PICASSO: Scaling CHERI Use-After-Free Protection to Millions of Allocations using Colored Capabilities

Merve Gülmez*, Ruben Sturm†, Hossam ElAtali‡, Håkan Englund*, Jonathan Woodruff§, N. Asokan‡,‖, Thomas Nyman¶
*Ericsson Security Research, †DistriNet, KU Leuven, ‡University of Waterloo,
§University of Cambridge, ‖KTH Royal Institute of Technology, ¶Ericsson Product Security

ARTIFACT EVALUATED USENIX AVAILABLE

ARTIFACT EVALUATED USENIX FUNCTIONAL

ARTIFACT EVALUATED USENIX REPRODUCED

https://github.com/coloredcapabilities

https://www.usenix.org/conference/usenixsecurity26/presentation/guelmez

Colored Capabilities introduce a limited form of indirection to CHERI which allows tracking allocation provenance (whether a capability corresponds to valid allocation)

Hardware support for **memory safety**, such as **CHERI**, is attractive for securing existing codebases in unsafe languages, e.g., C and C++

- CHERI replaces conventional pointers with **capabilities**

An ordinary "pointer" (reference to memory)

`0x7ffe204`

Just a number. Nothing records where it's allowed to point or what it's allowed to do – a bug can send it anywhere!

A CHERI capability consists of:

- **Address** – where it points to
- **Bounds** – the range it may touch
- **Permissions** – read / write / execute
- **Validity tag** – proof it hasn't been forged

On CHERI processors memory accesses checked by hardware to ensure they:

Stay within bounds (spatial safety)
A capability can only reach memory range it is valid for; out-of-bound accesses are refused

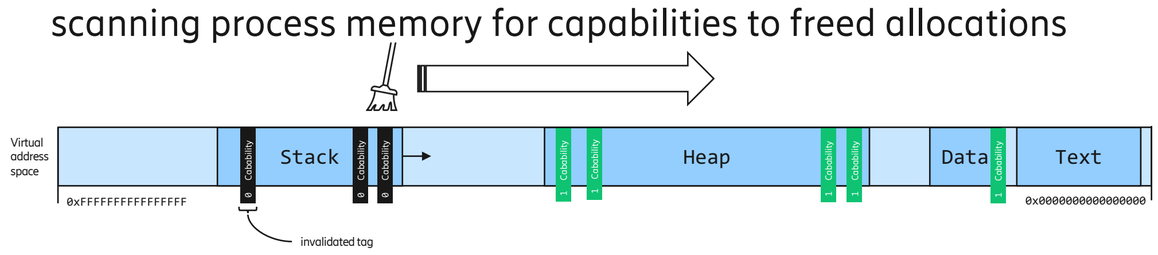
Can't be forged
Every capability is associated with a hardware tag; attempts to tamper with a fabricate a capability instantly invalidates it

Has the correct rights
Capabilities embody the principle of least privilege; can only be used in permitted operations

Any violation of these rules makes the hardware immediately block attempted memory access; buffer overflows, forged capabilities or out-of-bounds reads never succeed

To ensure **temporal safety** (memory not accessed outside lifespan) CHERI capabilities must be **revoked** when a heap allocation is freed

- Revocation requires knowing a capability's (and its validity tag's) location in memory and invalidating its tag
- CHERI temporal safety relies on **sweeping-based revocation:** scanning process memory for capabilities to freed allocations



The revocation sweep is **CHERI's Achilles' heel**; temporal-safety solutions for CHERI postpone revocation by quarantining freed memory


