Glibc Heap Exploitation for beginners

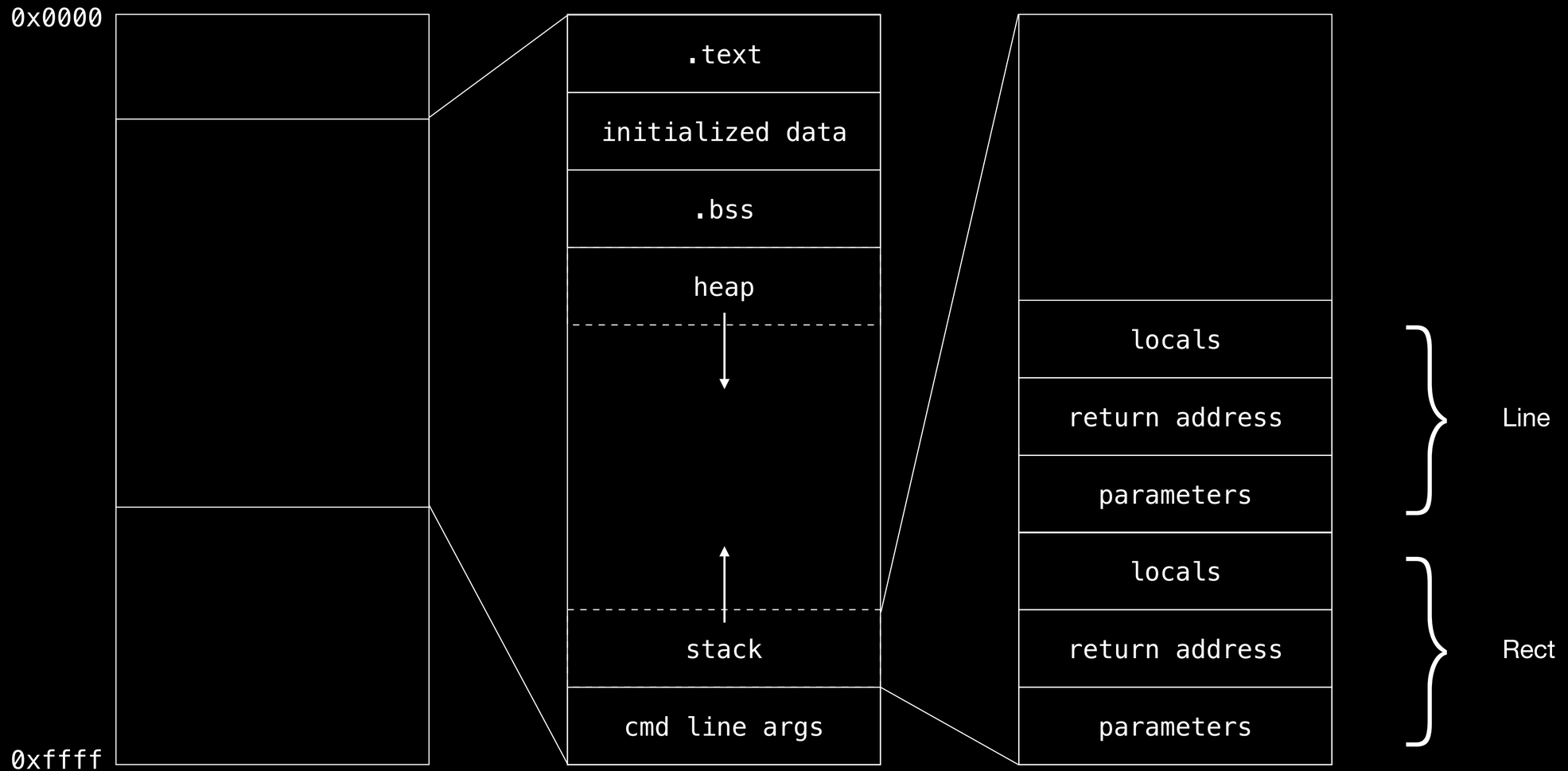
Dynamic allocator misuse

hanemile - 2025-03-02

Basics

Memory - Binary layout

Where’s everything located?



Glibc?

