CHERI Update

Robert N. M. Watson, Simon W. Moore, Peter Sewell, Peter G. Neumann

Hesham Almatary, Ricardo de Oliveira Almeida, Jonathan Anderson, Alasdair Armstrong, Rosie Baish, Peter Blandford-Baker, John Baldwin, Hadrien Barrel, Thomas Bauereiss, Ruslan Bukin, Brian Campbell, David Chisnall, Jessica Clarke, Nirav Dave, Brooks Davis, Lawrence Esswood, Nathaniel W. Filardo, Franz Fuchs, Dapeng Gao, Ivan xGomes-Ribeiro, Khilan Gudka, Brett Gutstein, Angus Hammond, Graeme Jenkinson, Alexandre Joannou, Mark Johnston, Robert Kovacsics, Ben Laurie, A. Theo Markettos, J. Edward Maste, Alfredo Mazzinghi, Alan Mujumdar, Prashanth Mundkur, Steven J. Murdoch, Edward Napierala, George Neville-Neil, Kyndylan Nienhuis, Robert Norton-Wright, Philip Paeps, Lucian Paul-Trifu, Allison Randal, Ivan Ribeiro, Alex Richardson, Michael Roe, Colin Rothwell, Peter Rugg, Hassen Saidi, Thomas Sewell, Stacey Son, Ian Stark, Domagoj Stolfa, Andrew Turner, Munraj Vadera, Konrad Witaszczyk, Jonathan Woodruff, Hongyan Xia, Vadim Zaliva, and Bjoern A. Zeeb

University of Cambridge and SRI International

DSbD All Hands Meeting – Virtual Session – 11 October 2022

Approved for public release; distribution is unlimited. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts FA8750-10-C-0237 ("CTSRD"), with additional support from FA8750-11-C-0249 ("MRC2"), HR0011-18-C-0016 ("ECATS"), FA8650-18-C-7809 ("CIFV"), and HR001122C0110 ("ETC") . The views, opinions, and/or findings contained in this article/presentation are those of the author(s)/presenter(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Approved for public release; distribution is unlimited.

This work was supported in part by the Innovate UK project Digital Security by Design (DSbD) Technology Platform Prototype, 105694.

This work was also 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"), and HR001122C0110 ("ETC") as part of the DARPA CRASH, MRC, and SSITH 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 of the Department of Defense or the U.S. Government.

We further acknowledge the EPSRC REMS Programme Grant (EP/K008528/1), the ERC ELVER Advanced Grant (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, and the Gates Cambridge Trust.

CHERI introduction

- **CHERI is a new processor technology that mitigates software security vulnerabilities**
  - Developed by the University of Cambridge and SRI International starting in 2010, supported by DARPA
  - Arm collaboration from 2014
  - Arm Morello CPU, SoC, and board announced 2019, with support from UKRI; shipping as of Jan 2022
- Today's talk:
  - Microsoft's CHERI Ibex working
  - CHERI software since the last DSbD All Hands Meeting
- http://www.cheri-cpu.org/

An early experimental FPGA-based CHERI tablet prototype running the CheriBSD operating system and applications, Cambridge, 2013.

High-performance Arm Morello chip able to run a full CHERI software stack, Cambridge, 2022

CHERI-RISC-V

What's the smallest variety of CHERI?

Microsoft Security Response Center

Report an issu

What's the smallest variety of CHERI?

Security Research & Defense / By Saar Amar / September 6, 2022

The Portmeirion project is a collaboration between Microsoft Research Cambridge, Microsoft Security Response Center, and Azure Silicon Engineering & Solutions. Over the past year, we have been exploring how to scale the key ideas from CHERI down to tiny cores on the scale of the cheapest microcontrollers. These cores are very different from the desktop and server-class processors that have been the focus of the Morello project.

Microcontrollers are still typically in-order systems with short pipelines and tens to hundreds of kilobytes of local SRAM. In contrast, systems such as Morello have wide and deep pipelines, perform out-of-order execution, and have gigabytes to terabytes of DRAM hidden behind layers of caches and a memory management unit with multiple levels of page tables. There are billions of microcontrollers in the world and they are increasingly likely to be connected to the Internet. The lack of virtual memory means that they typically don't have any kind of process-like abstraction and so run unsafe languages in a single privilege domain.

