CHERITech'26
CONFERENCE

CHERI

12-13 Nov 2026
Munich, DE

Introduction and State of CHERI

Why CHERI, how it works, and where it is going

Jonathan Woodruff
Assistant Research Professor
University of Cambridge

Located at semi | SEMICON EUROPA

Why CHERI?

- A new instruction-set extension: **every pointer doubles in size**
- The operating system and applications must be **recompiled**
- At least as big a shift as **32- to 64-bit** – a transition we are still finishing

So why would anyone do this?

1960s–70s: programs outgrow their protection

Inside a program

- Programs grow: more objects, more components, many authors
- Enforce the language's rules – scope, types, sizes – so nobody else's code surprises you
- New languages to express the protection programmers wanted

Between programs

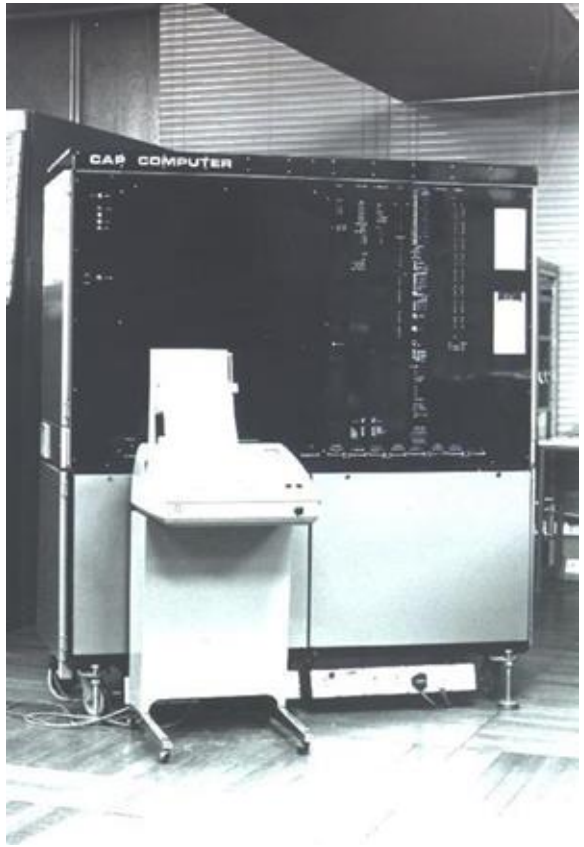
- **Multi-programmed** computers: many users, many jobs, one machine
- Users must be protected from each other – they aren't even doing the same job

+ Security

- Defence funding: it must be impossible to spy on or subvert the computer

Programs were often written in the instruction set itself – so **hardware designers took on the problem**