Dynamic allocator

Interaction

- void* malloc(size_t bytes)
- void* calloc(size_t n, size_t elem_size)
- void* realloc(void* oldmem, size_t bytes)
- void* free(void* mem)

Malloc hooks

malloc/hooks.c

- __after_morecore_hook
- __free_hook
- __malloc_hook
- __malloc_initialize_hook
- __memalign_hook
- __realloc_hook

Top Chunk

Where to get memory from

- ads

Bins

Where stuff lands 1/2

- fastbin -> smallbin -> largebin -> mmap()

- small: double linked, < 1024 bytes, 62 in total
- large: double linked, > 1024 bytes, 63 in total
- unsorted: double linked, chunks that don’t fit into fast or tcache
- fast: single linked, <88 bytes, 10 in total
- tcache: like fast, threadlocal, 64 in total, 7 chunks of 24..1032 bytes

Bins

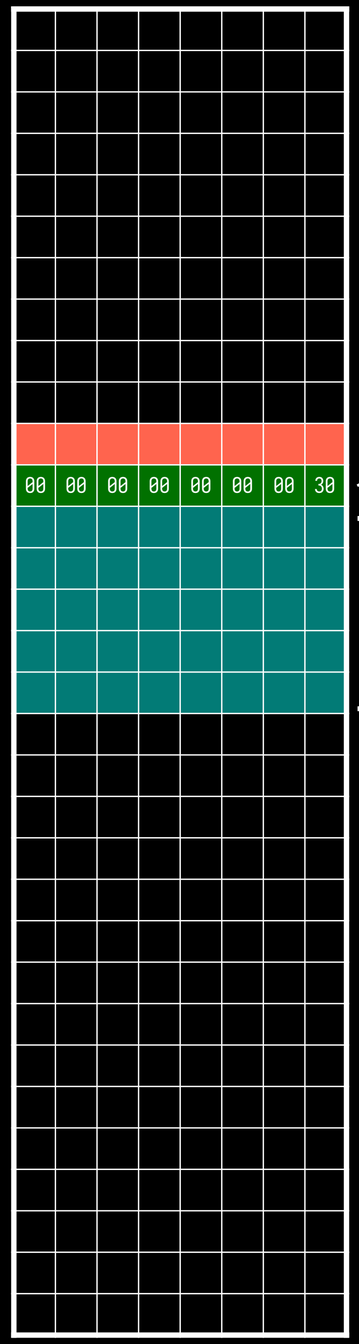
Where stuff lands 2/2

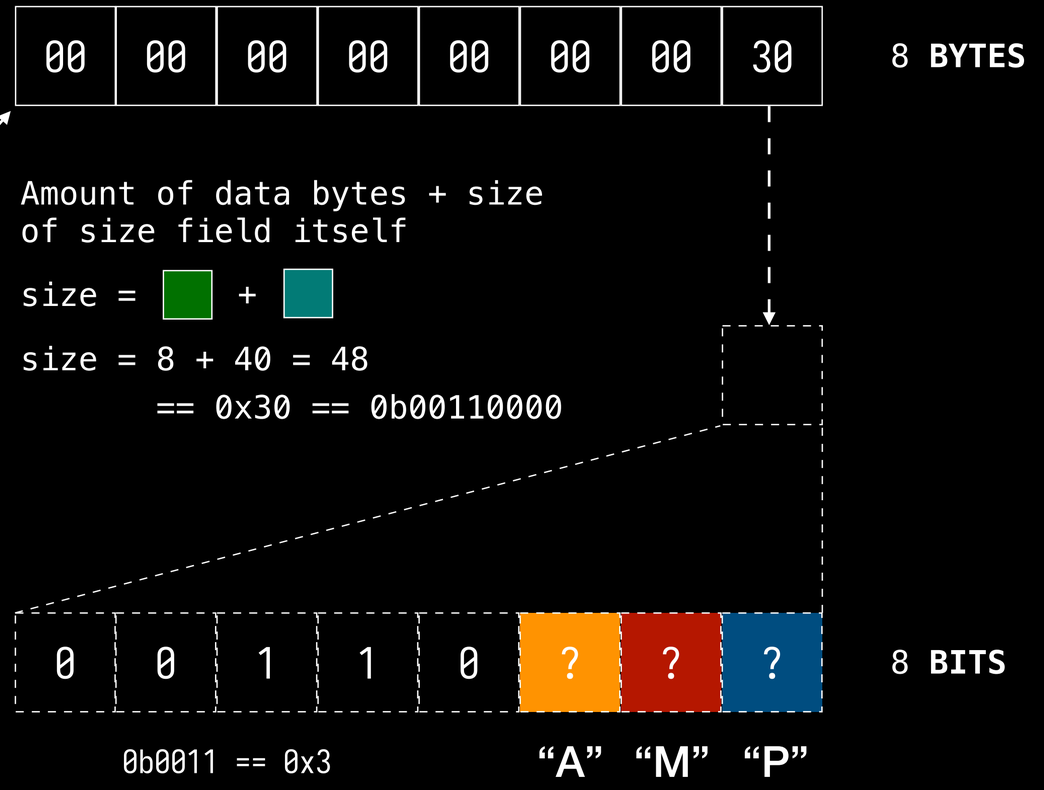
- Allocator frees into bins: tcache -> fastbin -> unsorted
- pointers forward and backwards are stored in the bins themselves