This project has now reached the stage where we have a working RTOS running existing C/C++ components in compartments. We will be open sourcing the software stack over the coming months and are working to verify a production-quality implementation of our proposed ISA extension based on the lowRISC project's Ibex core, which we intend to contribute back upstream.

- Production-quality CHERI-RISC-V-extended Ibex core
  - Small-scale microcontroller (baseline CPU is used in OpenTitan)
  - CHERI-RISC-V specialized to small microcontrollers
  - Memory-safe, compartmentalized OS
  - Open-source hardware and software
  - RISC-V embedded standardization candidate
- Microsoft Research, MSRC, Azure Silicon, and Azure Edge + Platform

https://msrc-blog.microsoft.com/2022/09/06/whats-the-smallest-variety-of-cheri/

RISC-V CHERI Special Interest Group (SIG)

- Created in early October 2022
  - SIG acting chair is Alex Richardson (Google)
  - Build interest and consensus around CHERI-RISC-V standardization
- Likely at least two closely coupled standardization efforts (WIP plan):
  - Microcontroller 32-bit CHERI-RISC-V building on Microsoft's work
  - 64-bit CHERI-RISC-V building on SRI/Cambridge's ISA
- We expect CHERI ISAv9 and the forthcoming Microsoft technical report to be direct inputs to this process
- Lots of open questions -- e.g., do we need multiple working groups, how to we best capture the commonalities of the two ISA encodings, etc.

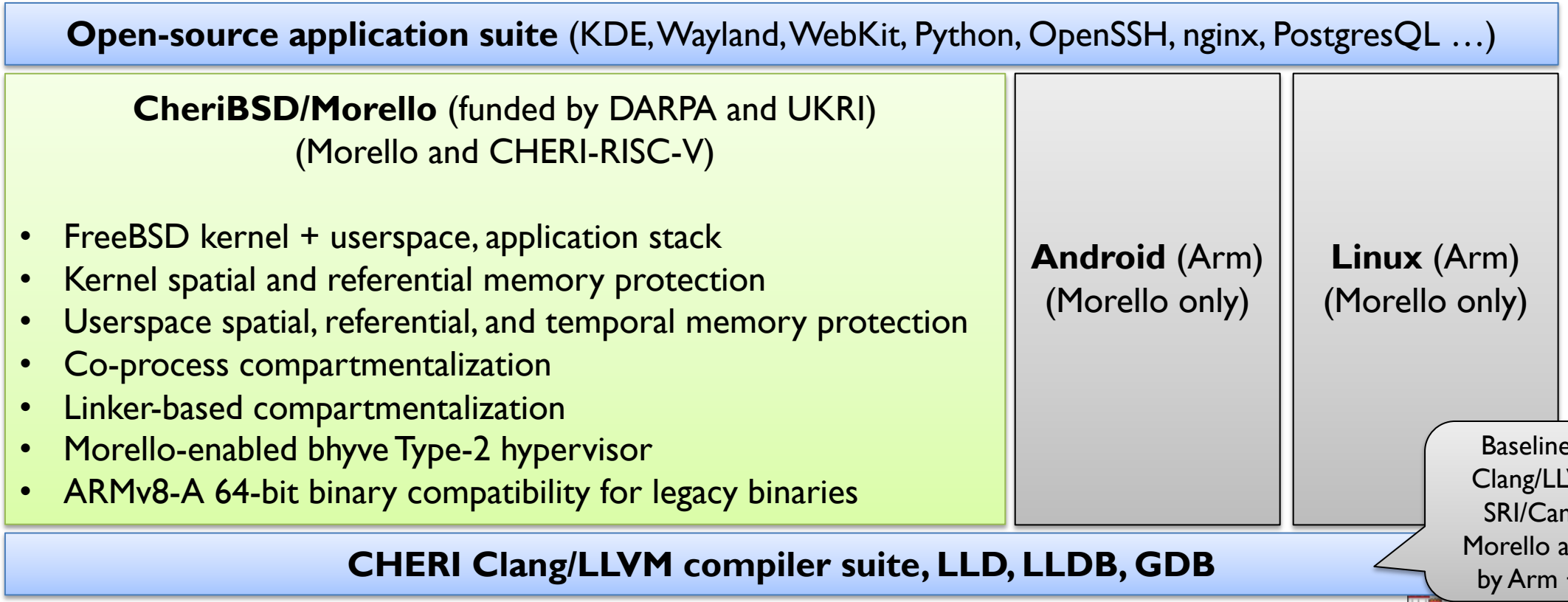
CHERI REFERENCE SOFTWARE STACK

Why port the CHERI stack to Morello?