Image generated using AI

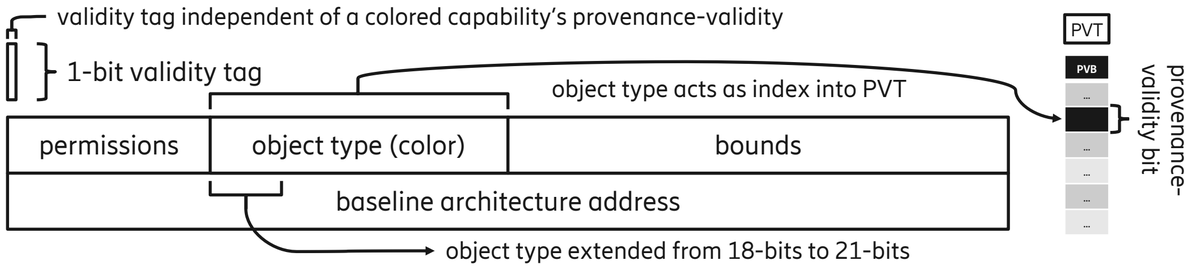
Cornucopia & Cornucopia Reloaded:

- Only address use-after-reallocation bugs: **use-after-free / use-after-reallocation gap**
- Quarantined allocations only reuseable after revocation: **add to memory overhead**
- Revocation frequency **scales** inversely with memory overhead

Filardo, N. W. et al. 2020. Cornucopia: Temporal Safety for CHERI Heaps. IEEE S&P '20.
Filardo, N. W. et al. 2024. Cornucopia Reloaded: Load Barriers for CHERI Heap Temporal Safety. ASPLOS '24

Colored Capabilities introduce a **limited form of indirection** to CHERI

- Track **allocation provenance** of capabilities: *whether* a capability corresponds to valid allocation (but not *which* allocation)
- Capabilities sharing provenance also share **provenance ID** ("color") stored in a capability's object type (otype) field
- Validity of ID's provenance tracked by 1-bit entry (PVB) in **provenance-validity table** (PVT)
- Capabilities with provenance IDs, but invalid provenance are **temporarily retracted** (rendered unusable, but not fully revoked)



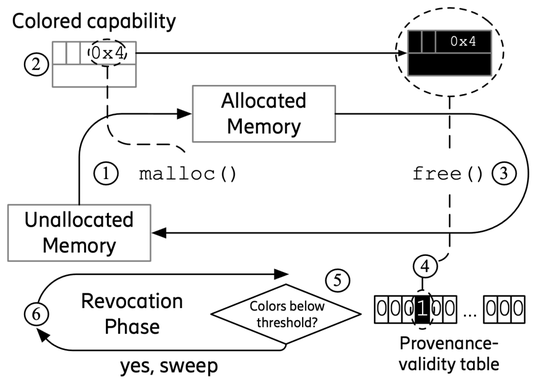
Colored Capabilities improve precision of CHERI's temporal safety:

Use After Reallocation (UAR)
Dereferencing dangling pointer after memory has been reallocated and used for another object

Use After Free (UAF)
Accessing memory through a dangling pointer after original allocation has been freed

Colored Capabilities:

+ Address the **UAF/UAR gap**
+ Allow **immediate reuse** of freed memory
+ Revocation frequency **scales with number of supported colors**



①, ② **Allocation**: assign unique provenance ID from unassigned color values to capability

③, ④ **Free**: Invalidate provenance ID's PVB in PVT

⑤, ⑥ **Unassigned colors below threshold**: Sweep memory and revoke capabilities with invalid PVB

We implement Colored Capabilities in PICASSO extension to CHERI ISA

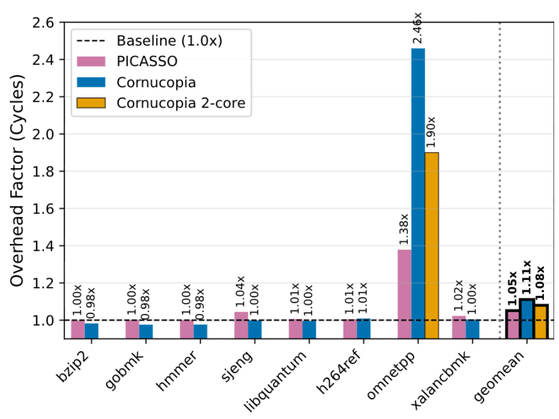
Hardware: Two versions of PICASSO provided in artifact
1. QEMU-system-CHERI128 full-system emulator, and
2. CHERI-Toooba processor with colored capability support

