Heap là gì?

- **Stack**: lưu biến cục bộ, kích thước cố định.
- **Heap**: dùng cho cấp phát động (`malloc`, `new`), kích thước phụ thuộc input của user.
- **Allocator**: bộ quản lý heap, chia memory thành các **chunks** với metadata (free/used, size, prev, next).

Ví dụ metadata (simplified):

```
struct control_block_t {
   int available;
   int size;
   void* next_free;
   void* prev_free;
};
```

Khi `malloc`/`free`, allocator thao tác trên metadata này.

Cách Heap Hoạt Động

- **brk() / sbrk()**: system calls để mở rộng heap.
- Allocator giữ **linked list** các free chunks.
- Khi `malloc(size)`:
  - Tìm chunk free đủ lớn.
  - Nếu không có ⇒ gọi `sbrk()` để xin thêm memory.
- Khi `free(ptr)`:
  - Đánh dấu chunk free, thêm vào free list.

Heap Overflow

- **Khác stack overflow**: không ghi đè return address trực tiếp.
- Khi `strcpy(buf1, input)` dài quá ⇒ ghi đè qua chunk kế bên ⇒ **corrupt metadata**.
- Khi allocator xử lý chunk hỏng metadata ⇒ có thể biến thành **arbitrary write primitive**.

Exploit Idea: Metadata Corruption

Ví dụ 2 chunks liên tiếp:

[Chunk1][Chunk2]

- Overflow từ Chunk1 → ghi đè metadata của Chunk2.

- Nếu sửa:
    - `prev_free_chunk->next_free_chunk = TARGET_ADDR`
    - `next_free_chunk = VALUE`
- Khi `free()` hoặc `alloc()`, allocator sẽ ghi `VALUE` vào `TARGET_ADDR`.
  Từ đây ⇒ attacker có **4-byte arbitrary write**.
- TARGET có thể là:
    - **GOT entry** (Global Offset Table).
    - **DTORS section** (destructors).
    - **Return address**.

Ví dụ Exploit Payload

1. Fill Chunk1 với:
    - **NOP sled + shellcode**.
2. Overwrite Chunk2 metadata:
    - `available = 0x11111111` (free).
    - `size = >= 128`.
    - `next_free = &shellcode`.
    - `prev_free = TARGET-8`.

Khi allocator unlink → ghi `&shellcode` vào TARGET.
Lần gọi tiếp theo đến TARGET (ví dụ GOT entry của `printf`) ⇒ EIP nhảy vào shellcode.

Tình huống thực tế

- **Heap overflow**:
    1. Khó phát hiện hơn stack overflow.
    2. Phụ thuộc implementation allocator (dlmalloc, ptmalloc, jemalloc…).
    3. Xuất hiện nhiều trong browsers, PDF readers, client apps.
- **Các biến thể heap bugs**:
    1. **Double Free**: gọi `free()` 2 lần ⇒ free list hỏng ⇒ arbitrary write.
    2. **Use-After-Free**: tiếp tục dùng chunk đã free ⇒ attacker chèn data khác.
    3. **Heap Spraying**: attacker fill heap với NOP sled + shellcode để tăng xác suất hit.

Key Takeaways

- Heap Overflow ≠ Stack Overflow: không đè return address mà **chơi với metadata**.
- Phải hiểu chi tiết **allocator implementation** trước khi exploit.
- Exploit thường → tạo **arbitrary write** ⇒ control flow hijack (GOT, DTORS, vtable, return).
- Heap exploitation là “elite-level”, khó hơn stack overflow, nhưng mạnh và vẫn phổ biến.