- **Validate** the Morello architecture (functional, sufficient)
- **Evaluate** the Morello implementation (performance, energy use, …)
- **Provide reference software semantics** (spatial and temporal safety, compartmentalization, POSIX integration, OS kernel use, …) that will be applicable to other adaptations
- **Act as a template and prototyping platform** for at-scale industrial and academic demonstration, including providing adaptations of common software dependencies (e.g., widely used libraries)
- **Provide a platform for future software research,** asking questions about what we can use CHERI for in {operating systems, compilers, language runtimes, applications, …}
- **Enable a growing academic and industrial community** around CHERI and Morello, including dozens of UK universities and companies associated with DSbD

CHERI prototype software stack on Morello

- **Complete open-source software stack** from bare metal up: compilers, toolchain, debuggers, hypervisor, OS, applications – all demonstrating CHERI
- Establish "best practice" software semantics when using CHERI
- Rich CHERI feature use, but fundamentally incremental/hybridized deployment



CheriBSD software releases this year

- 22.05 release – First release supporting the Morello board
  - Memory-safe kernel and userspace ("CheriABI")
  - USB stick installation
  - Aarch64 and CheriABI third-party packages
  - "Getting Started with CheriBSD"
- 22.05p1 release – Patch release
  - Work around firmware bug to enable software reboot
  - Install debug symbols for libraries
  - Fix software crashes with thread-local storage (TLS)
- 22.10 (11?) release – Second major software release (details next slide)

Some of our in-flight software R&D efforts

Feature	Status	Availability
Morello GPU device drivers	**Hybrid + pure-capability kernel driver Hybrid + pure-capability user driver Hybrid + pure-capability applications**	**Autumn 2022**
Linker-based compartmentalization	**Prototype runs some UNIX applications; limited debugger support**	**Autumn 2022 as (highly) experimental feature**
Userlevel heap temporal safety	**Prototype runs SPEC benchmarks**	**Autumn 2022 (development branch), but "plug-in" to release**
bhyve (Type-2) hypervisor	Prototype boots pure-capability guest OS, but much more testing + review required	Autumn (development branch)
Co-process compartmentalization	Prototype runs some compartmentalized software (e.g., OpenSSL); API co-design	Early 2023 (development branch)

Other forthcoming feaures in 22.10 include:

- Precompiled / easily installed CheriABI packages for Wayland memory-safe display server and KDE
- Morello GDB adaptation with significant functional improvements

