Crash course in CHERI in 10 minutes (OMG)

Professor Robert N. M. Watson
University of Cambridge, CHERI Alliance, and Capabilities Limited

CHERI Blossoms Conference
University of Cambridge
26-27 March 2026

Approved for public release; distribution is unlimited.

Development of the CHERI concept 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"), FA8650-18-C-7809 ("CIFV"), HR001122C0110 ("ETC"), HR001123C0031 ("MTSS"), and FA8750-24-C-B047 ("DEC") as part of the DARPA I2O CRASH, I2O MRC, MTO SSITH, and I2O CPM research programs. The views, opinions, and/or findings contained in this document 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.

Arm's Morello and the Morello-enabled software stack, as well as development of the RV64Y Instruction-Set Architecture (ISA) standard, were supported by the **Innovate UK** projects 105694 ("Digital Security by Design (DSbD) Technology Platform Prototype"), 107145 ("Assessing the Viability of an Open-Source CHERI Desktop Software Ecosystem"), 10027440 ("Developing and Evaluating an Open-Source Desktop for Arm Morello"), 10027332 ("MOJO - A Robust Java Virtual Machine for Morello"), 10168042 ("CheriBSD feature extraction, maturity, and testing), and 10168492 ("CHERI for Operational Safety in Memory-Isolated Cores").

Research from 2025 was supported primarily by the **DSIT and EPSRC** CHERI Research Centre (CRC) grant UKRI3001.

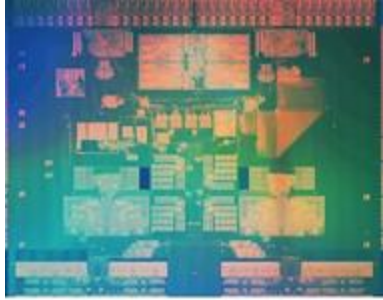
We further acknowledge EPSRC REMS (EP/K008528/1), EPSRC CHaOS (EP/V000292/1), ERC ELVER (789108), the Isaac Newton Trust, the UK Higher Education Innovation Fund (HEIF), Thales E-Security, Microsoft Research Cambridge, Arm Limited, Google, Google DeepMind, HP Enterprise, Codasip, and the Gates Cambridge Trust.

First: A very special welcome!

- **Thank you to the CHERI Alliance for organizing this event that we are proud to host!**
- Please join us for the **CHERI 15th Birthday Party** + conference reception
  - Today starting at 17:30
  - A few words of thanks
  - Some light dinner and birthday cake!
- Do visit the **CHERI Exhibition** in the Street
  - The most recent test chips for CHERI microcontroller tape-outs, various processor IP products and prototypes on FPGA, Arm Morello chips and demos, and a large (and still growing) pile of PhD dissertations
  - Now extended through 30 April 2026

A taste of the history

What is CHERI?



Morello prototype – 7nm quad-core multi-GHz Arm processor and SoC with CHERI extensions, Arm, 2022

CHERI is a processor **architectural protection model**

- Composes a **capability-system model** with hardware and software
- Adds new security primitives to Instruction-Set Architectures (ISAs)
- Implemented by microarchitectural extensions to the CPU and SoC
- Enables new security behaviour in software

CHERI mitigates vulnerabilities in **C/C++ Trusted Computing Bases**

- **C/C++ TCBs**: Hypervisors, operating systems, language runtimes, browsers, ….
- **Fine-grained memory protection** deterministically closes many arbitrary code execution attacks, and directly impedes common exploit-chain tools
- **Scalable compartmentalization** mitigates many vulnerability classes .. Even unknown future classes .. by extending the idea of software sandboxing

Multiple industrial implementations (Arm, Microsoft, Codasip, CapLtd, Google, …)

Capability systems: Origins in 1970s computing research

The CAP computer project ran from 1970-1977 at the University of Cambridge, led by R. Needham, M. Wilkes, and D. Wheeler.

