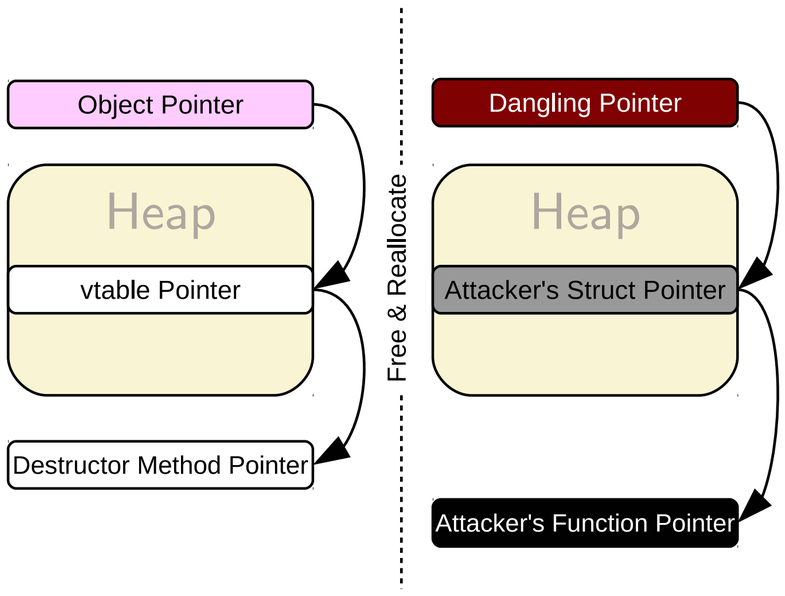
CHERIvoke: Characterising Pointer Revocation using CHERI Capabilities for Temporal Memory Safety

Hongyan Xia, Jonathan Woodruff, Sam Ainsworth, Nathaniel W. Filardo, Michael Roe, Alexander Richardson, Peter Rugg, Peter G. Neumann*, Simon W. Moore, Robert N. M. Watson and Timothy M. Jones

University of Cambridge, *SRI International

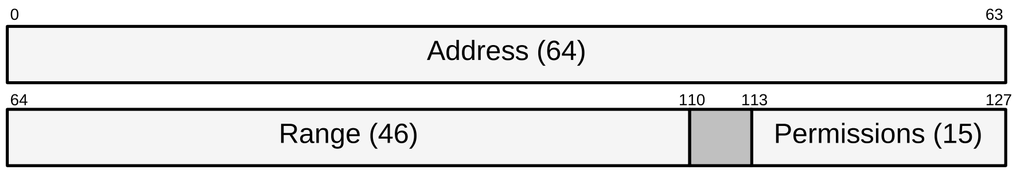
Use-After-Free Attacks



A lack of temporal safety in low-level languages has led to an epidemic of use-after-free exploits. These have surpassed in number and severity even the infamous buffer-overflow exploits violating spatial safety.

