Advanced Cybersecurity Topics

–

Heap Exploitation

20-21

Exploit Heap Overflow to gain RCE

- Use after free
- Exploit glibc implementation to get:
  - Arbitrary Write
  - EIP Control

Memory Allocations

- syscall
  - mmap (allocate memory page)
  - munmap (deallocate memory page)
  - brk/sbrk (change the location of the program break)
- libc
  - malloc - allocate a chunk of memory
  - calloc - allocate and zero-out memory
  - realloc - change size of an allocation
  - free - free a chunk of memory

The HEAP Allocators

- **ptmalloc** (glibc)
- dlmalloc (was in glibc)
- tcmalloc (chromium)
- jemalloc (FreeBSD, Firefox, Android)
- splittings, fits, coalescing, segregations (free list, storage, non determinism)

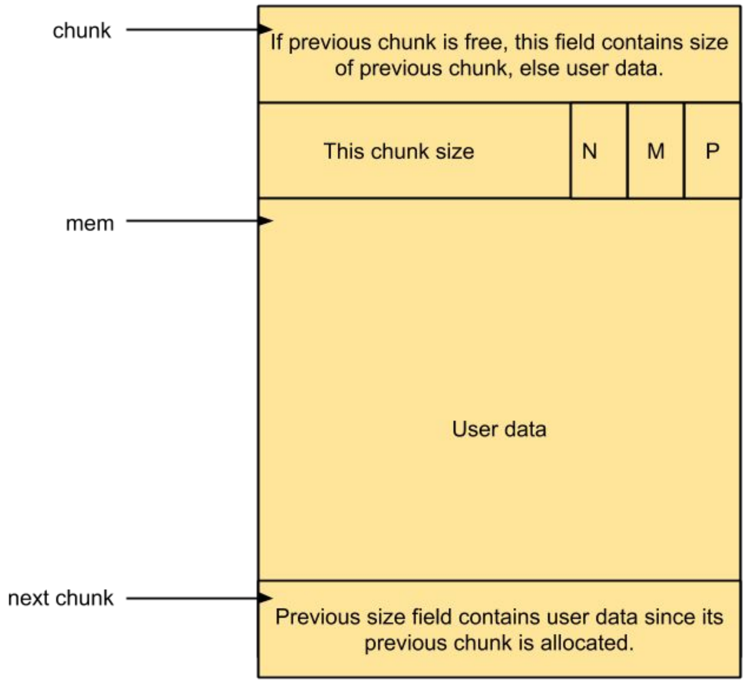
ptmalloc2 (aka the malloc of glibc)

- **splittings** (how to divide in chunk)
- **fits** (match requested size with)
- **coalescing** (how to merge chunks)
- **segregations** free list
- NO segregations storage
- **deterministic**

