CHERITech’24 - Welcome

The William Gates Building
University of Cambridge
23 April 2024

Professor Robert N. M. Watson
Professor Simon W. Moore
Franz A. Fuchs
Welcome!

- The CHERI project kicked off in this building in late 2010 with the generous support of DARPA.
- Since joined by Arm, Google, Microsoft, and of course funders such as InnovateUK, and others in supporting a massive expansion of scope and interest.
- We thank these and other sponsors for their support for this event!

- Capability systems are, of course, an old idea, even if our application of the concept is very contemporary. Do make sure you take a look at the CAP Computer in The Street, completed in 1977!
Some administrative things

- You are in Lecture Theatre 1 (LT1), which will be where all of our talks are
- Just outside is The Street, where we will have poster sessions, coffee, and lunch
- Bathrooms can be found off The Street, opposite the stairs
- If you need help with a taxi or similar matters, the building’s Reception can surely lend a hand
- There are no planned fire drills today ...
Enunciating visions for CHERI adoption

• CHERI is a very challenging technology to transition
  • Hardware and software R&D cycles that differ enormously
  • Supply-chain challenges make deploying any disruptive change hard
  • Evaluating and selling security to generalist markets very difficult

• But the potential wins are also huge
  • Achieve strong memory safety without ground-up rewrite of all software
    (nearly infinitely expensive, and so will not happen in less than 30 years)
  • Enable compartmentalized software designs capability of resisting arbitrary
    code execution, software supply-chain adversaries

• To succeed, we need to identify and engage with
  • Clearly enunciated visions for use and deployment
  • Both technical and non-technical obstacles to use and adoption
Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-Design and -Default

Publication: April 13, 2023
Cybersecurity and Infrastructure Security Agency
NSA | FBI | ACSC | NSC-US | CPO | BSI | NCSC-NL | CERT NZ | NCSC-NZ

Disclaimer: This document is marked TLP:CLEAR. Disclosure is not limited. Sources may use TLP:CLEAR when information cannot remain as no foreseeable risk of misuse. In accordance with applicable laws and procedures for public release. Subject to standard copyright rules. TLP:CLEAR information may be distributed without restriction. For more information on the Traffic Light Protocol, see http://www.cisecurity.org/tlp/

The authoring agencies encourage the use of Secure-by-Design tactics, including principles that reference SSDF practices. Software manufacturers should develop a written roadmap to adopt more Secure-by-Design software development practices across their portfolio. The following is a non-exhaustive list of illustrative roadmap best practices:

- **Memory safe programming languages (SSDF PW.6.1):** Prioritize the use of memory safe languages wherever possible. The authoring agencies acknowledge that other memory specific mitigations, such as address space layout randomization (ASLR), control-flow integrity (CFI), and fuzzing are helpful for legacy codebases, but insufficient to be viewed as secure-by-design as they do not adequately prevent exploitation. Some examples of modern memory safe languages include C#, Rust, Ruby, Java, Go, and Swift. Read NSA’s memory safety information sheet for more.

- **Secure Hardware Foundation:** Incorporate architectural features that enable fine-grained memory protection, such as those described by Capability Hardware Enhanced RISC Instructions (CHERI) that can extend conventional hardware Instruction-Set Architectures (ISAs). For more information visit, University of Cambridge’s CHERI webpage.

- **Secure Software Components (SSDF PW 4.1):** Acquire and maintain well-secured software components (e.g., software libraries, modules, middleware, frameworks,) from verified commercial, open-source, and other third-party developers to ensure robust

- **Static and dynamic application security testing (SAST/DAST) (SSDF PW.7.2, PW.8.2):**
The chip, in particular, is an important hardware building block to consider. There are several promising efforts currently underway to support memory protections through hardware. For example, a group of manufacturers have developed a new memory-tagging extension (MTE) to cross-check the validity of pointers to memory locations before using them. If they are invalid, the CPU produces an error. This technique is an effective method to detect memory safety bugs, but this approach should not be considered a comprehensive solution to prevent all memory safety exploits. Another example of a hardware method is the Capability Hardware Enhanced RISC Instructions (CHERI). This architecture changes how software accesses memory, with the aim of removing vulnerabilities present in historically memory unsafe languages.
From drumbeat to standardisation

• An idea gaining increasing currency to help answer the question: “How can consumers ask for systems with memory safety?”

• Complex path including technical consensus building (“what is memory safety?”) and vast tricky tradeoffs and potential pitfalls

• Want an inclusive definition across methodologies: At least (perhaps multiple) of CHERI, Rust, formal methods, … but accept:
  • Differing capabilities, adoption tradeoffs, adversary models, limitations

• Reward early adoption (exploit mitigations) while motivating ratcheting up of ambition over time (e.g., from PAC to CHERI)
  • From secrets-based/probabilistic to deterministic protection
  • Mature adversary models including attackers with arbitrary code execution

• Be aware that this discussion is coming ..
  • May go nowhere, but likely essential to produce the supply-chain pull to get beyond widely lauded but actually token Rust deployments with limited impact
CHERI Alliance CIC

• A UK-based Community Interest Corporation (CIC)
• Provide a space enabling companies, universities, and governments to pool resources to promote and enable CHERI
• CIC now created legally, and in framework development
• Planned Autumn launch with initial membership
• Support efforts across CHERI-enabled architectures, such as:
  • Common marketing material
  • Standardisation and certification activities
  • Software ecosystem enablement
  • Efforts such as memory-safety standardization
• Your feedback and participation very much invited!
CheriBSD 2024.05 software release coming soon

Reference design for CHERI integration into a mainstream, open-source OS and application stack

Approaching 100MLoC of memory-safe, compartmentalised C/C++ on a shipping prototype Arm Morello board today:

- CheriBSD kernel with DRM + Panfrost drivers
- CheriBSD userspace with libraries and tools
- OpenGL, Wayland display server
- Desktop: Plasma, KDE base applications including Dolphin, Okular, Kate, Konsole, …
- Server: nginx, Postgres, gRPC, …
- “Ubiquitous” Library compartmentalization of all memory-safe userlevel components
- Complete software development environment including Clang/LLVM, Git, GDB, Ghidra, …
- Roughly 10K memory-safe third-party software packages, and 20K aarch64 packages

Some more complex, un-adapted applications (e.g., Chromium, OpenJDK) running with 64-bit Arm support
Demonstration

CheriBSD 2024.05 development snapshot running on Arm Morello desktop system