The hardware answer: capability machines

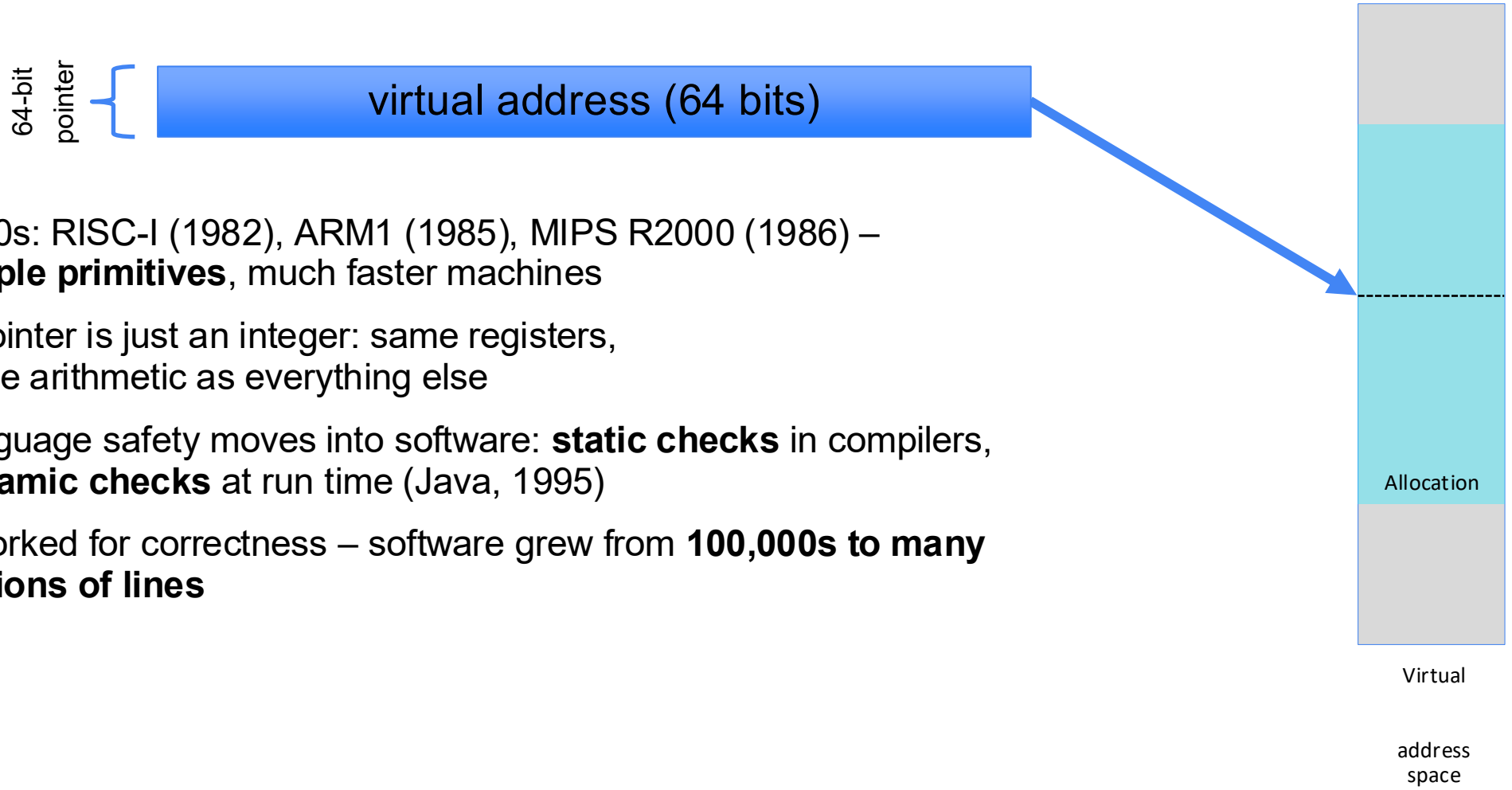


The CAP computer project ran from 1970-1977 at the University of Cambridge, led by R. Needham, M. Wilkes, and D. Wheeler.

Capability: an unforgeable token of authority. A program holds exactly the capabilities it needs, and every memory object is bounded.

1961	Burroughs B5000 – tagged memory and bounds-checked descriptors, built for ALGOL
1966	Dennis & Van Horn, MIT – capabilities for multiprogrammed computers
1972	Plessey System 250 – first operational capability-addressed computer
1970–77	CAP computer, Cambridge – Wilkes, Needham, Wheeler
1973–80	PSOS, SRI – a formally specified capability operating system
1979–81	IBM System/38, Intel iAPX 432 – capabilities go commercial

The RISC bargain: addresses are integers



- 1980s: RISC-I (1982), ARM1 (1985), MIPS R2000 (1986) – **simple primitives**, much faster machines
- A pointer is just an integer: same registers, same arithmetic as everything else
- Language safety moves into software: **static checks** in compilers, **dynamic checks** at run time (Java, 1995)
- It worked for correctness – software grew from **100,000s to many millions of lines**

Then everything got connected

359,000

Code Red · 2001

servers infected within a week – IIS buffer overflow

10 min

SQL Slammer · 2003

to infect most of its 75,000 victims – SQL Server buffer overflow

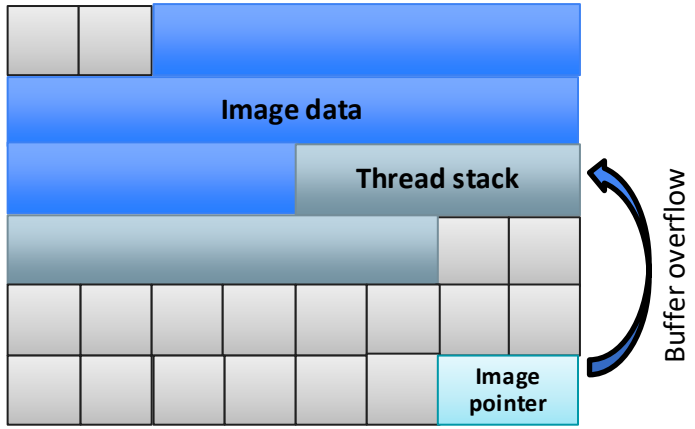
423,000

Blaster · 2003

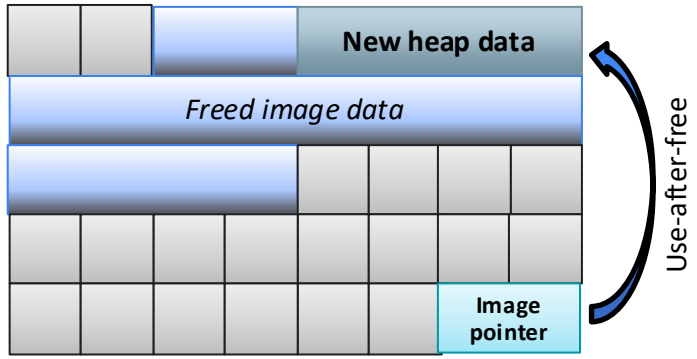
PCs infected in four days – Windows RPC buffer overflow

Every computer was now one hop from organised crime

Memory safety: still ~70% of the problem



Example of spatial unsafety



Example of lifetime unsafety

~70%

of the vulnerabilities Microsoft assigns a CVE each year are memory-safety issues

