CheriABI

Enforcing Valid Pointer Provenance and Minimizing Pointer Privilege in the POSIX C Run-time Environment

Brooks Davis, Robert N. M. Watson, Alexander Richardson,
Peter G. Neumann, Simon W. Moore, John Baldwin, David Chisnall,
James Clarke, Nathaniel Wesley Filardo, Khilan Gudka, Alexandre Joannou, Ben Laurie, A. Theodore Markettos, J. Edward Maste, Alfredo Mazzinghi,
Edward Napierala, Robert Norton, Michael Roe, Peter Sewell, Stacey Son,
Jonathan Woodruff

SRI International, University of Cambridge, Microsoft Research, Google, Inc

Approved for public release; distribution is unlimited. This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts FA8750-10-C-0237 ("CTSRD") and HR0011-18-C-0016 ("ECATS"). 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.

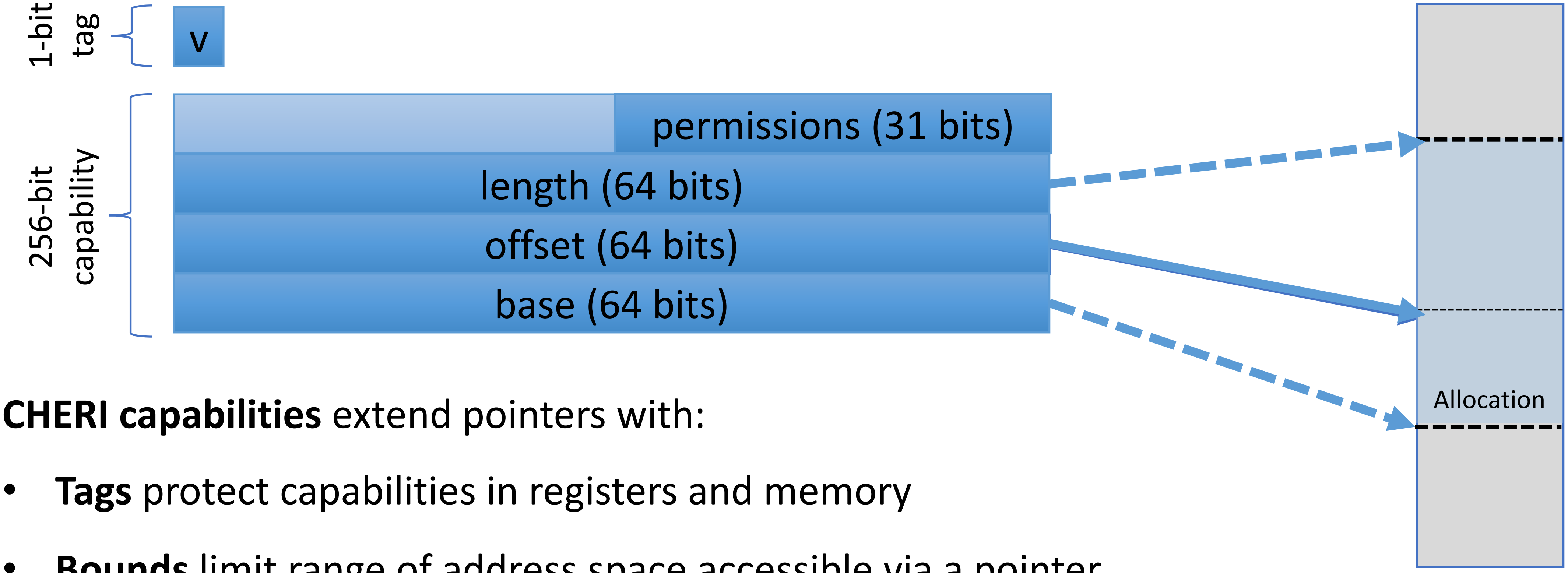
Introduction to CHERI

- CHERI introduces a new register type: the **capability**
  - In addition to integer and floating point
- CHERI capabilities grant access to bounded regions of virtual address space
  - Protected by tags

Watson, et al. **CHERI: a research platform deconflating hardware virtualization and protection.** RESoLVE 2012.

Woodruff, et al. **The CHERI capability model: Revisiting RISC in an age of risk**. ISCA 2014.