MORELLO DESKTOP SUPPORT

CheriBSD Morello GPU and HDMI kernel support

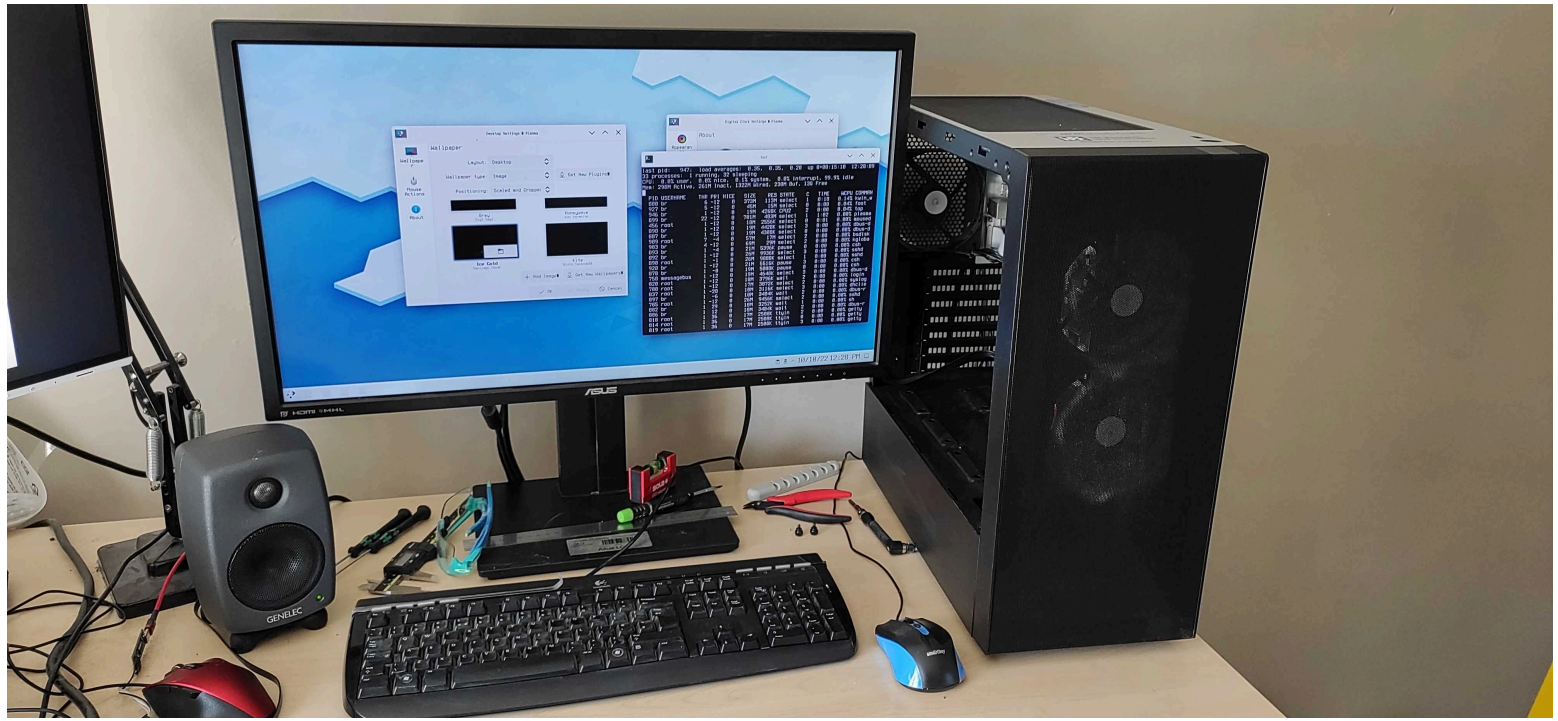
Quite significant engineering effort at Cambridge and CapLtd:

- Import + update Direct Rendering Manager (DRM) in FreeBSD
- Develop Arm IOMMU / SMMU driver for FreeBSD
- Develop Arm Komeda display controller driver
- Develop Arm Arm Mali GPU device driver
- Integrate HDMI controller device driver

General status:

- All compiles and runs with hybrid or memory-safe kernel and userlevel
- Shipping in CheriBSD Autumn 2022 release