~70%

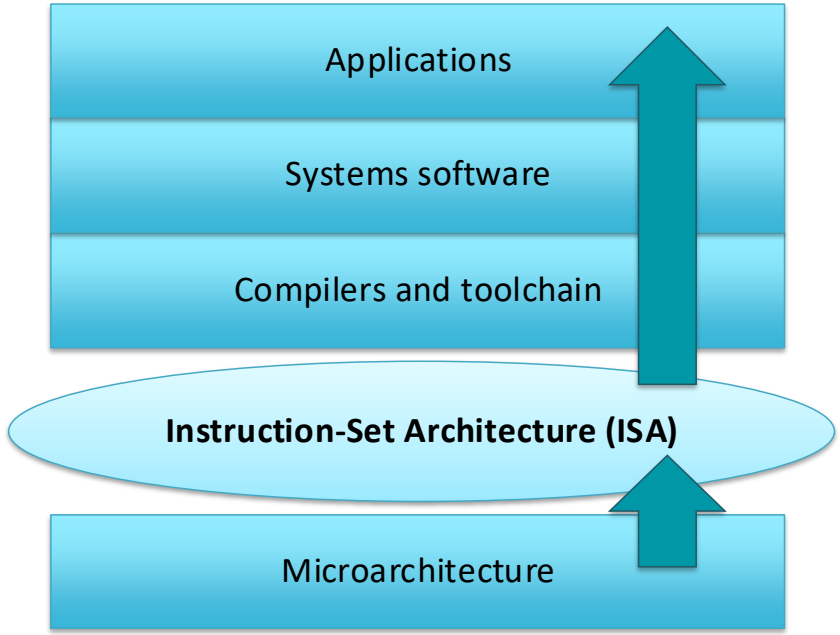
of Chromium's high-severity security bugs are memory-safety problems

Two classes dominate: **spatial** (buffer overflow) and **temporal** (use-after-free) – despite our best software defences