The capability system is a **design pattern** for how CPUs, languages, OSes, … can control access to resources:

- **Capabilities** are communicable, unforgeable tokens of authority
- In **capability-based systems**, resources are reachable only via capabilities

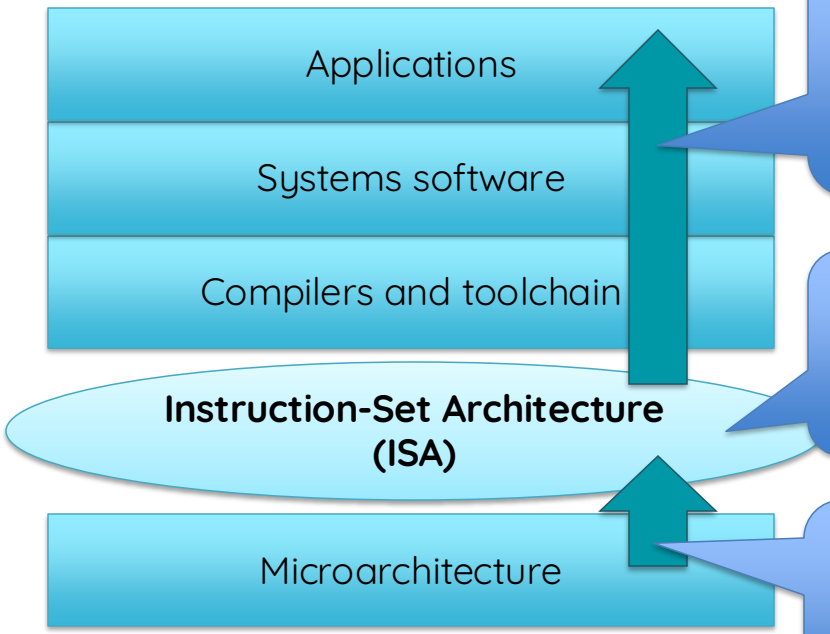
Capability systems limit the **scope and spread of damage** from accidental or intentional software misbehaviour

They do this by making it **natural and efficient** to implement, in software, two security design principles:

- The **principle of least privilege** dictates that software should run with the minimum privileges to perform its tasks
- The **principle of intentional use** dictates that when software holds multiple privileges, it must explicitly select which to exercise

These principles are the heart of the CHERI design

Architectural primitives for software security



Software configures and uses capabilities to continuously enforce safety properties such as **referential, spatial, and temporal memory safety,** as well as higher-level security constructs such as **compartment isolation**

CHERI capabilities are an **architectural primitive** that compilers, systems software, and applications use to constrain their own future execution

The microarchitecture implements the **capability data type** and **tagged memory**, enforcing invariants on their manipulation and use such as **capability bounds**, **monotonicity**, and **provenance validity**

Two key applications of CHERI in software security

Efficient, fine-grained memory protection for C/C++

- Strong source-level compatibility, but requires recompilation
- Deterministic and secret-free referential, spatial, and temporal memory safety
- Retrospective studies estimate ⅔ of memory-safety vulnerabilities mitigated
- Generally modest overhead (0%-5%, some pointer-dense workloads higher)

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 FPGA-based benchmarks)

CHERI allows incremental deployment in software

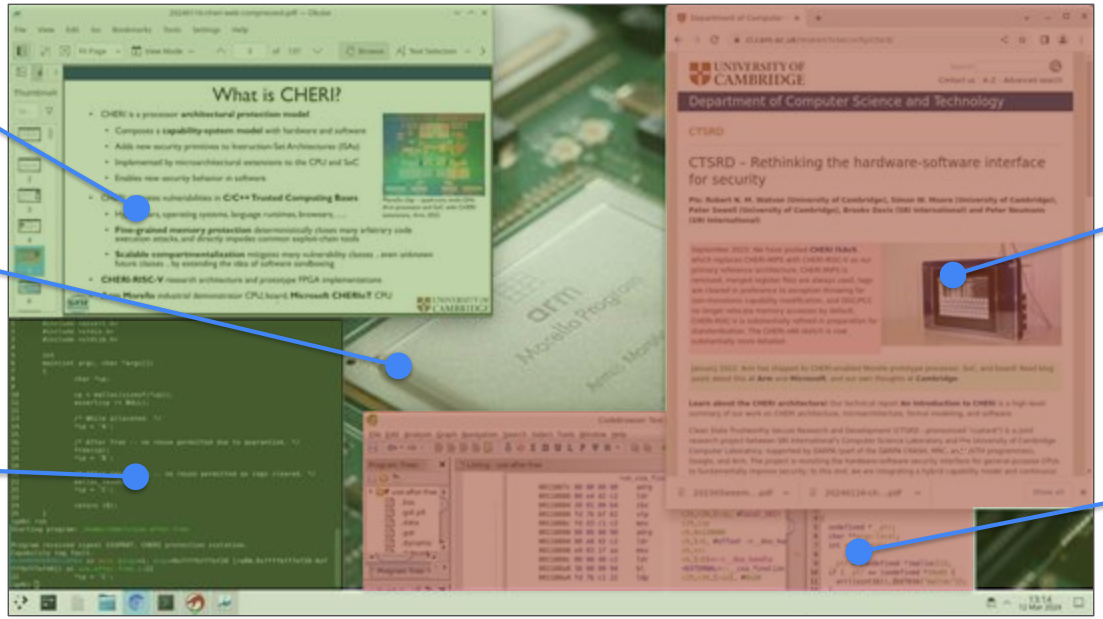
Enables incremental application migration to memory safety

