Cookie map:

an alternative mmap() API

Brooks Davis

BSDCam 2017

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.

SRI International
University of Cambridge

mmap() usage

- Create stacks
- Allocate heap for malloc()
- Map files (i.e. install, cp)
  - Map libraries (multiple mappings in a region)
- Create mini-address-space (bhyve, libcheri)
- Manage regions for JIT

The mmap() family

- mprotect() – set permissions
- madvise() – alter paging strategy
- msync() – write modified pages to backing store
- minherit() – behavior of mapping across fork()
- mincore() – get status of pages

What's wrong with mmap()?

- Encourages explicit virtual address management
- Can replace any page, even the wrong one
- Permissions model doesn't fit W^X
  - On OpenBSD most pages can be made executable later
  - No way to support pointer permissions with JIT
- Must return exactly the number of pages requested
  - No rounding up for super pages, etc

Design goals

- Allow regions to be reserved
- Allow allocations to be rounded up
- Permit, but discourage explicit address management
  - Make changes to part of a reservation easy
- Support introspection without compromising ASLR
- Support immutability and non-reuse
- Allow multiple pointers (with different permissions) to be returned for a region (CHERI)

API sketch 1/2

- int cmmap(vaddr_t hint, size_t len, int prot, int flags, int fd, off_t offset, cm_t *cookiep)
  - Reserve a region and return a cookie.
- int cmgetptr(cm_t cookie, void **ptrp)
  - Get a pointer to the region.
- int csubmap(cm_t cookie, size_t mem_offset, size_t len, int prot, int flags, int fd, off_t offset)
  - Replace part (all) pages in a region

API sketch 2/2

- int cmclose(cm_t cookie, int flags)
  - Close a cookie optionally unmapping
- int cmrestrict(cm_t cookie, XX ops, XX *oops)
  - Restrict the set of operations on a cookie
- int cmstat(cm_t cookie, size_t index, struct cm_stat * cs)
  - Return data on a series of submaps
- cmadvise(), cmincore(), cminherit(), cmsync(), cmunmap()
  - Like mmap() counterparts, but within region

API Sketch (CHERI extensions)

- int cmgetcap(cm_t cookie, void **ptrp, perm_t perms)
  - Get a capabiltiy pointer with the requested permissions
- int cmandperm(cm_t cookie, perm_t perms, perm_t *operms)
  - Update the set of allowed permission for new capabilities

Alternative cmmap()/cmsubmap() API

```
struct cmreq cmr;

cm_t cmp;

void *ptr;

CMREQ_INIT(&cmr, len, prot); // Anon memory

CMREQ_SETFD(&cmr, fd); // Map a file at offset 0

CMREQ_SETFILEOFFET(&cmr, off); // Map file off

CMREQ_SETMAPOFFSET(&cmt, 4096); // Map at 4k offset

CMREQ_SETSHARED(&cmr); // Map shared

cmmap(&cmr, &cmp);

cmgetptr(&cmr, &ptr);
```

Feedback requested

- Does this meet your needs?
- Does it seem sane?
- Is a request struct too un-UNIX-like?