Chunk

the size field

(of an allocated chunk)





min chunk size

`0x20`

“fencepost chunk” size
(used internally)

`0x10`

NON MAIN ARENA
When set: chunk does not belong to main arena

IS MMAPPED
When set: chunk allocated using mmap()

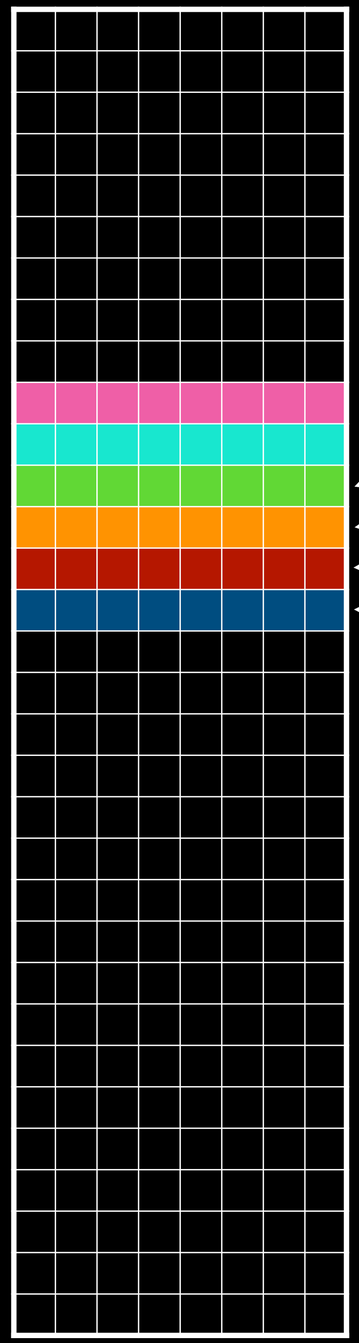
PREV IN USE
When set: Previous chunk in use
When free: Previous chunk free

Chunk

Free

prev_size: size of the previous chunk

chunk_size: size of the current chunk

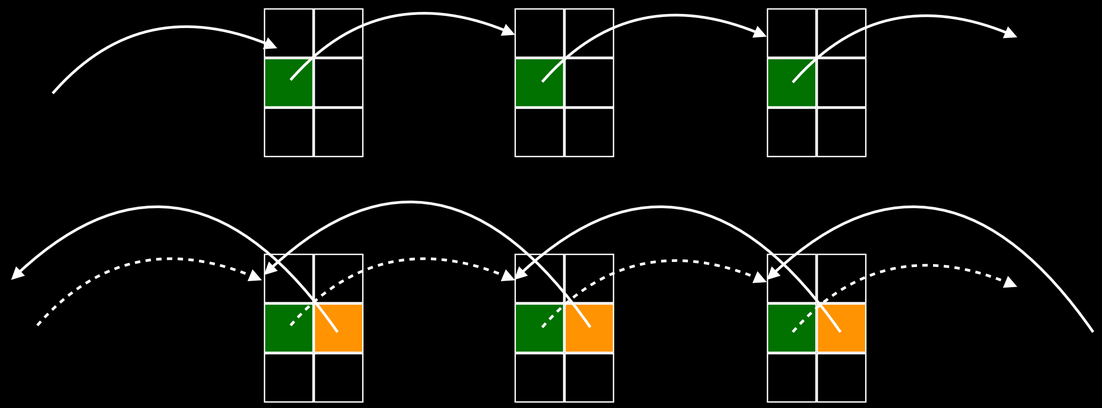


fd: Forward pointer to next free chunk
used by all kinds of bin

bk: Backwards pointer to previous free chunk
only in bins using double linked lists
(unsortedbin, smallbin, …)

