants, and addresses distinct industrial transition opportunities and use cases
  - Android – Mobile devices (Google and other Android-based phones and tablets, etc.)
  - FreeBSD – Embedded/server devices (iOS, Junos, Playstation, Netapp, etc.)

Potential areas for CHERI research

- Quantitative ISA optimization
- Compiler semantics and optimization
- Superscalar microarchitectures
- Tag tables vs. native DRAM tags
- Toolchain: linker, debugger, …
- C++ compilation to CHERI
- Growing the software corpus
- CHERI and ISO C/POSIX APIs
- Compartmentalization frameworks
- MMU-free CHERI microkernel
- Safe Foreign Function Interfaces (FFIs)
- Safe inter-language interoperability
- C-language temporal memory safety
- Integration with managed languages
- Formal proofs of ISA properties
- Formal proofs of software properties
- Verified hardware implementations
- Use with large or non-volatile memory
- Security analysis and red teaming
- Microarchitectural optimization opportunities from exposed software semantics
- MMU-free HW designs for “IoT”

Invitation to collaborate

- It is an exciting moment for CHERI
  - Finally allowed to talk about significant industrial collaboration
  - Experimental adaptations to mainstream architecture(s)
  - Rich and maturing software baselines for experimentation
  - Strong formal foundations available to be built on
  - New research funding opportunities (especially for the UK) create opportunity to broaden collaborations

www.cheri-cpu.org