Best documentation is source code.

```
    chunk-> +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            |             Size of previous chunk, if unallocated (P clear)  |
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            |             Size of chunk, in bytes                     |A|M|P|
      mem-> +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            |             User data starts here...                          .
            .                                                               .
            .             (malloc_usable_size() bytes)                      .
            .                                                               |
nextchunk-> +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            |             (size of chunk, but used for application data)    |
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
            |             Size of next chunk, in bytes                |A|0|1|
            +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

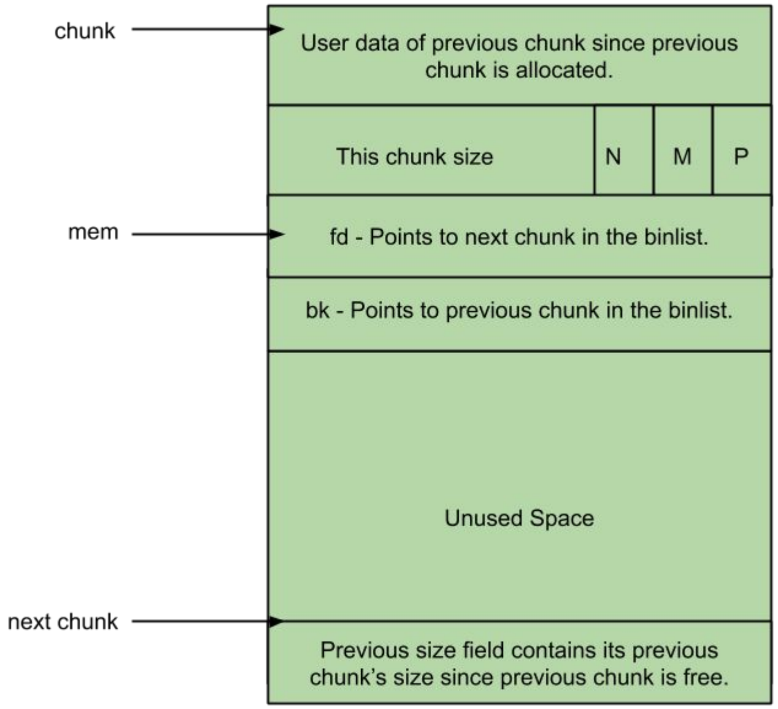
Chunks


chunk
If previous chunk is free, this field contains size of previous chunk, else user data.
This chunk size
N
M
P
mem
User data
next chunk