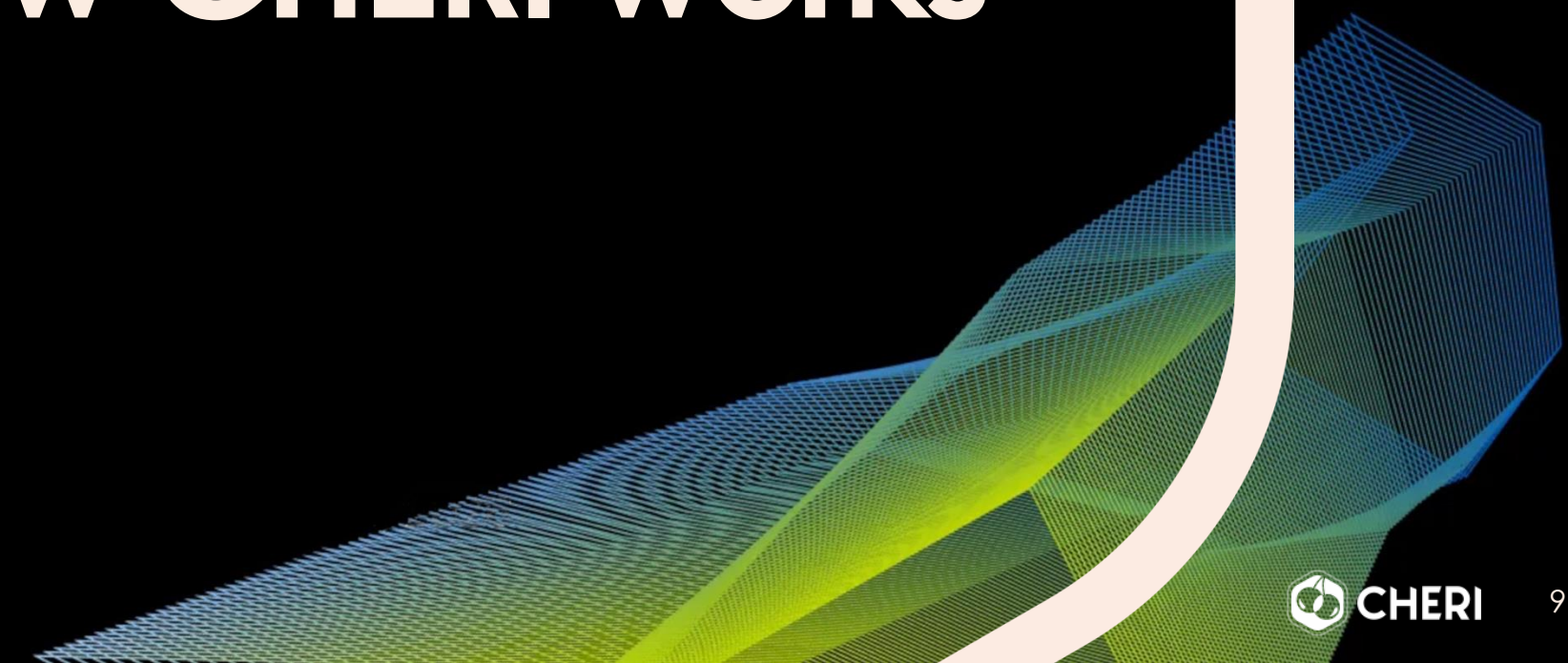
CHERI rethinks RISC

Capability Hardware Enhanced RISC Instructions

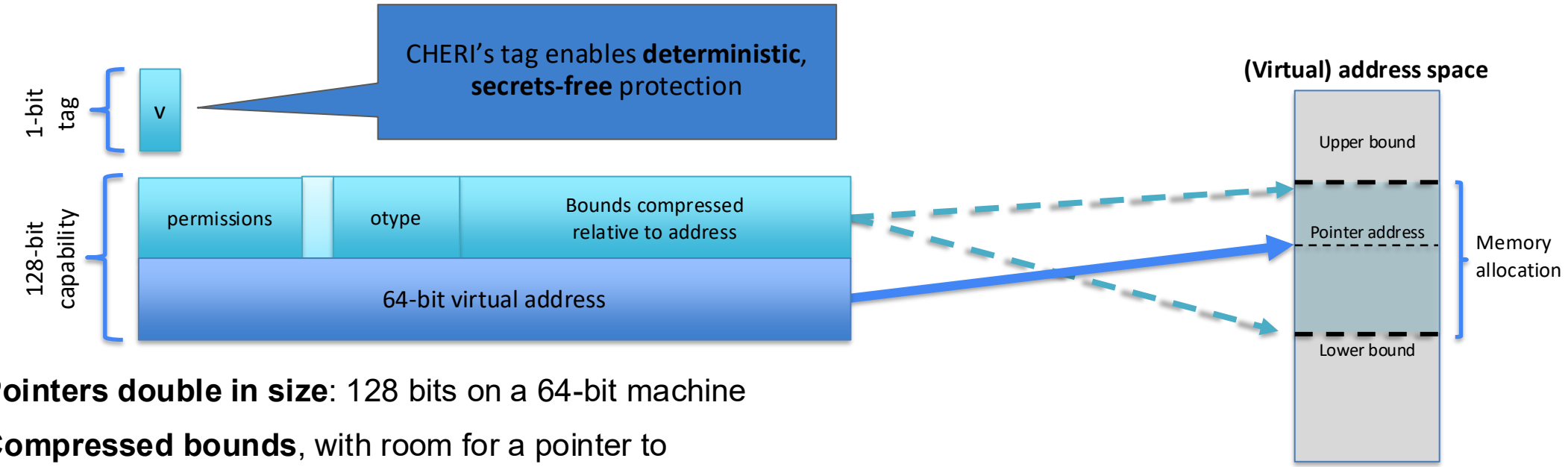
- Not a new ISA: new instructions added to **existing** RISC ISAs
- RISC optimises the common case – and now **memory safety is the common case**
- MMUs already enforce isolation between address spaces; CHERI enforces it for **language objects**:
  - in bounds
  - only what you were delegated
  - only what you intended
  - only while it exists
- A foundation for **compartments** within a program – or even for **process isolation**



How CHERI works



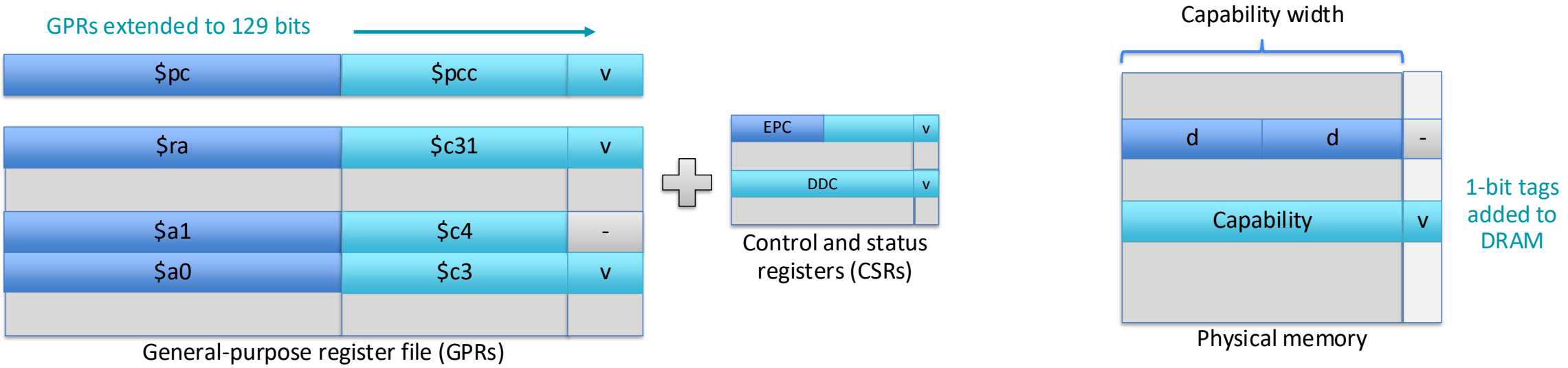
CHERI capabilities: bounded, unforgeable



- **Pointers double in size**: 128 bits on a 64-bit machine
- **Compressed bounds**, with room for a pointer to wander out of bounds
- **Permissions** (read, write, execute, …) and an **object type** for sealing
- **1-bit tag** outside the address space: data writes clear it, nothing can set it – **unforgeable**

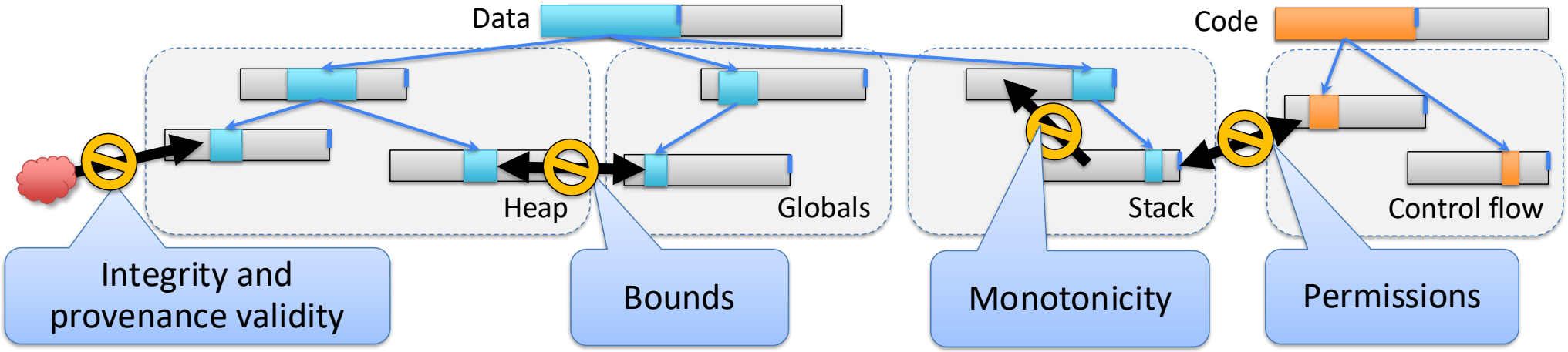
Provenance validity and **monotonicity** (i.e. legal modification can only reduce privilege) make these **capabilities** and not simply 'fat pointers'