Architectural CHERI capabilities



CHERI capabilities extend pointers with:

- **Tags** protect capabilities in registers and memory
- **Bounds** limit range of address space accessible via a pointer
- **Permissions** limit operations – e.g., load, store, instruction fetch

Actual implementation is compressed to **128-bits** with floating-point bounds

Virtual address space

CHERI Operation

- All memory access via capabilities
  - Explicit (new instructions):
    - Capability load, store, branch, jump
  - Implicit (legacy MIPS ISA):
    - via Default Data Capability (DDC) or Program Counter Capability (PCC)
- Capabilities are used and manipulated in capability registers with capability instructions
  - Manipulations are monotonic (can only reduce bounds and permissions)
- Capabilities can be stored in memory, protected by tags

Capabilities as C pointers

- CHERI capabilities are designed for use as C pointers
  - Allowed to be out of bounds between dereferences
  - Can store 64-bit integers (untagged)
- Two compilation modes:
  - Hybrid: `__capability` annotation applied to select pointers
  - Pure-capability: all pointers are capabilities

Chisnall, et al. **Beyond the PDP-11: Processor support for a memory-safe C abstract machine.** ASPLOS 2015.

CheriABI: Pure-capability process environment

- Built on CheriBSD (FreeBSD modified for CHERI)
- All pointers are capabilities
  - Including syscall arguments and return values
- Bounds are minimized
  - C-language objects
  - Pointers provided by the kernel
- Goal: run pure-capability programs with simple recompilation

Watson, et al. **CHERI: A Hybrid Capability-System Architecture for Scalable Software Compartmentalization.** Oakland 2015.

Chisnall, et al. **CHERI-JNI: Sinking the Java security model into the C.** ASPLOS 2017.

Abstract capabilities

How should the systems programmer **think** about bounds?

New concept: *abstract capability*

- Set of permissions of the process
- Tracks ghost state across swapping, etc
- Constructed and maintained by a collaboration of the kernel and language runtime