Previous size field contains user data since its previous chunk is allocated.

Allocated Chunk

- PREV_INUSE (P) – This bit is set when previous chunk is allocated.
- IS_MMAPPED (M) – This bit is set when chunk is mmap'd.
- NON_MAIN_ARENA (N) – This bit is set when this chunk belongs to a thread arena.

Free Chunks


chunk
User data of previous chunk since previous chunk is allocated.
This chunk size
N
M
P
mem
fd - Points to next chunk in the binlist.
bk - Points to previous chunk in the binlist.
Unused Space
next chunk
Previous size field contains its previous chunk's size since previous chunk is free.

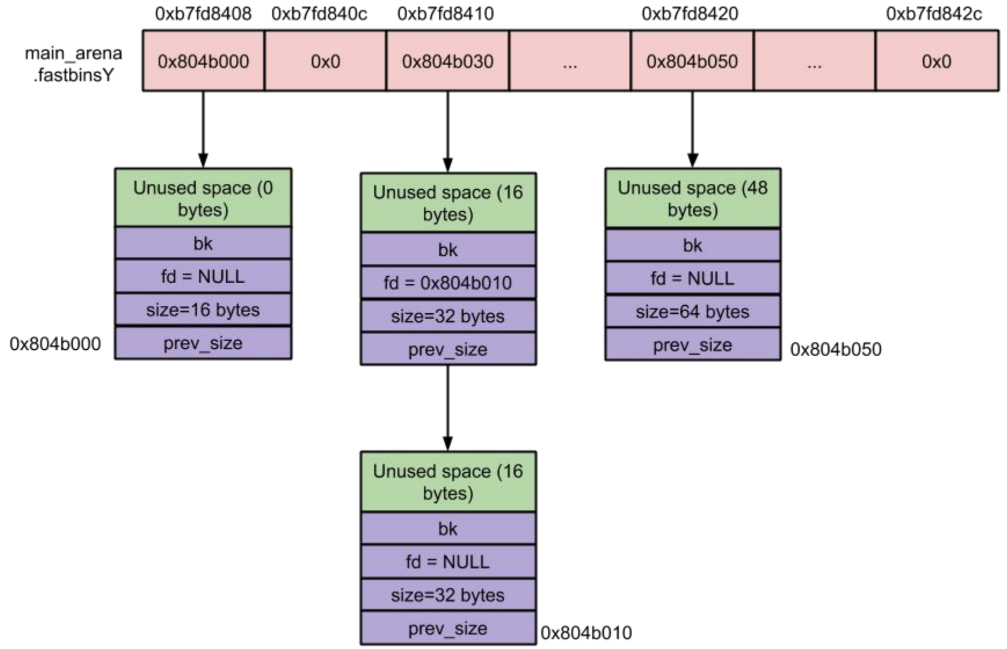
Free Chunk

- PREV_INUSE (P) – This bit is set when previous chunk is allocated.
- IS_MMAPPED (M) – This bit is set when chunk is mmap'd.
- NON_MAIN_ARENA (N) – This bit is set when this chunk belongs to a thread arena.

Bins

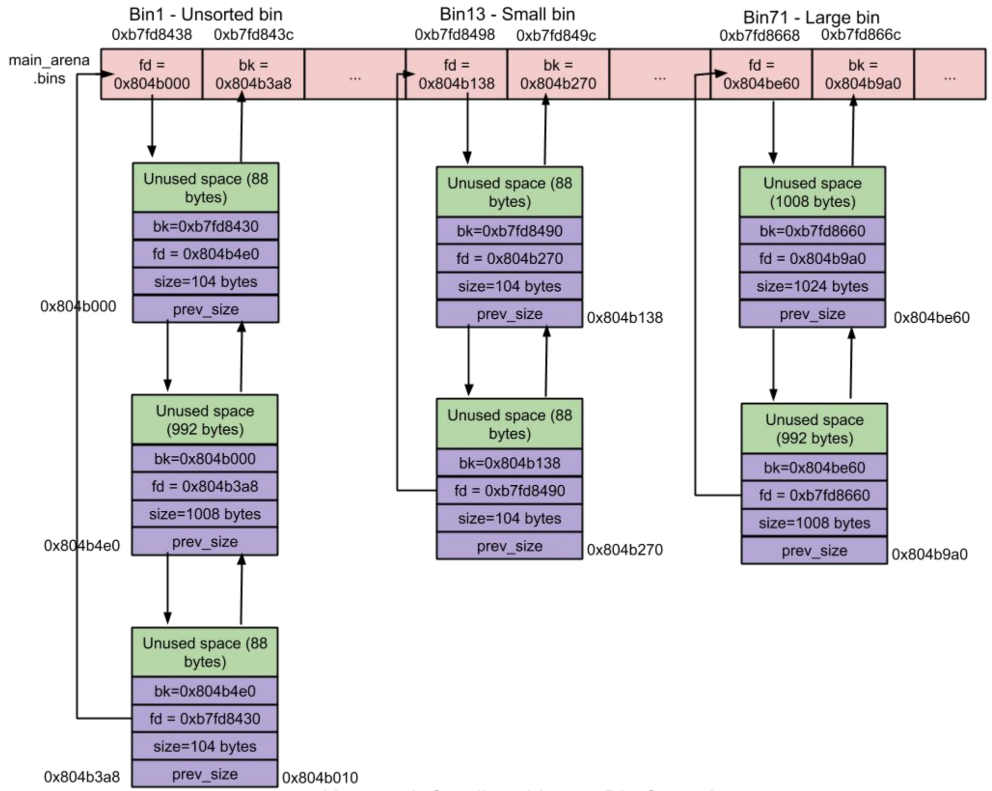
- t-cache
- Fast bin (16 to 80 bytes)
- Unsorted bin
- Small bin (< 512 bytes)
- Large bin (>= 512 bytes)
- top-chunk

Fast Bins



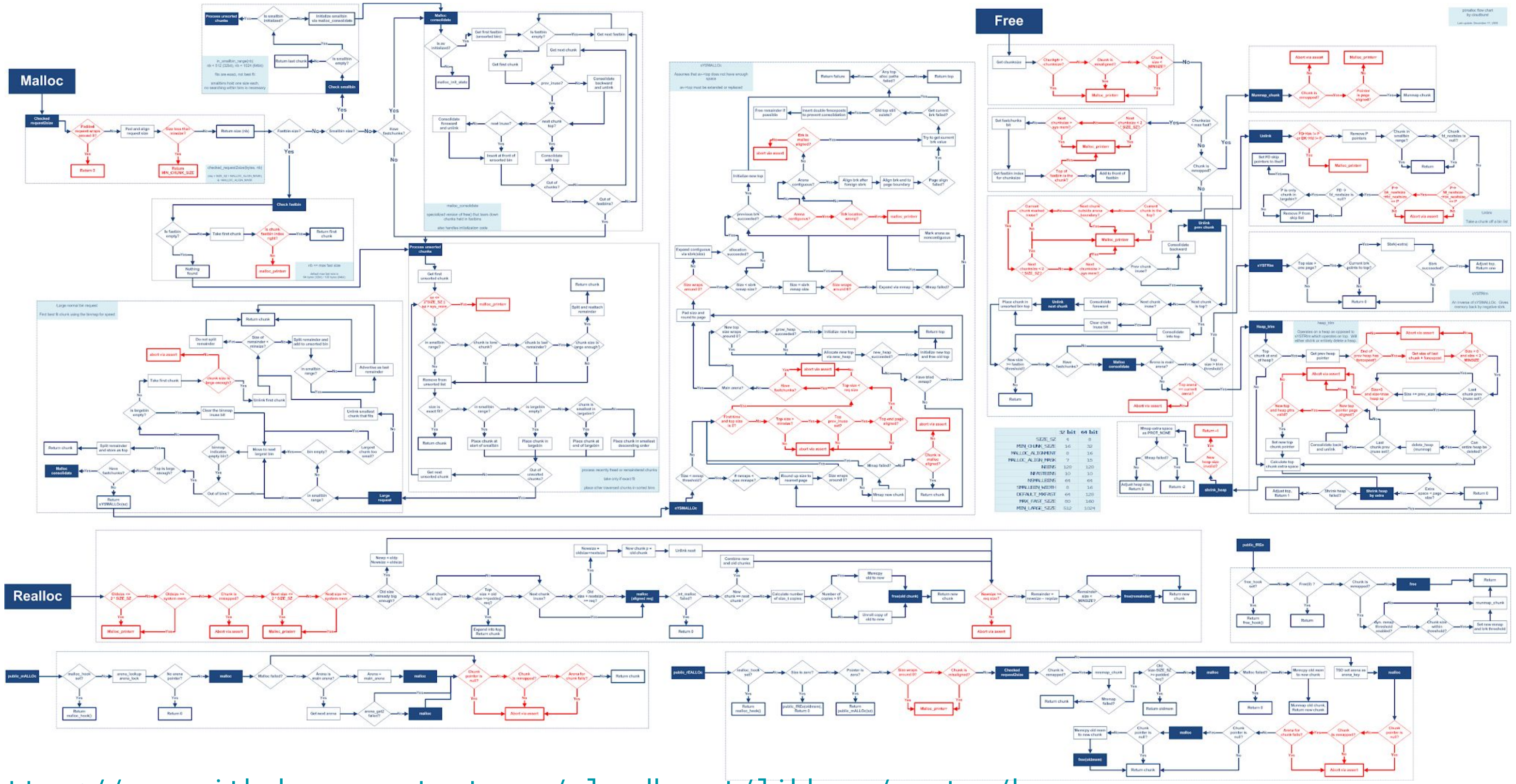
Fast Bin Snapshot

Bins (Unsorted, Small, Large)



Unsorted, Small and Large Bin Snapshot

Algorithm



	32 bit	64 bit
SIZE_SZ	4	8
MIN_CHUNK_SIZE	16	32
MALLOC_ALIGNMENT	8	16
MALLOC_ALIGN_MASK	7	15
NBINS	128	128