Capability registers and tagged memory



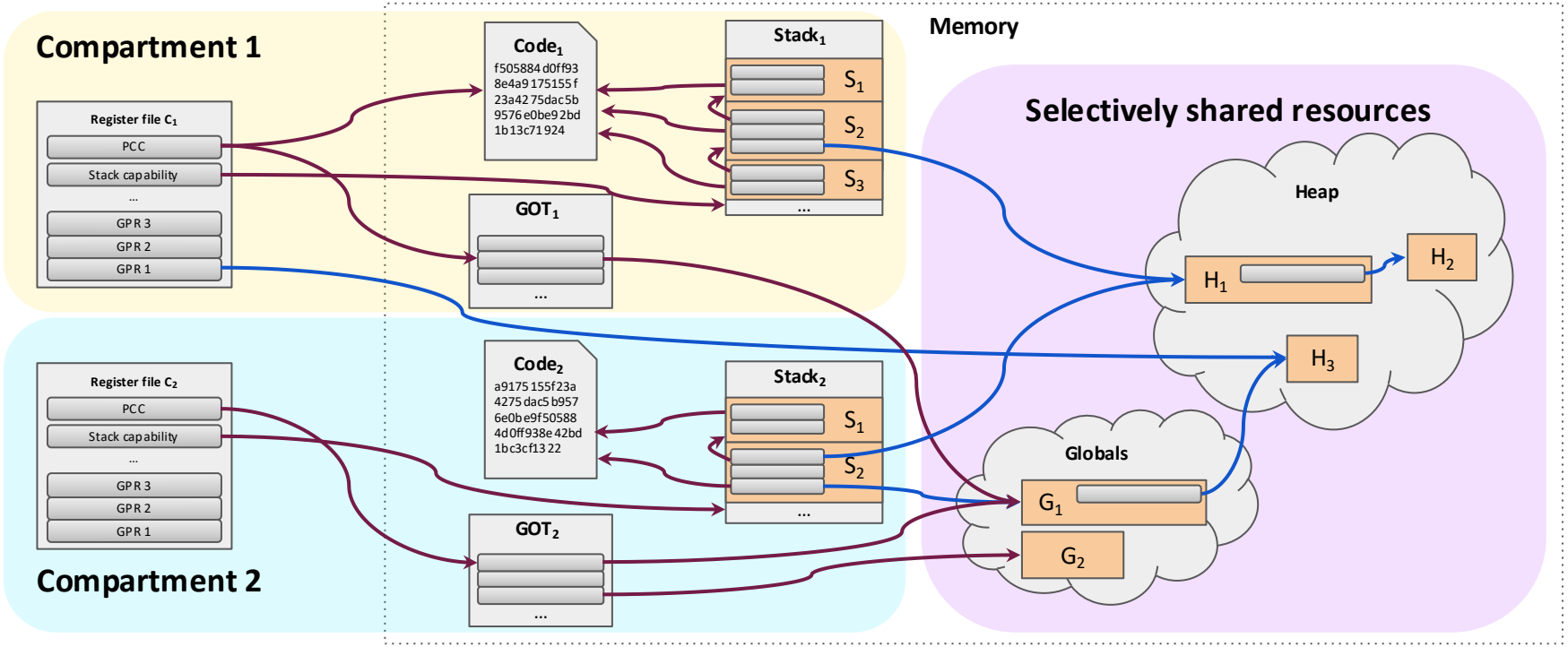
- Integer registers widen to hold capabilities – just as they did from 32 to 64 bits
- The program counter and system registers that hold pointers become capabilities
- One **tag bit per capability-sized word** in memory, carried through caches and the interconnect
- Legacy loads and stores are constrained by a **default data capability** – old code keeps working

Memory safety by construction



- Compile with a **capability ABI**: every pointer – explicit or implied – is a capability
- A heap pointer can never become a pointer to globals or the stack
- Code capabilities aren't writable; data capabilities can't be executed
- **Temporal safety**: revoke pointers to freed memory, efficiently
- Deterministic and secret-free – **recompile, don't rewrite**