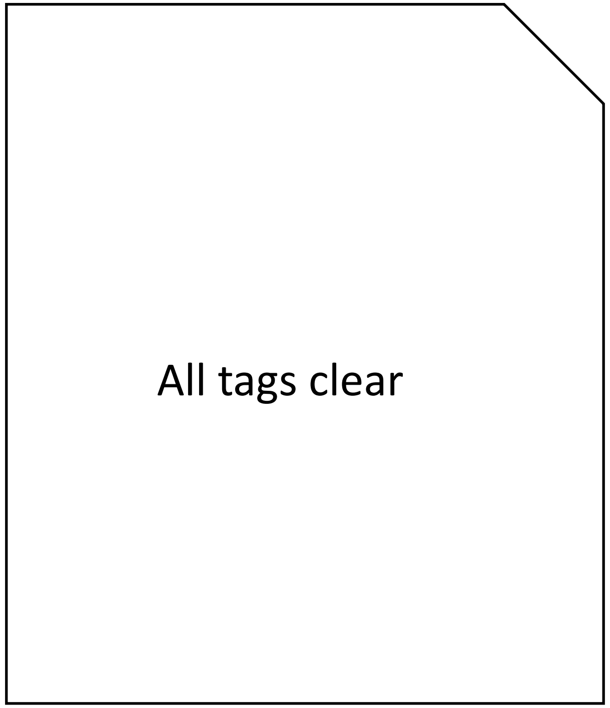
System startup

Power-on state

Registers

DDC	RWX	0x0	-	0xFF…FF
PCC	RWX	0x0	-	0xFF…FF
C1-31	NULL			

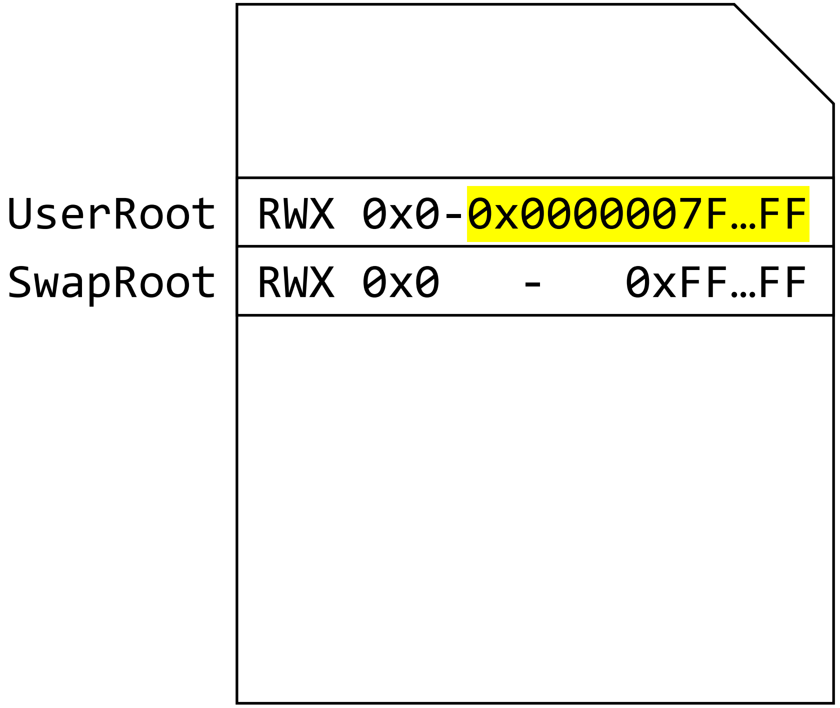
Memory



All tags clear

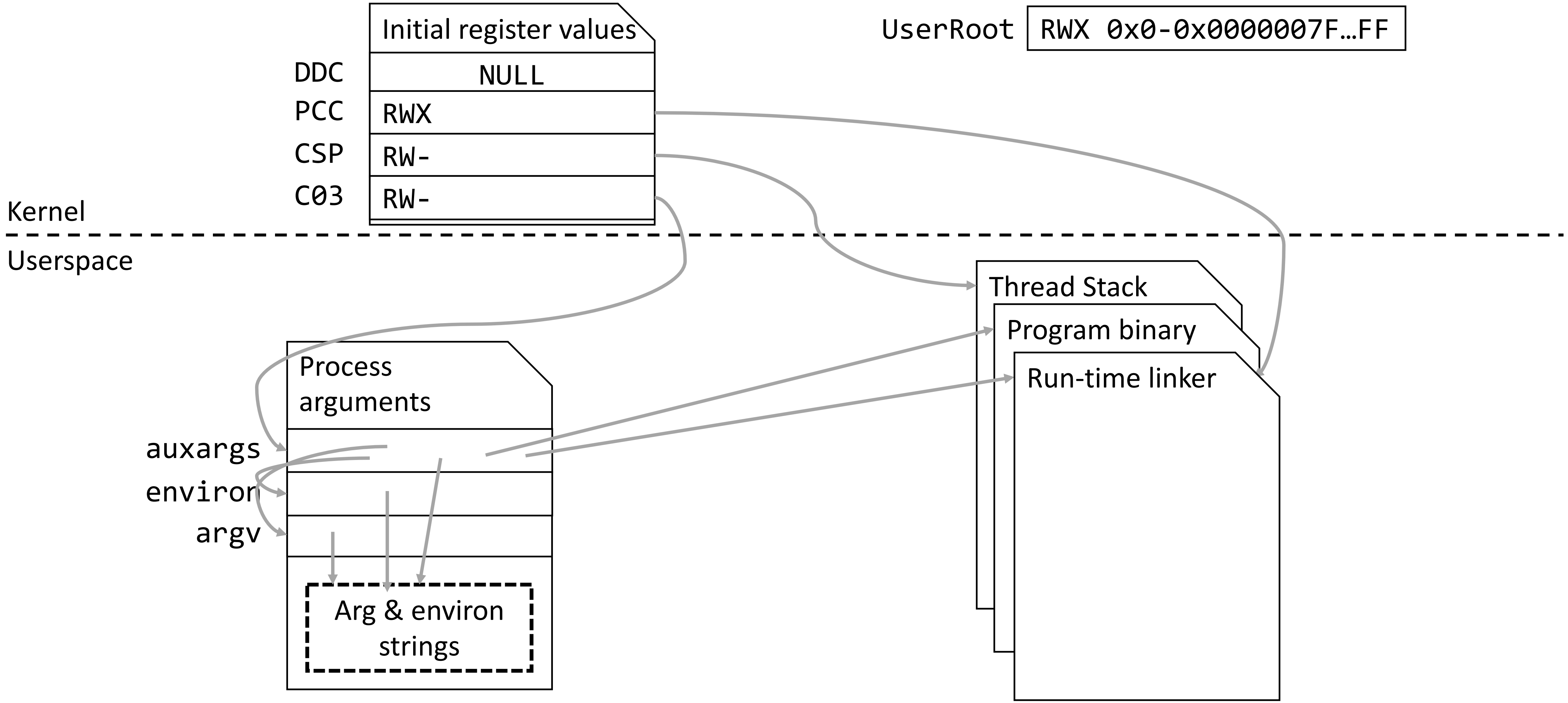
Early boot

DDC	RW-	0x0	-	0xFF…FF
PCC	R-X	0x0	-	0xFF…FF
C1-31	*Working set*			



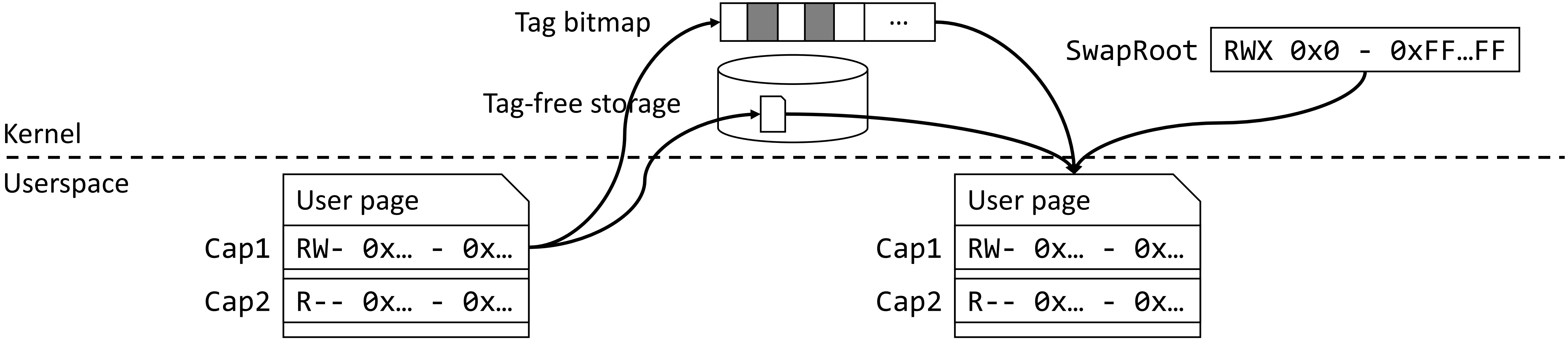
UserRoot	RWX	0x0-0x0000007F…FF
SwapRoot	RWX	0x0 - 0xFF…FF

Execve


Initial register values
DDC
NULL
PCC
RWX
CSP
RW-
C03
RW-
UserRoot
RWX 0x0-0x0000007F…FF
Kernel
Userspace
Thread Stack
Program binary
Run-time linker
Process arguments
auxargs
environ
argv
Arg & environ strings

Virtual-memory system

- Programmer visible:
  - Provides capabilities to newly mapped regions via `mmap()` and `shmat()`
  - Alters and frees mappings
- Abstract capability maintenance:
  - Ensures correct virtual to physical mappings
  - Preserves stored capabilities in swapped pages



Run-time linker

- Loads and links dynamic libraries
- Resolves symbols and synthesizes capabilities
- Jumps to program entry point