Memory-safe Wayland and KDE



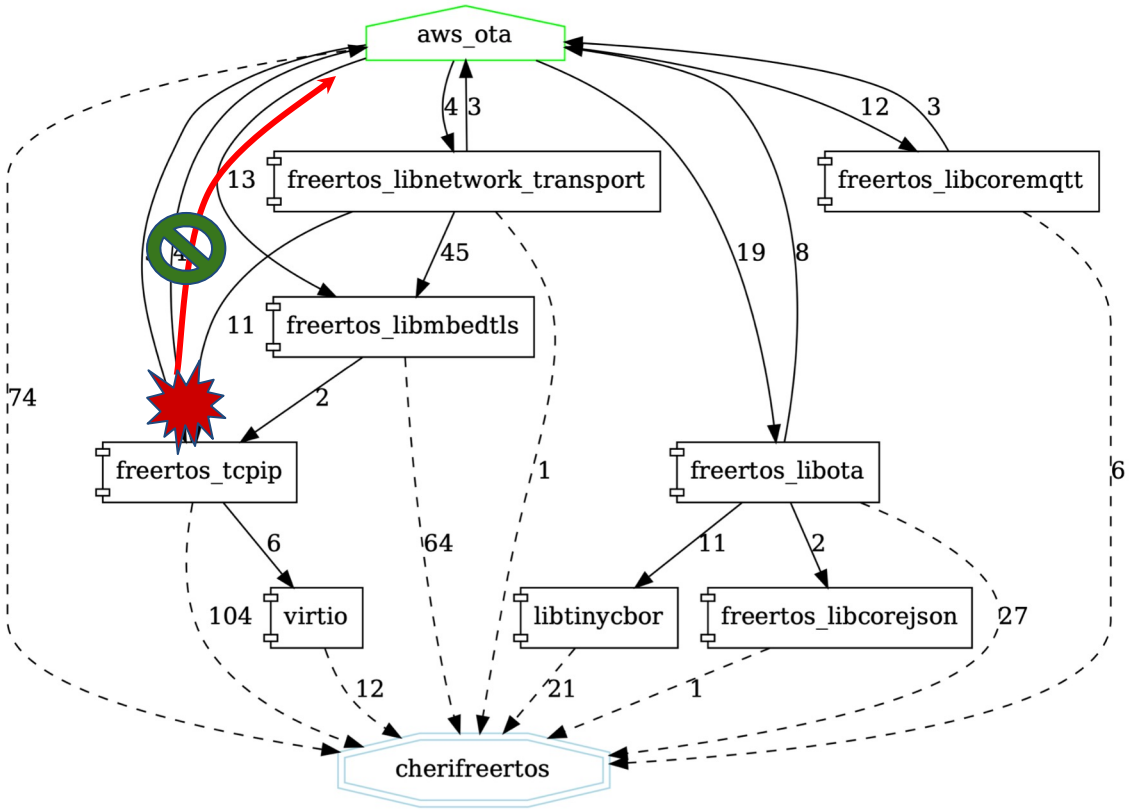
- Shifted from X11 to Wayland compared to earlier CapLtd work
- CheriABI Packages for Mesa, Wayland, Qt, KDE, …
- Shipping in CheriBSD Autumn 2022 release

Now on to the secure desktop grand challenges

- We are now within reach of an exciting – and historically highly vulnerable – application corpus to which we can apply CHERI protections
- Memory-safe desktop applications at scale – especially those that contain one or more language runtimes:
  - Web browsers
  - Mail readers
  - Office suites
- Extending this to fine-grained compartmentalization as software prototypes mature – library compartmentalization, coprocesses, further models, …
- For example: UKRI- and Google-funded efforts around the Chromium web browser at CapLtd, Kings College London, Arm, and Cambridge

EXPERIMENTAL LIBRARY COMPARTMENTALISATION SUPPORT

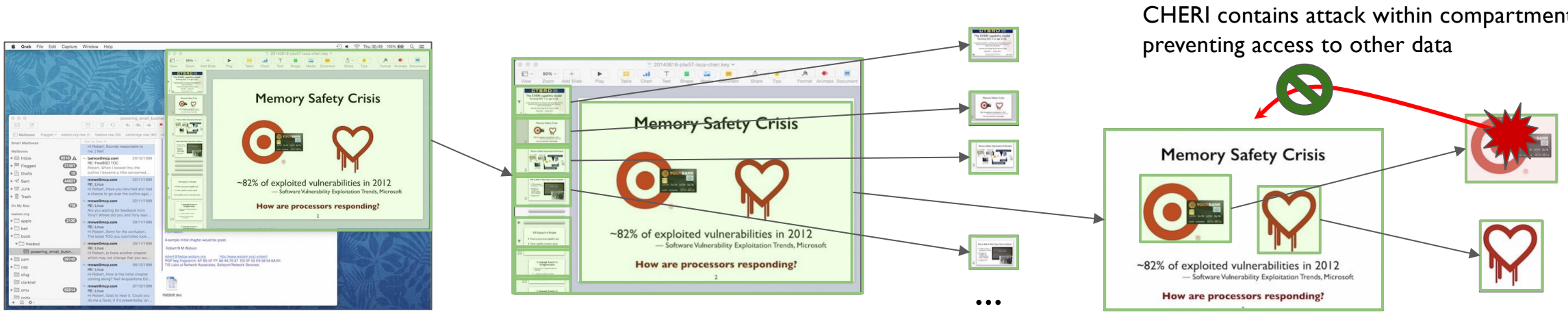
What is software compartmentalization?



CheriFreeRTOS components and the application execute in compartments. CHERI contains an attack within TCP/IP compartment, which access neither flash nor the internals of the software update (OTA) compartment.