Compartments in a single address space



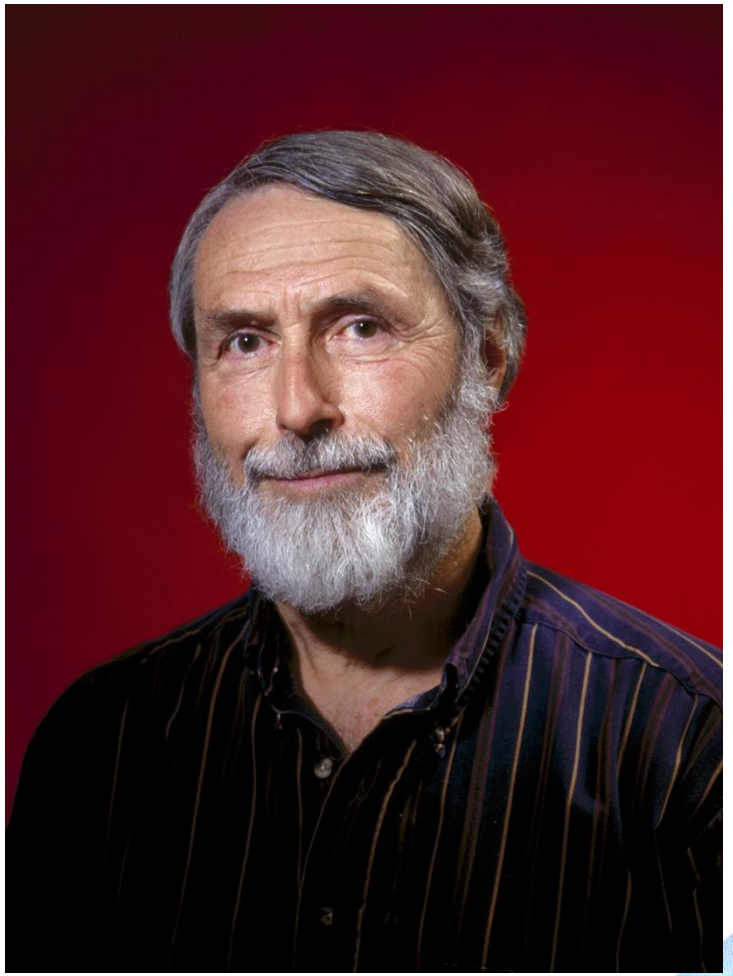
- A compartment is just the set of capabilities it holds – **non-contiguous**, with no new page tables
- Domain transitions at function-call speed; sharing memory is **passing a pointer**
- Library compartmentalisation ships in CheriBSD; co-located processes are being explored

How we got here

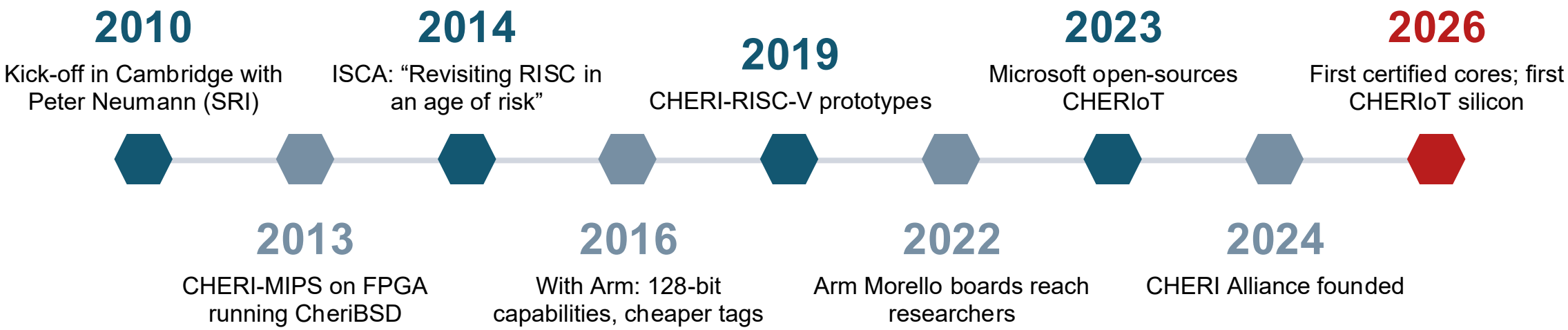
Peter G. Neumann

1932 – 2026

- Worked on **Multics** in the 1960s; at SRI from 1971
- Led **PSOS** (1973–80): a capability operating system, formally specified layer by layer
- SRI's principal investigator on **CHERI** from its start in 2010
- Mentor and friend of many, many of us on the CHERI project for 16 years