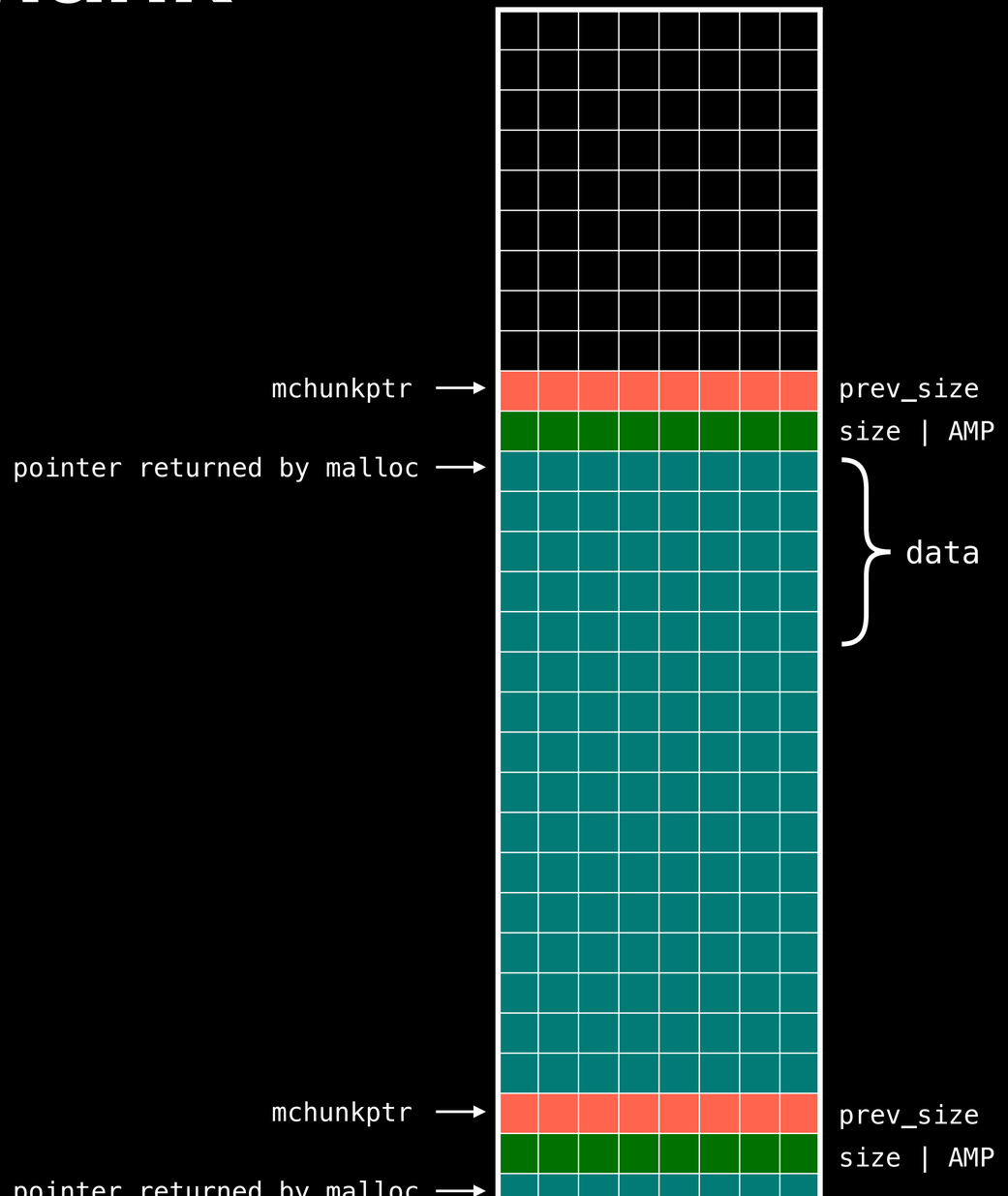
fd next size: size of next free chunk
only used in largebins

