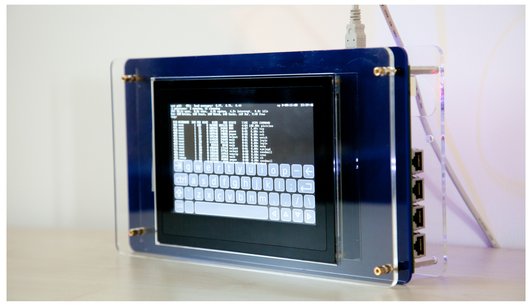
CHERI

Capability Hardware Enhanced RISC Instructions (CHERI) - Architectural Support for Fine-Grained Memory Protection and Scalability Compartmentalization

PIs: Robert N. M. Watson (University of Cambridge), Simon W. Moore (University of Cambridge), and Peter G. Neumann (SRI International)
Jonathan Anderson, John Baldwin, Hadrien Barrel, Ruslan Bukin, David Chisnall, Nirav Dave, Brooks Davis, Lawrence Esswood, 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, Hassen Saidi, Peter Sewell, Stacey Son, Domagoj Stolfa, Andrew Turner, Munraj Vadera, Jonathan Woodruff, Hongyan Xia, and Bjoern A. Zeeb



Capability Hardware Enhanced RISC Instructions (CHERI) extend contemporary 64-bit RISC architectures with a new hardware type, the **architectural capability**, used to represent and protect hardware- and software-defined pointers. CHERI supports the granular implementation of the **principles of least privilege and intentional use**, which naturally mitigate vulnerabilities by limiting the rights gained (and further attack surfaces reachable) by attackers. CHERI is a **hybrid capability model** that cleanly composes with RISC ISAs, Virtual Memory implemented using Memory Management Units (MMUs), MMU-based general-purpose OS designs such as UNIX, and the C and C++ programming languages, supporting incremental deployment of the approach within current hardware-software ecosystems.

CHERI's **fine-grained memory protection** utilizes **capability registers** and **tagged memory** to *(a)* protect **pointers** through hardware-supported pointer integrity, provenance validity, and monotonicity that constrain manipulation, and *(b)* protect **pointee code and data** through fine-grained bounds and permissions that control use. Collectively, these features mitigate many common vulnerability types and memory-based exploit techniques in C- and C++-language software Trusted Computing Bases (TCBs) such as OSes, server applications, language runtimes, and web browsers — including buffer overflows, integer-overflow attacks on pointers, format-string vulnerabilities, Return-Oriented Programming (ROP), Jump-Oriented Programming (JOP), "Stack Clash", and many other pointer-/memory-based attacks.

CHERI's **scalable software compartmentalization** is grounded in **software-defined sealed pointers**, which, combined with its pointer and pointee protection, allow MMU-based processes to be sub-divided into many isolated (but closely coupled) compartments with much greater scalability than MMUs support. CHERI compartment-switching and memory sharing costs are comparable to a function call rather than Inter-Process Communication (IPC). CHERI's compartmantalization performance facilitates more granular software sandboxing to mitigate attacks in a vulnerability- and exploit-technique-independent manner — **defending against future, as-yet undiscovered attack techniques**.

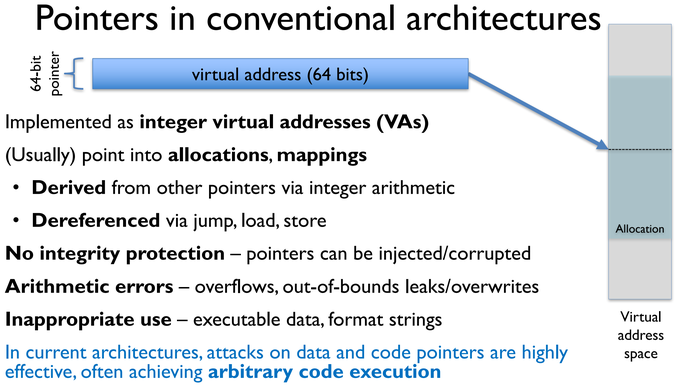
We have developed CHERI over 8 years of **hardware-software co-design** at Cambridge and SRI that has been supported by DARPA, and also by Google, HPE, ARM, EPSRC, and other sponsors. We have implemented formal models of the ISA that enable automated reasoning about CHERI's security properties, a fast ISA-level emulation in Qemu, and a pipelined, multicore FPGA processor design to explore microarchitectural impacts. CHERI's hybrid capability model has allowed us to adapt the **Clang/LLVM compiler** to utilize capabilities, and explore how a lightly modified version of the **FreeBSD OS** and its **open-source application stack** can utilise architectural memory protection and scalable compartmentalization.

We have published about various aspects of CHERI in ISCA, ICCD, ASPLOS, ACM CCS, IEEE SSP, PLDI, and IEEE Micro. Our most recent research has developed an 128-bit in-memory capability representation and efficient tagged memory, and has explored the impact of strong pointer and memory protection on a full UNIX software-stack implementation, demonstrating strong vulnerability mitigation, good source-level compatibility, and low overhead.