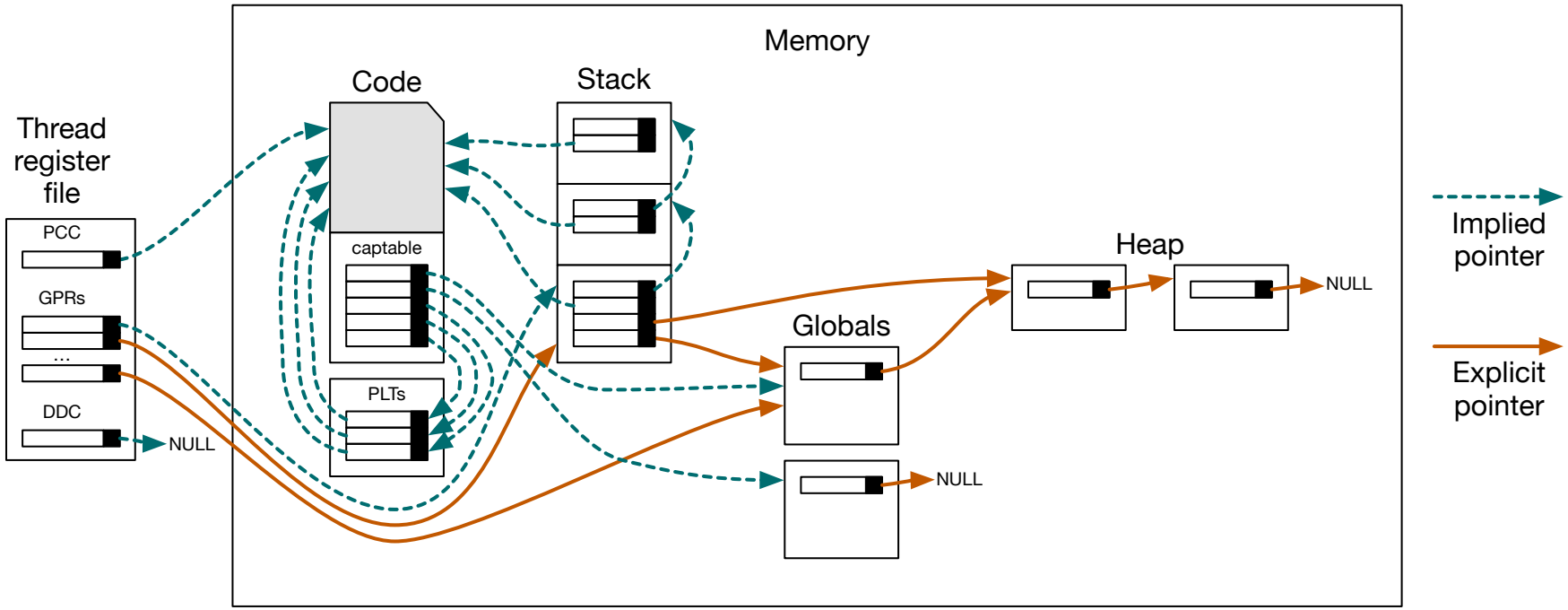
- Fine-grained decomposition of a larger software system into **isolated modules** to constrain the impact of faults or attacks
- Goals is to **minimize privileges yielded by a successful attack, and to limit further attack surfaces**
- Usefully thought about as a **graph of interconnected components**, where the attacker's goal is to compromise nodes of the graph providing a route from a point of entry to a specific target