Memory-safe PDF viewer

Memory-safe desktop environment

Memory-safe terminal window and commands

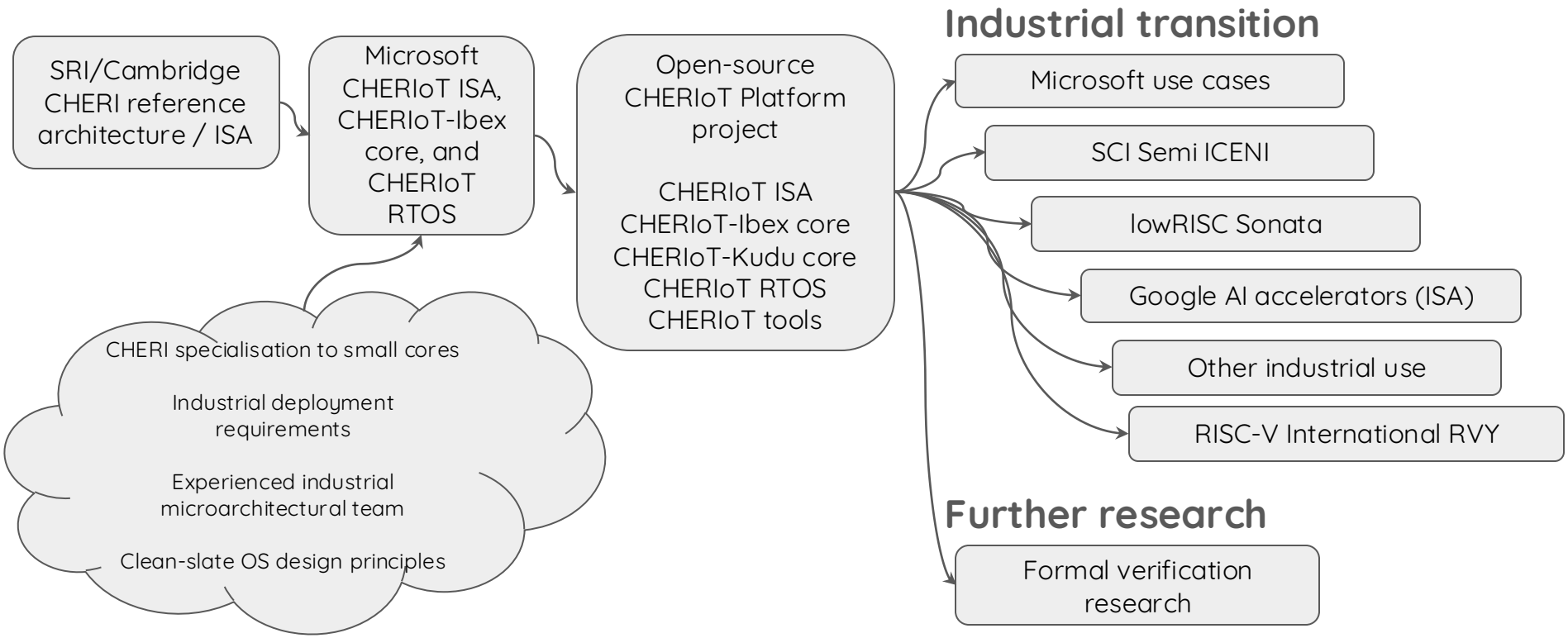
Memory-safe OS kernel



Legacy 64-bit Arm Chromium browser - **Memory-safe prototype as of late 2025!**

Legacy 64-bit Arm JVM

CHERIoT Platform and the importance of open-source reference designs



See CHERI at a range of scales in the exhibit!