bk next size: size of previous free chunk
only used in largebins

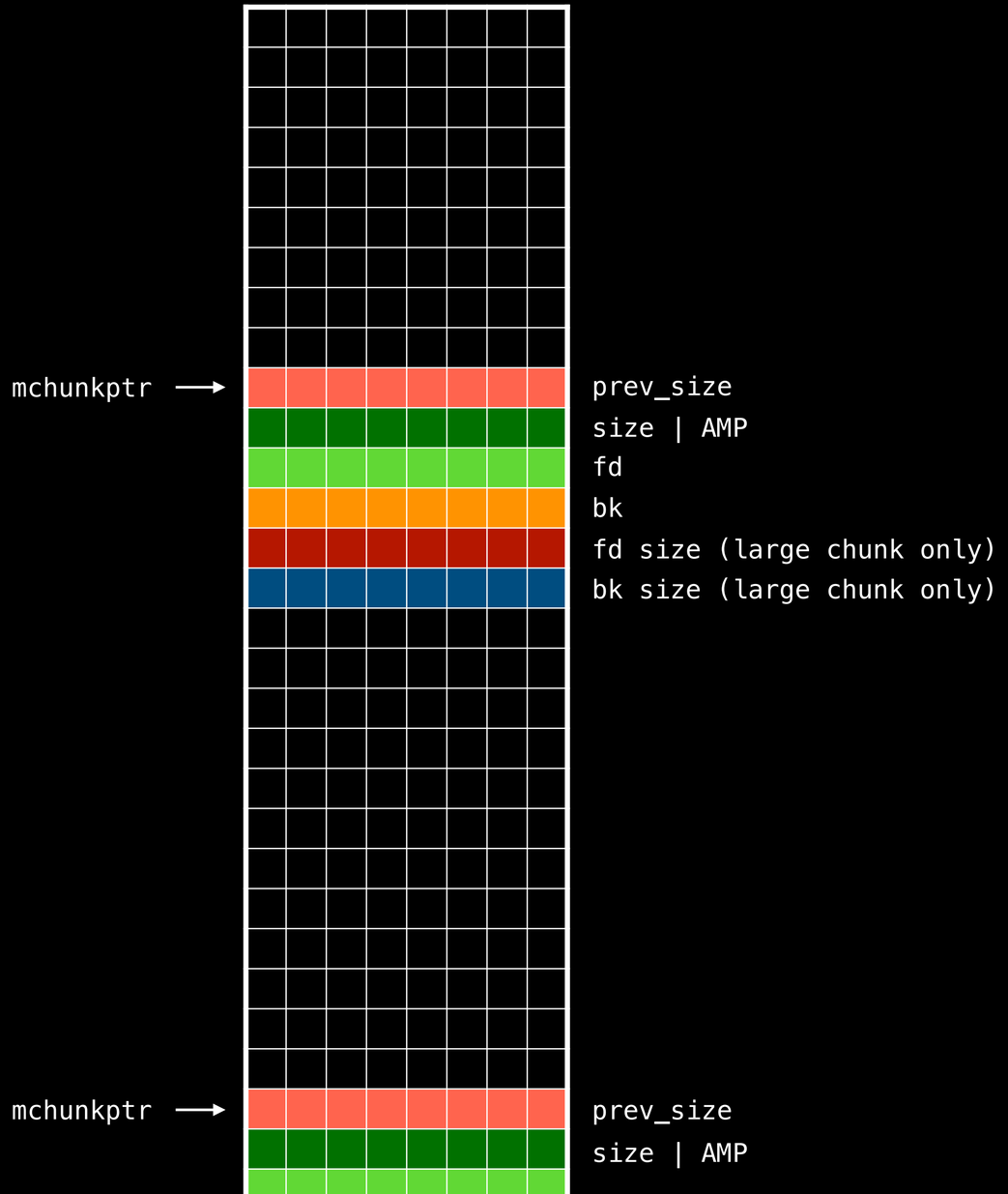


Chunk

In Use



Free



House of force

House of force

Prerequisites

- Allocate arbitrary large chunks
- Heap based buffer overflow
- glibc <2.27

House of force

Setup notes

- Specify the path of the interpreter:
  - `patchelf --set-interpreter ld-linux-x86-64.so.2`
- Specify the path to the library:
  - `patchelf --set-rpath . a.out`
- glibc <2.27