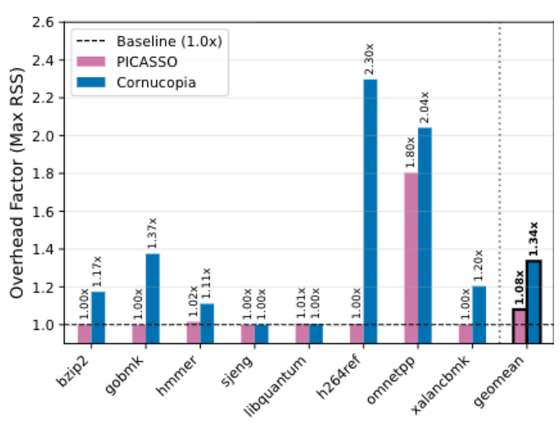
Software: Colored Capability support in CheriBSD 25.03
- Drop-in userspace support through CheriBSD malloc revocation shim (MRS)
- Userspace provenance-identifier management (unr allocator)
- Integration with CheriBSD in-kernel revocation service

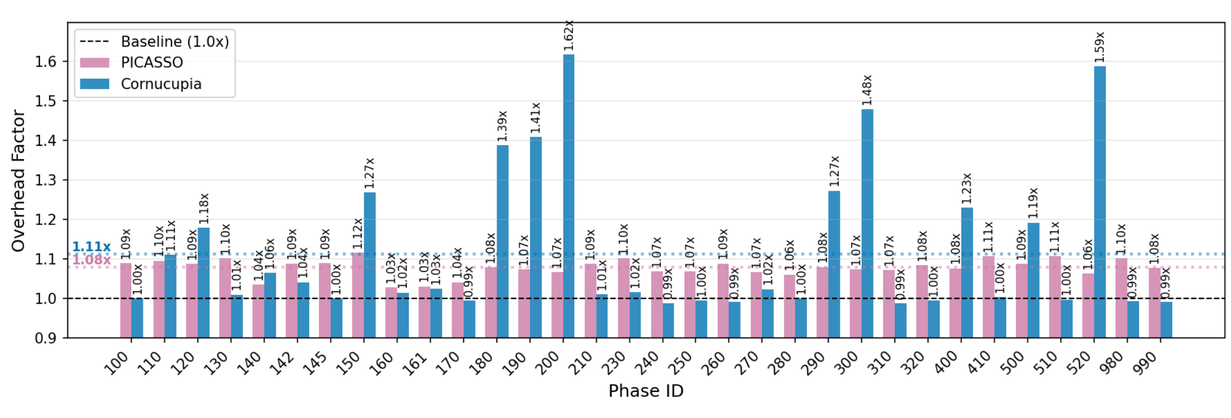
We evaluate PICASSO using Juliet Test Suite:
- All CWE-416 (Use After Free) test cases
- All CWE-415 (Double Free) test cases
- In total: 1385 "bad" test cases
- Equal number of "good" (patched) test cases

PICASSO detects **all bad cases** and **no false positives**

We compare our results against CheriBSD Cornucopia in two-different configurations:
- Default configuration fails to detect any bad CWE-416 cases (since test cases do not reallocate)
- In **revoke-on-free** configuration Cornucopia detects all CWE-416 cases but at unpractical overhead
- Both configurations detect all CWE-415 bad cases







Evaluation: SPEC CPU 2006
Performance overhead: PICASSO (5% g.m.) and CheriBSD 25.03 Cornucopia (11% g.m. single-core, 8% g.m. 2-core)
Memory overhead (measured as maximum resident set size): PICASSO (8% g.m.) and Cornucopia (34% g.m.)

Performance overhead : SQLite
- PICASSO: 8% avg., 12% max.
- Cornucopia: %12 avg., 62% max.

Memory overhead: SQLite
- PICASSO: no memory overhead
- Cornucopia: 99% memory overhead across whole benchmark