Mobile devices, data centers

- Arm Morello application core + SoC, based on existing Arm Neoverse N1 μarch
- 64-bit Arm-A baseline ISA
- Multicore, MMU-enabled, out-of-order core 2.5GHz
- CHERI-adapted FreeBSD, Linux, seL4, VxWorks OSes

Automotive, embedded, high-end IoT, secure enclaves

- Codasip X730 and CapLtd CVA6-CHERI application cores
- 32/64-bit RISC-V baseline ISA
- Dual-issue, pipelined, with MMU
- CHERI-adapted FreeBSD, Linux, seL4 OSes

IoT, roots of trust

- Microsoft CHERIoT Ibex microcontroller
- 32-bit RISC-V baseline ISA
- 3-stage pipeline, no MMU, 200-300MHz
- CHERIoT RTOS embedded OS
- **Taped out test chips on display!**

Results in real-world software

AIxCC nginx evaluation with CHERI

DARPA AI Cyber Challenge (AIxCC) applies AI to vulnerability discovery and patching

- Reintroduced past vulnerabilities / created new vulnerabilities, in open-source software – Challenge Problems (CPs)

One such problem adds vulnerabilities to the open-source nginx web server

- Already running with CHERI spatial/temporal safety following DSTL study
- AIxCC added 14 memory-safety Challenge Problem Vulnerabilities (CPVs)
- ASAN reproduction trigger provided for each CPV

The AIxCC team didn't know about CHERI, and the CHERI team didn't know about the AIxCC nginx work – so this is a great science experiment:

- **Query:** How would CHERI fare in mitigating vulnerabilities selected without regard to the protection approach?
- **Methodology:** Adversarial analysis (e.g., exploit aarch64 version, and attempt exploitation for CHERI version)

AIxCC nginx conclusions

CHERI deterministically mitigated 114% of 14 applicable CPVs:

12 intended spatial and temporal vulnerabilities deterministically mitigated

2 further unintended memory-safety vulnerabilities introduced by system designers (NULL overwrite, linear overflow) also deterministically mitigated

These sorts of empirical studies generate not only interesting evidence about CHERI's effectiveness, but also new methodology in understanding exploitation in the presence of memory safety

Sample vulnerability: CVE-2023-4863 (“BLASTPASS”)

CHERI deterministically mitigates this critical Chromium vulnerability without any awareness about the nature, location, or origin of the vulnerability during development.

This memory-safety vulnerability was discovered “in the wild” following targeted attacks against civil rights activities in the US using NSO Group’s Pegasus product:

- **Naturally occurring vulnerability** in Google’s libwebp library
  - Heap-memory buffer overflow exploitable for remote arbitrary code execution
  - Undiscovered for years despite fuzzing due to (modest) complexity of Huffman coding logic
- Affected **Chrome**, **Edge**, and **WebKit**
  - 1st-party code for Google
  - 3rd-party for Apple and Microsoft
  - Zero interaction exploitation of Apple iOS
- Discovered **after** our adaptation of webp (0% LoC change)

Deterministically mitigated due to CHERI C/C++ memory safety

Conclusion

It is a very exciting time for CHERI

- **Validation of the key design concepts** across multiple microarchitectural scales and hundreds of millions of lines of C/C++
- **Industrial adoption** spans startups to trillion-dollar tech companies; early deployment focus on “vertically integrated ecosystems”:
  - Microcontrollers for industrial control, critical infrastructure, consumer IoT
  - Essential TCBs in all scales of devices: Roots of Trust and Secure Enclaves
  - AI accelerator hardware with critical exposure to personal data, use in biometrics
- Next step is engaging with “open ecosystems” on application cores, building on “embedded” progress: harder but very high impact pitch
- A very active research community looking at topics such as integration with language runtimes and programming languages, novel microarchitecture, formal verification, adversarial research, …

IEEE S&P Test of Time Award for CHERI compartmentalization

IEEE CS 2024 Best Paper Award for CHERI memory protection at scale

Questions?