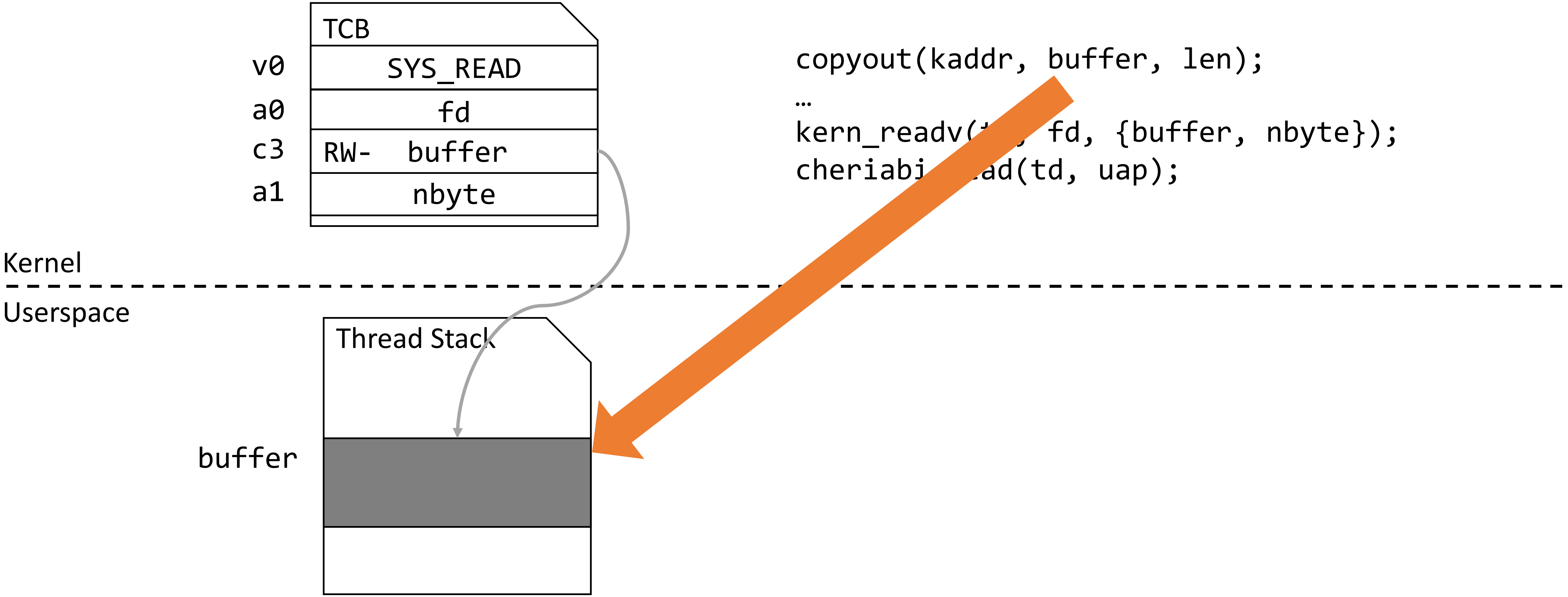
- Provides on-demand loading of libraries and supports exception handling

C runtime

- Objects allocated by `malloc()` are bounded to requested size
- `realloc()` adjusts bounds or allocates new storage as required
- Thread-local storage is bounded
  - Currently to per-thread storage
- Compiler generated code sets bounds on stack, automatic, and global objects