From a sketch to silicon: 2010–2026



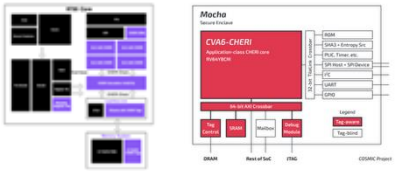
Demonstrated at every scale

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



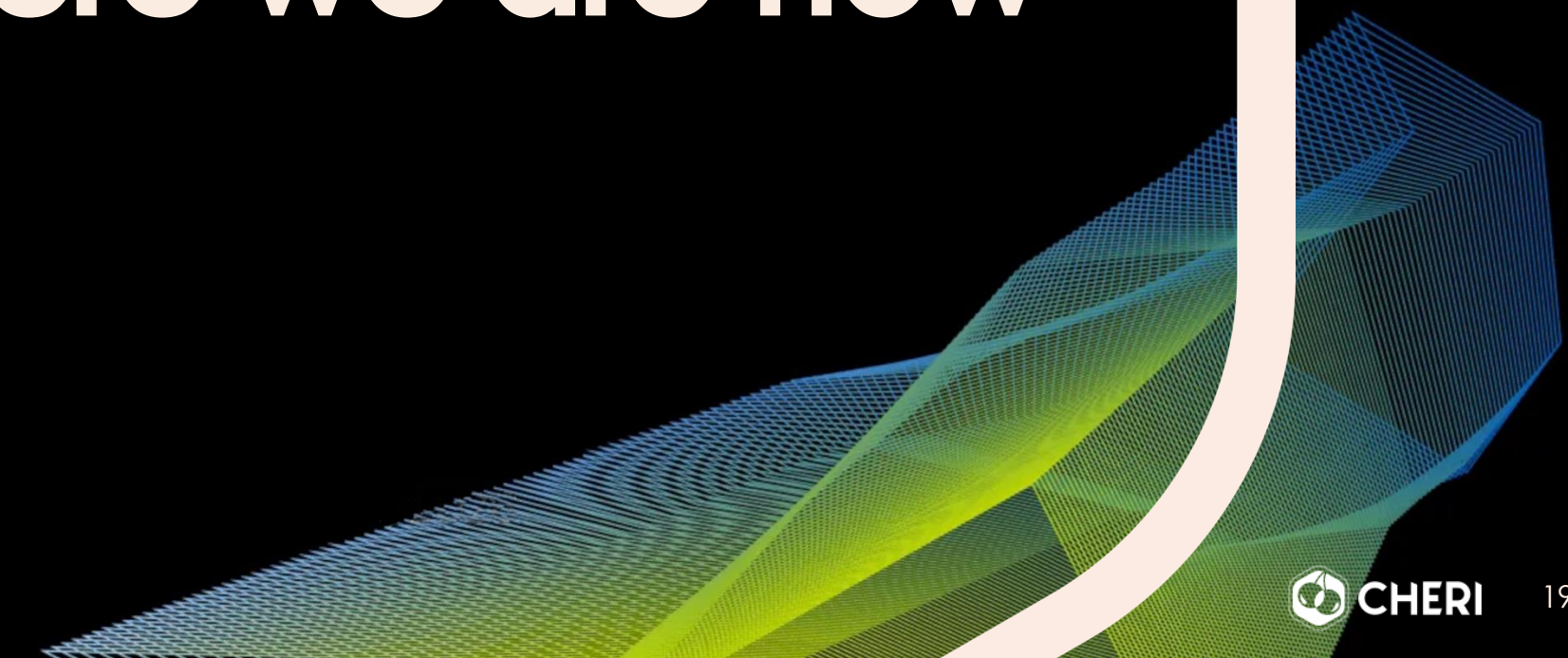
- Codasip X730 and CapLtd CVA6-CHERI application cores
- 32/64-bit RISC-V baseline ISA
- Dual-issue, pipelined, with MMU
- CHERI-adapted FreeBSD, Linux, seL4 OSes

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 available today

Where we are now



Standardising CHERI-RISC-V: RVY

- RISC-V International gave CHERI the letter **Y**: RV32Y and RV64Y
- Draft specification is **Stablising** (v0.9.x); final feedback from the Architecture Review Committee
- Expect a **frozen spec in the coming months**, then ratification
- Multiple vendors already building IP to the draft

Microsoft lowRISC Google EnSilica SCI Codasip CAPABILITIES LIMITED SECQAI

Multiple test chips for RVY silicon targeting 2026/2027



Products aren't waiting

Codasip X730

- Dual-issue, Linux-capable application core
- Alliance-certified; championing ratification
- Higher-performance cores in development

CVA6-CHERI

- Open-source RV64Y application core (CapLtd)
- Runs CheriBSD; COSMIC secure enclave with lowRISC

Research cores

- Cambridge CHERI-Toooba and others tracking the near-ratified RVY

Microsoft CHERIoT-Ibex

- Open-source microcontroller core
- CHERI Alliance–certified (Mar 2026)

SCI Semiconductor ICENI

- CHERIoT microcontroller: first silicon back
- On sale next year; heavy software investment

SECQAI SE01

- CHERI TPM with post-quantum crypto
- Taped out in 22nm (Nov 2025)

Welcome to CHERITech'26

- New research platforms and research results
- Product and software progress from across the community
- Standardisation progress,
  both regulatory and industry

cheri-cpu.org · cheri-alliance.org

CHERITech'26 CONFERENCE

12–13 Nov 2026
Munich, DE

Located at semi | SEMICON EUROPA



CHERI

THANK YOU

Contact Jonathan.Woodruff@cl.cam.ac.uk

Web cheri.cst.cam.ac.uk

CHERI™ and the CHERI logo are protected trademarks. Use of these marks is subject to the CHERI Alliance Trademark Policy – https://cheri-alliance.org/legal-and-policies/trademark/

This work © 2026 by CHERI Alliance is licensed under CC BY-SA 4.0 (Creative Commons Attribution-ShareAlike 4.0 International) – https://creativecommons.org/licenses/by-sa/4.0/