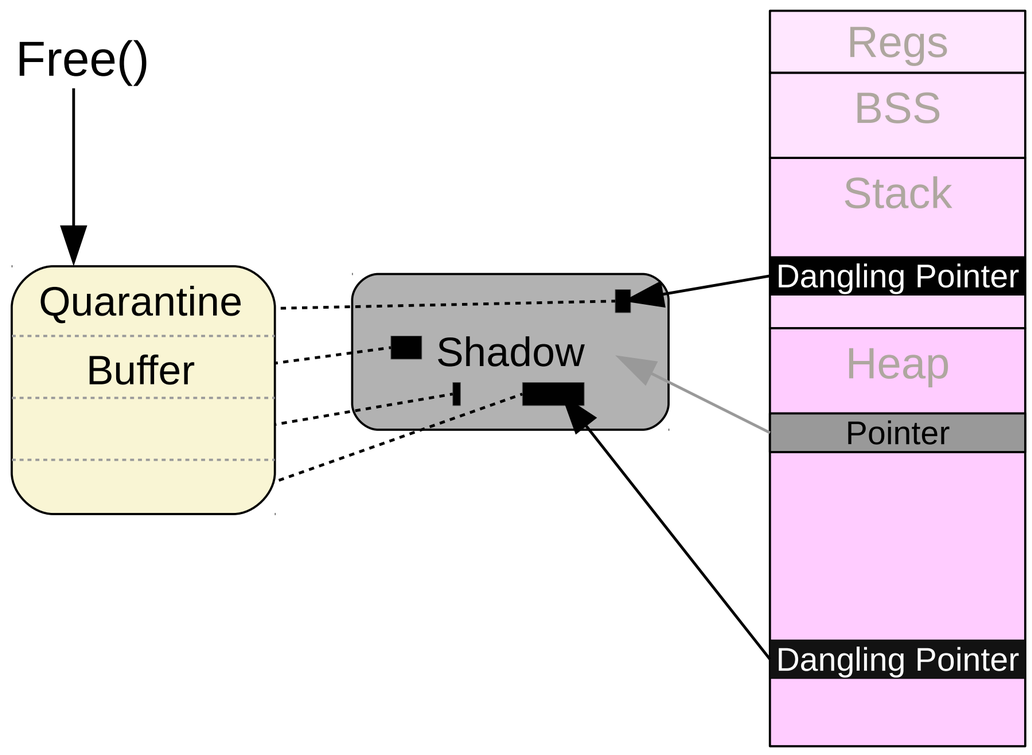
CHERI Capabilities

CHERI is an architectural extension to provide hardware capability addressing for *spatial* safety. It allows unique identification of pointers at the architectural level, with distinct access bounds, and identification of pointer-free cache lines and pages.



We develop CHERIvoke, a technique for deterministic and fast sweeping revocation to enforce *temporal* safety on CHERI systems. CHERIvoke quarantines freed data before periodically using a small shadow map to revoke all dangling pointers in a single sweep of memory.

CHERIvoke Algorithm

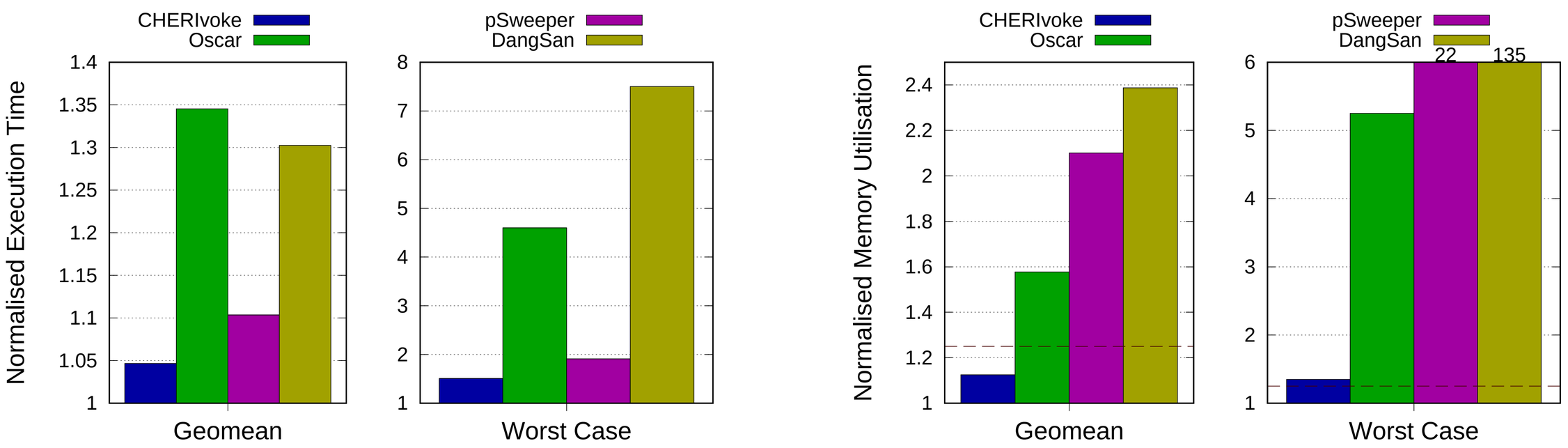


- Quarantine manual frees until we can clear dangling references.
- Fill 1/128-sized shadow region with "poison" bits, to tell us which capabilities to revoke.
- Fast sweep through memory region, to remove dangling capabilities based on shadow space and optimised to avoid capability-free pages and cache lines, once 25% of memory is in quarantine.

Acknowledgements

This work is part of the CTSRD and ECATS projects sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contracts FA8750-10-C-0237 and HR0011-18-C-0016. This work was also supported by the Engineering and Physical Sciences Research Council (EPSRC), through grant references EP/K026399/1, EP/P020011/1, and EP/K008528/1 and by Arm Limited and Google, Inc.

Low Overhead on Real Systems



We evaluate CHERIvoke using high-performance x86 processors, simulating the existence of capabilities. When configured with a heap-size overhead of 25%, we find that CHERIvoke achieves an average execution-time overhead of under 5%, far below the overheads associated with traditional garbage collection, revocation, or page-table systems.