System calls

```
read(fd, buffer, nbyte);
```

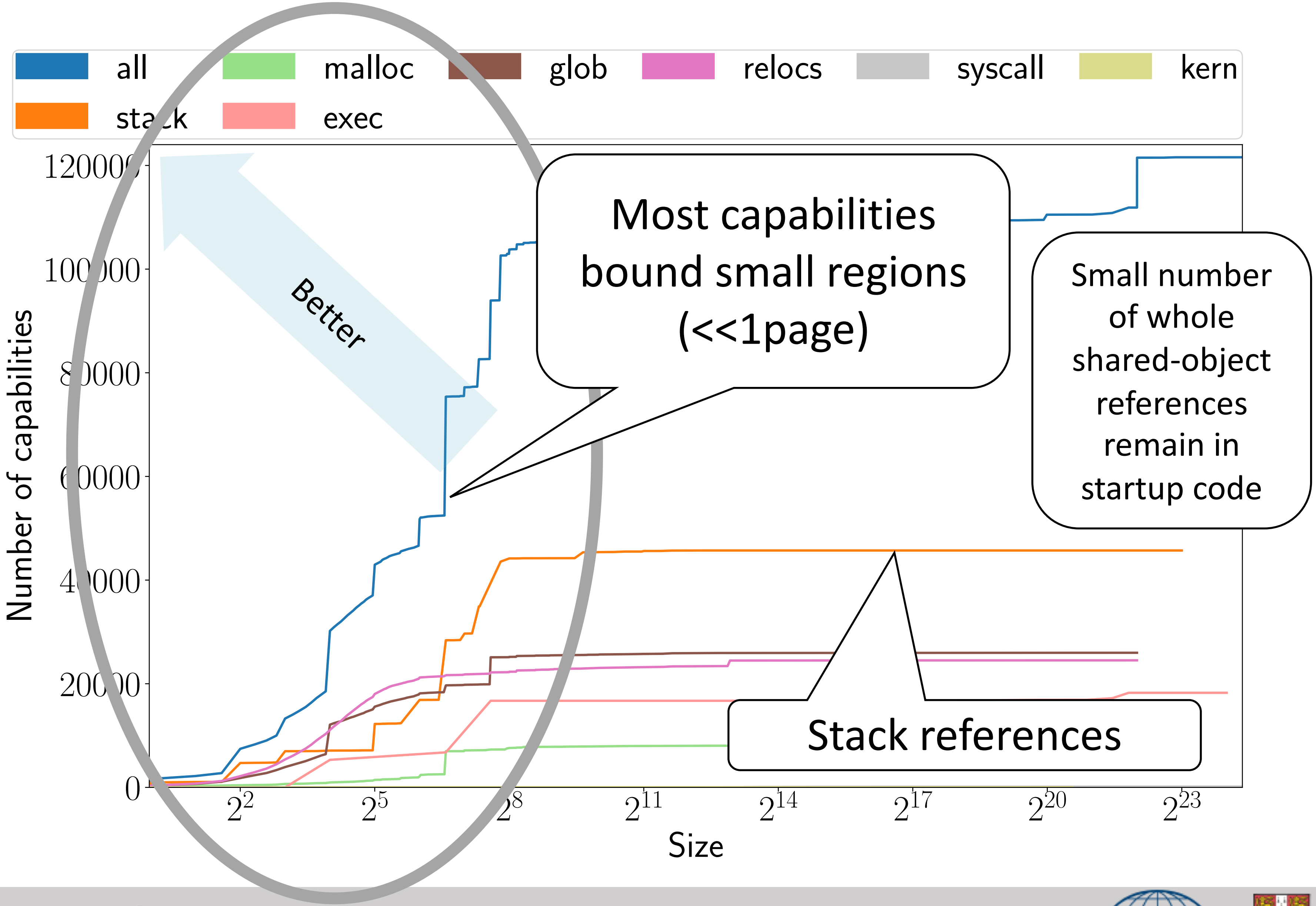


```
copyout(kaddr, buffer, len);
…
kern_readv(td, fd, {buffer, nbyte});
cheriabi_read(td, uap);
```

Required source code changes

- Userspace: 1% (~200) of files required changes
  - Concentrated in libraries
  - Most programs require no changes
- Kernel: <6% of files (~750) required changes
  - Pervasive changes to `iovec`, signal handlers, network interface `ioctl` handlers
  - A pure-capability kernel could reduce changes
- Many changes improve code quality
  - Upstreaming to FreeBSD and other projects often possible

Capability bounds minimization (OpenSSL)

