Capability Sealing and Controlled Amplification

CHERI capabilities have “object type” field:

- Type -1: capabilities as described so far.
  Rights- and bounds-mediated access to memory.
- Otherwise (“sealed”): immutable, no loads or stores.

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 contract FA8750-10-C-0237. 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.

Capability Sealing and Controlled Amplification

Simplest sealed forms are “entry” capabilities:

- Any executable capability can be sealed as an entry.
- Unseal only by jumping to it:
  Unsealed form installed as PCC.
- (Shamelessly stolen from M-Machine)

Good for

- Control flow integrity: can’t change code entry.
- Creative use of trampolines can hide globals from other compilation units.
  - Entry grants capability load rights to trampoline
  - PCC-relative fetch of loader-provided globals pointer

Capability Sealing and Controlled Amplification

More advanced forms of sealing are mediated by capabilities.

- In addition to virtual addresses access, capabilities can authorize sealing and unsealing for ranges of object types.
- `cseal`: use seal-authorizing cap to construct sealed copy of an unsealed input.
- `cunseal`: use unseal-authorizing cap to get unsealed copy, if object type in authorized region.

Good for object pointers:

- Object constructor seals result.
- Methods enforce that arguments are sealed at right type.

Capability Sealing and Controlled Amplification

Often want both method and data to be sealed.
`ccall` instruction simultaneously unseals two capabilities with matching object type fields.

- One is installed as PCC.
- The other lands in a dedicated register for code's use.