Software compartmentalization at scale



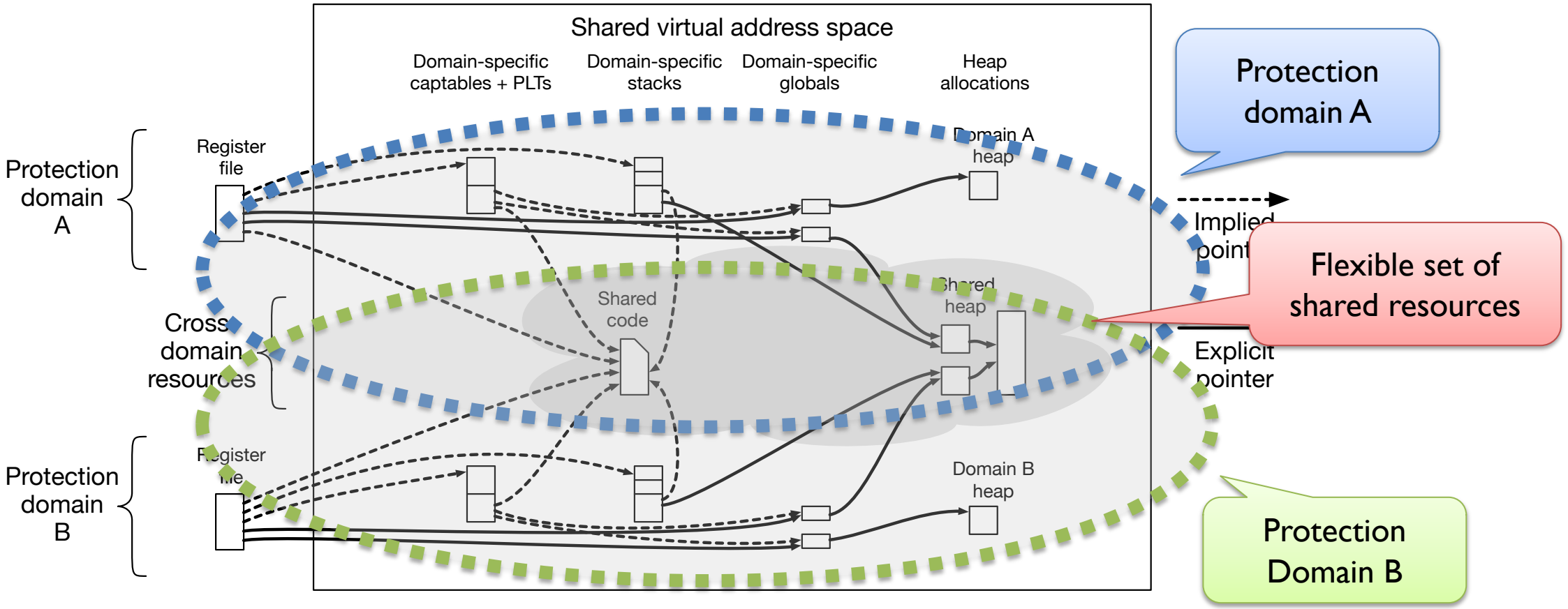
- Current CPUs limit:
  - The number of compartments and rate of their creation/destruction
  - The frequency of switching between them, especially as compartment count grows
  - The nature and performance of memory sharing between compartments
- CHERI is intended to improve each of these – by at least an order of magnitude

CHERI-based pure-capability process memory



- Capabilities are substituted for integer addresses throughout the address space
- Bounds and permissions are minimized by software including the kernel, run-time linker, memory allocator, and compiler-generated code
- Hardware permits fetch, load, and store only through granted capabilities
- Tags ensure integrity and provenance validity of all pointers

CHERI-based compartmentalization



- Isolated compartments can be created using closed graphs of capabilities, combined with a constrained non-monotonic domain-transition mechanism

Compartmentalization scalability

- CHERI dramatically improves **compartmentalization scalability**
  - More compartments
  - More frequent and faster domain transitions
  - Faster shared memory between compartments

Early benchmarks show a 1-to-2 order of magnitude performance inter-compartment communication improvement compared to conventional designs

- Many potential use cases – e.g., sandbox processing of each image in a web browser, processing each message in a mail application
- Unlike memory protection, software compartmentalization requires **careful software refactoring** to support strong encapsulation, and affects the software operational model