Backup



Recompile, don't rewrite

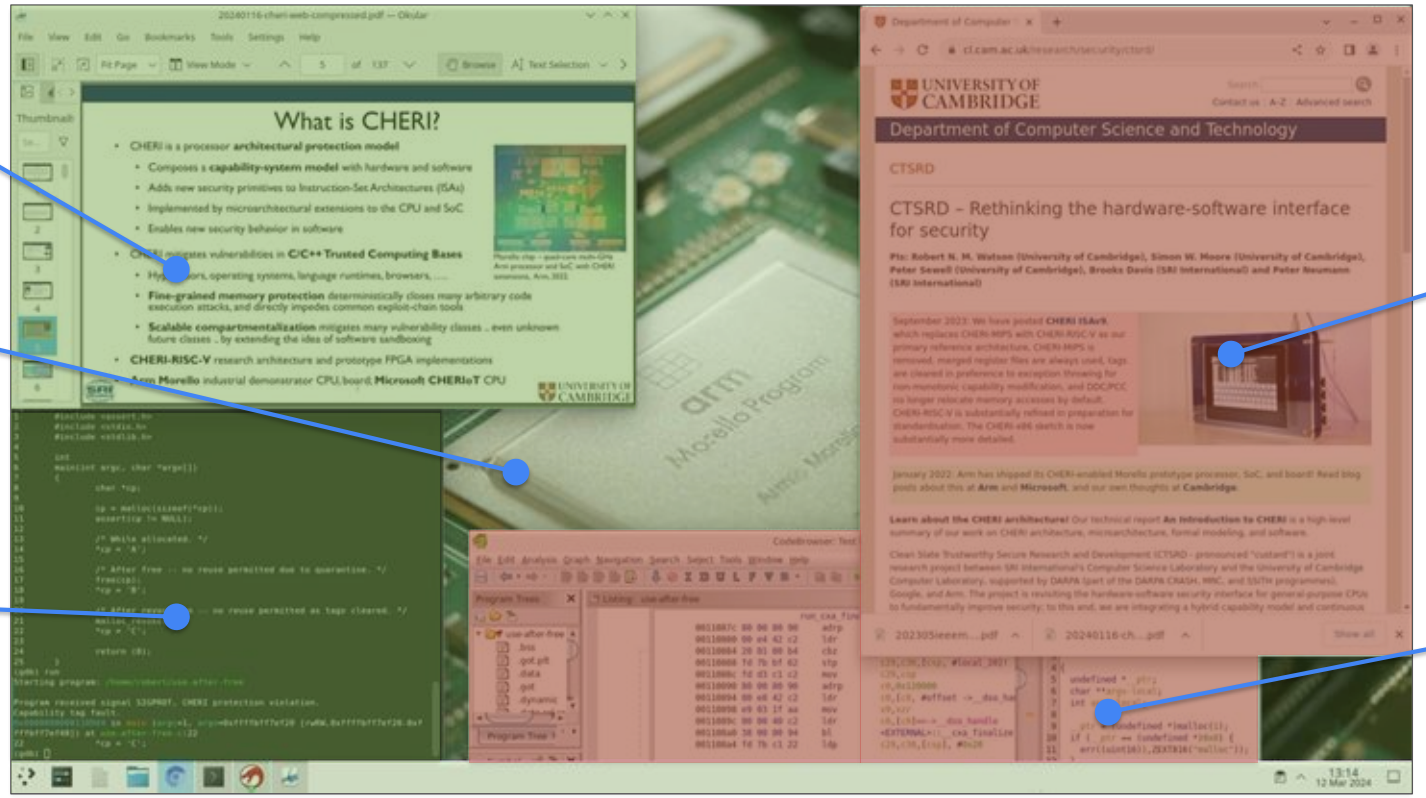
Enables incremental application migration to memory safety

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

The evidence: deterministic mitigation

≥ ⅔

of Microsoft's 2019 critical memory-safety vulnerabilities deterministically mitigated (MSRC study)

14 / 14

injected AIxCC nginx vulnerabilities mitigated – plus two extra bugs CHERI uncovered

0 LoC

changed in libwebp to stop CVE-2023-4863 "BLASTPASS" in CHERI Chromium

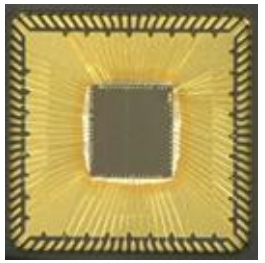
CHERI stops vulnerabilities by class, without knowing where the bug is

Silicon is arriving

Product	Market	µarch	MMU	Language	Tape-out announced	License
Cambridge CHERI-Toooba	Research application core	Out-of-order	Yes	Bluespec SystemVerilog	No	Open source
CapLtd CVA6-CHERI	Commercial application core	In-order	Yes	SystemVerilog	No	Open source
Codasip X730	Commercial application core	In-order	Yes	SystemVerilog	No	Proprietary
SECQAI CHERI-Ibex	Commercial microcontroller	In-order	No	SystemVerilog	Yes (test chips!)	Open source
Codasip L730	Commercial microcontroller	In-order	No	SystemVerilog	Yes	Proprietary
Microsoft / SCI / lowRISC CHERIoT-Ibex	Commercial microcontroller	In-order	No	SystemVerilog	Yes (test chips!)	Open source



SECQAI CHERI-Ibex, dual-core lockstep 22nm TPM chips (Oct 2025)



SCI Semi CHERIoT-Ibex, single-core 22nm IoT ICENI chips (Feb 2026)

- **March 2026:** first CHERI Alliance–certified cores – Microsoft CHERIoT-Ibex and Codasip X730

Next milestones to mass-market adoption



Ratified RVY

Stable ISA for 32- and 64-bit; certified IP from multiple vendors

Embedded products

Microcontrollers, roots of trust, TPMs and secure enclaves shipping

Across the SoC

Accelerators too – CHERI is on Google's Coral NPU roadmap

Open ecosystems

Application cores with upstream compilers, Linux, FreeBSD and browsers

Demand signal

ETSI memory-safety standard; certification; procurement and regulation