NFASTBINS	10	10
NSMALLBINS	64	64
SMALLBIN_WIDTH	8	16
DEFAULT_MXFAST	64	128
MAX_FAST_SIZE	80	160
MIN_LARGE_SIZE	512	1024

https://raw.githubusercontent.com/cloudburst/libheap/master/heap.png

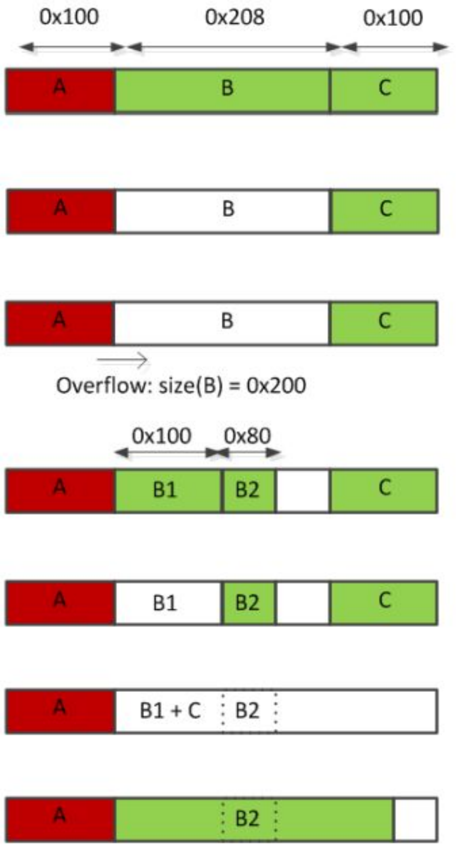
House of Force

- modify size of top-chunk
- malloc of arbitrary size
- malloc will return an arbitrary address

Poison Null Byte

```
char *buf = malloc(128);
int read_length = read(0, buf, 128);
buf[read_length] = 0;
```

Poison Null Byte



Initial state

B is free

Overflow into B
- Size truncated to 0x200 from 0x208
- Further allocations in that space do not properly update C's "prev_size" field

Two allocations within the old B chunk
The first is not a fastbin

The beginning of the old B chunk is free

C is freed and merged with the old B, where a valid non-fastbin free chunk resides

1+ allocations larger than B1's initial size
B2 is overlapped

https://heap-exploitation.dhavalkapil.com/attacks/shrinking_free_chunks.html

Useful Links / Reading Material

- https://github.com/shellphish/how2heap
- https://sploitfun.wordpress.com/2015/02/10/understanding-glibc-malloc/
- https://heap-exploitation.dhavalkapil.com
- https://www.usenix.org/conference/usenixsecurity18/presentation/heelan (Automatic Heap Manipulation)