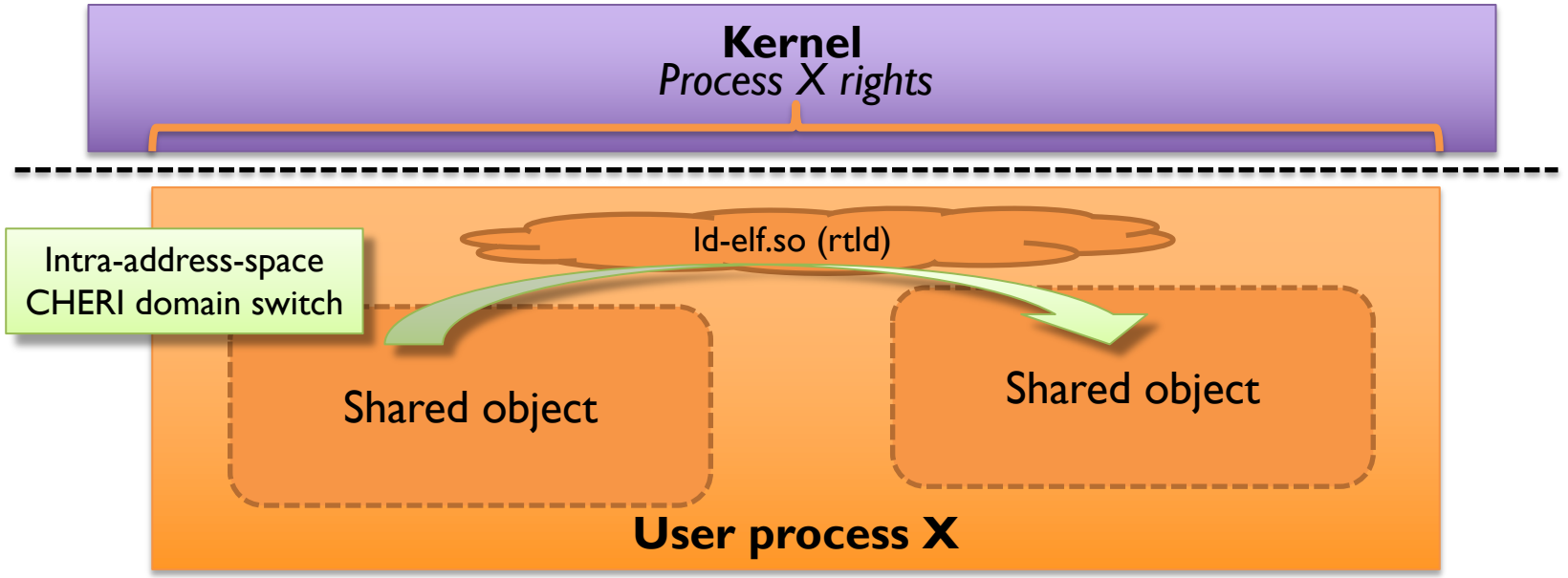
Operational models for CHERI compartmentalization

- CHERI is an **architectural protection model** enabling new software behavior
- As with virtual memory, multiple **software operational models** can be supported
  - E.g., with an MMU: Microkernels, processes, virtual machines, etc.
  - How are compartments created/destroyed? Function calls vs. message passing? Signaling, debugging, …?
- We have explored multiple viable CHERI-based models to date, including:

  Isolated dynamic libraries Efficient but simple sandboxing in processes

  UNIX co-processes Multiple processes share an address space
- Improved performance and new paradigms using CHERI primitives
- Both will be available in CheriBSD/Morello

Sandboxed shared libraries



- Shared libraries execute within "sandboxes":
  - Libraries have access only to explicitly linked or dynamically delegated resources
  - CHERI used to isolate compartments within a shared address space
- Compartmentalisation-aware run-time linker (rtld), libc, libthr (pthreads)
  - rtld injects domain-switching call and return shims during PLT linkage
- Lots of subtlety around topics like threads, signals, system calls, setjmp/longjmp, etc.

Sandboxed library prototype in CheriBSD

- Early prototype implementation intended to make work more broadly available, especially for related research projects, but be aware / beware:
  - Incomplete implementation
  - Unstable APIs
  - Depends on new and forthcoming compiler ABI changes
  - No security claims [yet]
- Opt in: Programs will need to specify the alternative run-time linker in their ELF header, set during compilation
- Shipping in CheriBSD Autumn 2022 release
- Intention is that the design principles (and APIs?) will be fully applicable on Android / Linux

CONCLUSION

Conclusion

- Extremely active research and engineering around CHERI
  - Creation of a CHERI-RISC-V SIG with the aim of standardizing
  - Microsoft CHERI-RISC-V work and (soon) open-sourced Ibex core
- Morello software stack rapidly maturing
  - Memory-safe GPU and early desktop environment
  - Library compartmentalization model
  - Easier use of experimental temporal memory safety work
  - Many other ongoing activities including around co-process compartmentalization, an adaptation of the Chromium browser
- CHERI software adaptation workshop at DSbD All Hands Meeting