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 pointer 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++-based systems, such as UNIX, and the C and C++ programming languages, supporting incremental deployment of the approach within current hardware-software ecosystems.

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 costs are comparable to a function call rather than Inter-Process Communication (IPC). CHERI’s compartmentalization 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, ESPRC, 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, multicores FPGA processor design to explore microarchitectural impacts. CHERI's hybrid capability model has allowed us to adapt the Clang/LVM 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 SSS, 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/](http://www.cheri-cpu.org/)