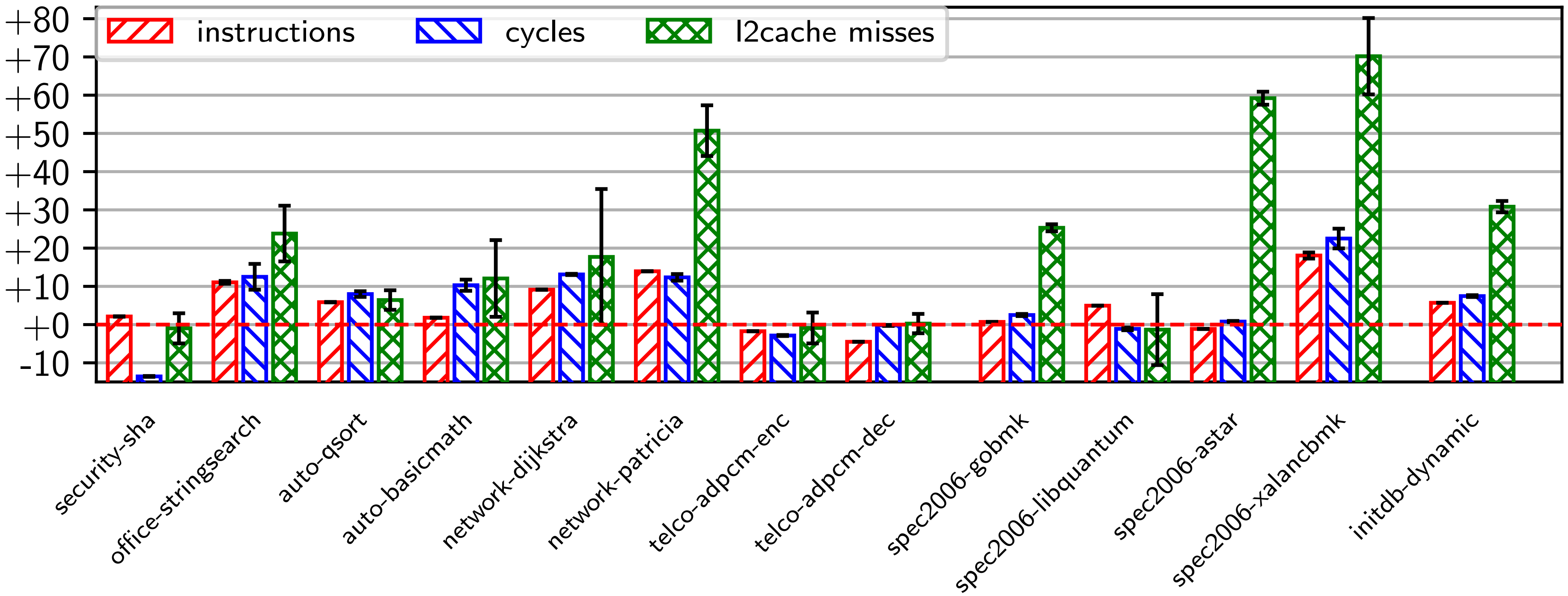


Most capabilities bound small regions (<<1page)

Small number of whole shared-object references remain in startup code

Stack references

Performance



- Micro-benchmark performance generally acceptable
  - <10% overhead in most cases
  - Graph excludes crypto and bit-manipulation outliers

Conclusions

- Full UNIX-like operating system with spatial and referential memory safety
  - Covers programs, libraries, and linkers
  - Kernel access to user memory
- Some fundamental operating system changes required
  - Generally non-disruptive
- 3rd-party software works: PostgreSQL database, Webkit

Further Reading

http://cheri-cpu.org/

Watson, et al., **Capability Hardware Enhanced RISC Instructions: CHERI Instruction-Set Architecture (Version 7),** Technical Report UCAM-CL-TR-927, Computer Laboratory, Cambridge UK, October 2018.

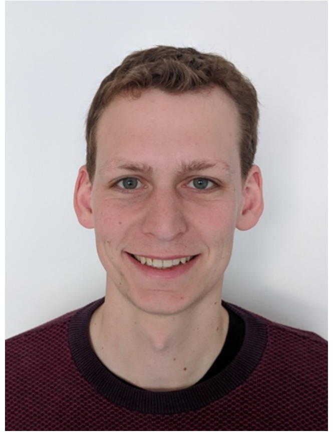
Davis, et al., **CheriABI: Enforcing Valid Pointer Provenance and Minimizing Pointer Privilege in the POSIX C Run-time Environment (Extended Version),** Technical Report UCAM-CL-TR-932, Computer Laboratory, Cambridge UK, January 2019.

Woodruff, et al., **CHERI Concentrate: Practical Compressed Capabilities,** IEEE Transactions on Computers, (forthcoming).

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA).
The views, opinions and/or findings expressed are those of the author 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.

Q & A

Brooks Davis
SRI International



Alexander Richardson
University of Cambridge

David Chisnall
Microsoft Research

Nathaniel Wesley Filardo
University of Cambridge