House of force

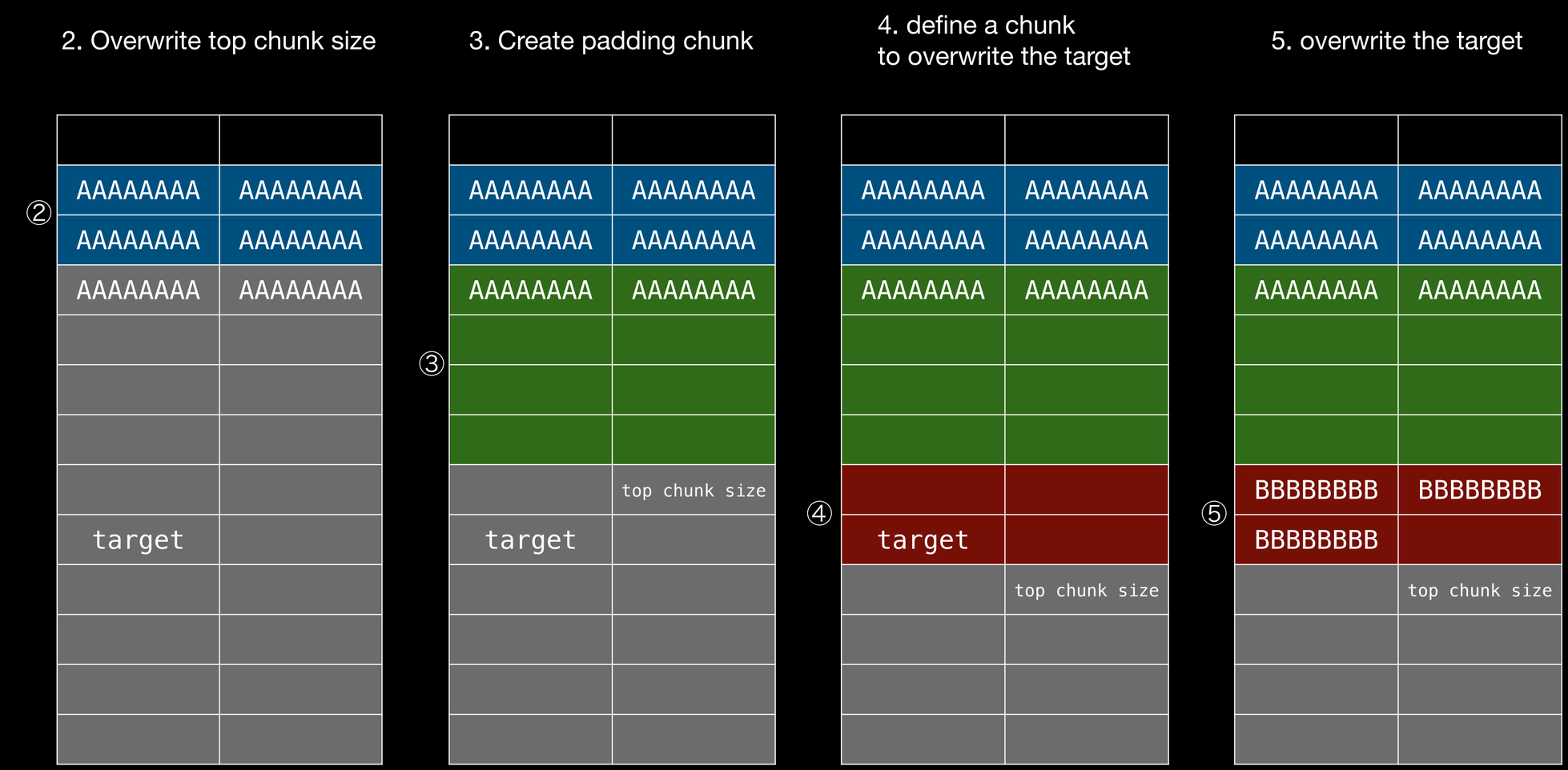
The steps in a non-visual way (aka. “the concept”)

- Overwrite top chunk size in order to be able to create arbitrary large chunks
- Create a “padding chunk” in bridging the gap to the address that should be overwritten
- Create a “malicious chunk” that can be used to overwrite the address we want to control
- Overwrite the address

House of force

When the target is after the heap

① a = malloc(small)
② strcpy(a, small++)
③ b = malloc(pad)
④ c = malloc(whatever)
⑤ strcpy(c, overwrite)



House of force

When the target is *before* the heap

① a = malloc(small)
② strcpy(a, small++)
③ b = malloc(pad)
④ c = malloc(whatever)
⑤ strcpy(c, overwrite)