Learn about the open-source CHERI architecture, hardware, and software on our website:
http://www.cheri-cpu.org/

17 October 2017

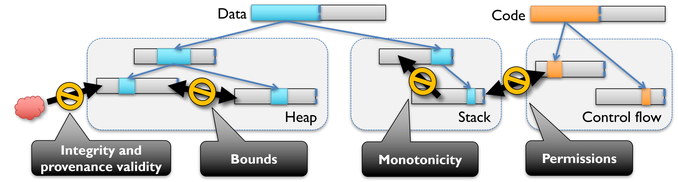
Pointers in conventional architectures



- Implemented as **integer virtual addresses (VAs)**
- (Usually) point into **allocations, mappings**
  - **Derived** from other pointers via integer arithmetic
  - **Dereferenced** via jump, load, store
- **No integrity protection** – pointers can be injected/corrupted
- **Arithmetic errors** – overflows, out-of-bounds leaks/overwrites
- **Inappropriate use** – executable data, format strings

➢ In current architectures, attacks on data and code pointers are highly effective, often achieving **arbitrary code execution**

CHERI pointer protection

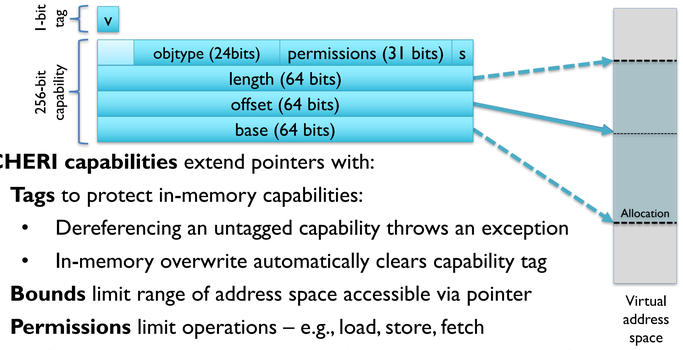


➡ **Provence and monotonicity control whether pointers can be dereferenced**
- **Valid pointers** are derived from other valid pointers via valid transformations
- E.g., received network data cannot be interpreted as a code pointer

➡ **Bounds and permissions constrain use, and can be minimized by software**
- E.g., pointers cannot be manipulated to access other heap or stack objects

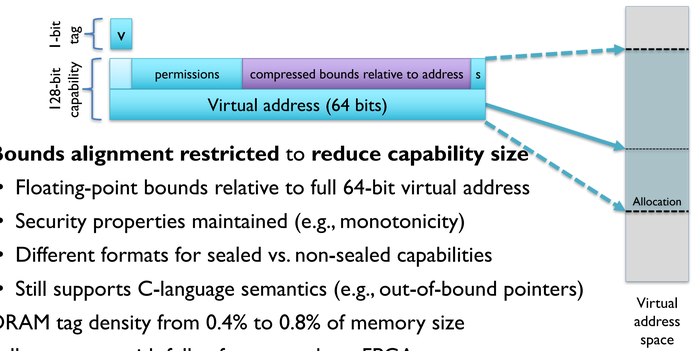
➡ Strong foundations for **software memory protection and compartmentalization**

256-bit architectural capabilities



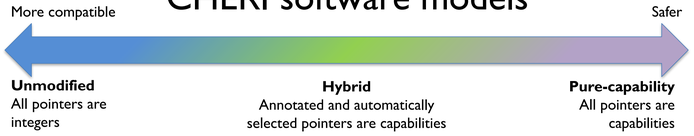
CHERI capabilities extend pointers with:
- **Tags** to protect in-memory capabilities:
  - Dereferencing an untagged capability throws an exception
  - In-memory overwrite automatically clears capability tag
- **Bounds** limit range of address space accessible via pointer
- **Permissions** limit operations – e.g., load, store, fetch
- **Sealing** for encapsulation: **immutable**, **non-dereferenceable**

128-bit micro-architectural capabilities



- **Bounds alignment restricted** to **reduce capability size**
  - Floating-point bounds relative to full 64-bit virtual address
  - Security properties maintained (e.g., monotonicity)
  - Different formats for sealed vs. non-sealed capabilities
  - Still supports C-language semantics (e.g., out-of-bound pointers)
- DRAM tag density from 0.4% to 0.8% of memory size
- Full prototype with full software stack on FPGA

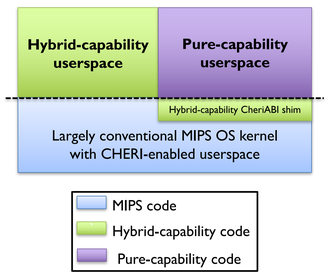
CHERI software models



- Source and binary compatibility – multiple C-language and code-generation models:
  - **Unmodified code**: Existing n64 code runs without modification
  - **Hybrid code**: E.g., capabilities used in return addresses, annotated data/code pointers, specific types, etc. (n64-interoperable)

    … But "hybrid" is a spectrum between manual and automatic use
  - **Pure-capability code**: Ubiquitous data- and data-pointer protection. (Non-n64-interoperable due to changed pointer size) – also a spectrum
- **CHERI Clang/LLVM compiler prototype generates code for all**

From hybrid-capability code to pure-capability code



- **n64 MIPS ABI:** hybrid-capability code
  - Early investigation – manual annotation and experimenal C semantics
  - Many pointers are integers (including syscall arguments, most implied VAs)
- **CheriABI:** pure-capability code
  - The last two years – fully automatic use of capabilities wherever possible
  - All pointers, implied virtual addresses are capabilities (inc. syscall arguments)
- **CheriABI runs a full UNIX userspace**

Approved for public release; distribution is unlimited. Sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts FA8750-10-C-0237 ("CTSRD") and FA8750-11-C-0249 ("MRC2") as part of the DARPA CRASH and DARPA MRC 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, either expressed or implied, of the Department of Defense or the U.S. Government. Additional support was received from St John's College Cambridge, the Google SOAAP Focused Research Award, the RCUK's Horizon Digital Economy Research Hub Grant (EP/G065802/1), the EPSRC REMS Programme Grant (EP/K008528/1), the EPSRC Impact Acceleration Account (EP/K503757/1), the Isaac Newton Trust, the UK Higher Education Innovation Fund (HEIF), Thales E-Security, ARM Ltd, and HP Enterprise.

SRI International

UNIVERSITY OF CAMBRIDGE