A. Stack Overflows

- Nguyên lý: biến cục bộ và địa chỉ return cùng nằm trên stack. Viết tràn dữ liệu có thể ghi đè return address.
- **Arbitrary Code Execution**: ghi đè return address bằng địa chỉ hàm có lợi hoặc shellcode.
- **Shellcode**:
  - Nhỏ gọn, tự chứa, không có null byte, độc lập vị trí.
  - Ví dụ payload: spawn shell, add admin user, download rootkit, reverse shell.
- Kỹ thuật **NOP sled + shellcode** để tăng khả năng EIP trúng shellcode.

B. Heap & Heap Overflows

- **Heap** dùng cho cấp phát động (`malloc`, `new`).
- Hiểu về **allocator** (chunk metadata, linked list free chunks).
- Heap overflow xảy ra khi ghi đè metadata ⇒ có thể đạt được **arbitrary write**.
- Ví dụ: ghi đè **prev_free_chunk** & **next_free_chunk** để chỉnh sửa GOT/DTORS.
- Đặc điểm:
  - Khó tìm và khai thác hơn stack overflow.
  - Phụ thuộc vào implementation của allocator (moving target).
  - Xuất hiện nhiều trong client-side (browser, PDF reader…).

C. Các kịch bản lỗ hổng khác

1. **Format String Vulnerability**
   - `printf(user_input)` ⇒ attacker điều khiển specifier (`%x`, `%n`) ⇒ leak info, ghi đè 4 byte bất kỳ.
2. **Integer Overflow / Signedness Errors**
   - Khi số nguyên tràn, phép toán size/len không như mong muốn ⇒ buffer nhỏ hơn dự kiến ⇒ overflow.
3. **Off-by-One Errors**
   - Chỉ cần ghi tràn 1 byte cũng có thể khai thác (ví dụ ghi đè `EBP` thấp).
4. **Uninitialized Variables**
   - Stack frame không reset ⇒ biến rác có thể chứa data do attacker kiểm soát ⇒ khai thác trong `sprintf`, `strcpy`.
5. **Double Free / Use After Free**
   - Giải phóng cùng một chunk nhiều lần hoặc dùng sau khi free ⇒ có thể lợi dụng để ghi đè tùy ý.

D. Thực tế lỗ hổng phần mềm

- Ví dụ từ **IMAP, ProFTP, Apache, Linux Kernel, Snort, OpenBSD FTP, FreeBSD kernel**.

- Các bug hiện đại thường: **off-by-one, integer overflow, signedness, bounds check sai** thay vì `strcpy` rõ ràng.

E. Tìm lỗ hổng

1. **Closed Source**
    - Fuzzing (tìm crash ⇒ có thể khai thác).
    - Reverse Engineering.
    - Binary Diffing (so sánh patch).
2. **Open Source**
    - Manual Source Code Inspection (hiệu quả nhất cho bug subtle).
    - Automated Tools (splint, static analyzers, nhưng nhiều false positive).
3. **Tips**
    - Tập trung vào chỗ xử lý input user.
    - Kiểm tra các loop/manual parsing, bounds check, phép toán với user-controlled integers.
    - Tìm trong comments/bugs cũ.

F. Công nghệ chống khai thác (Exploit Mitigation)

1. **DEP / NX Bit**
    - Ngăn stack/heap thực thi.
    - Có thể bypass bằng **ret2libc** (return-to-libc).
2. **ASLR (Address Space Layout Randomization)**
    - Random hóa địa chỉ stack/heap/libc.
    - Làm khó dự đoán địa chỉ, nhưng không hoàn toàn random ⇒ attacker vẫn có cách.
3. **Kết hợp ASLR + DEP**
    - Ngăn chặn nhiều attack, nhưng vẫn tồn tại bypass (ROP - Return Oriented Programming)