BÁO CÁO THỰC HÀNH LẬP TRÌNH HỆ THỐNG LAB5 NHÓM 12

22520825 – Nguyễn Đức Luân

22520661 – Vũ Ngọc Quốc Khánh

22521110 – Đào Hoàng Phúc

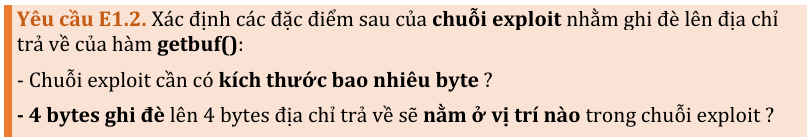
Level 0: (Mục tiêu thực thi hàm smoke)

A close-up of a question

Description automatically generated

A diagram of a computer

Description automatically generated

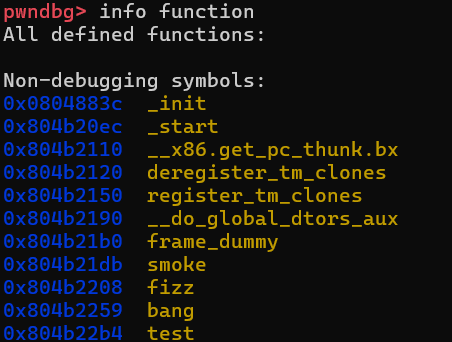


-Theo như mô hình stack trên, vậy để đè được lên địa chỉ return address của hàm getbuf() thì ta phải nhập hết 50 bytes của chuỗi buf, sau đó đè lên hết 4 bytes đoạn Saved EBP và sau đó là địa chỉ của hàm smoke

-4 bytes ghi đè lên địa chỉ trả về nằm ở vị trí cuối trong chuỗi exploit, vì chuỗi chúng ta nhập vào sẽ lần lượt truyền vào từ dưới (địa chỉ thấp) lên trên (địa chỉ cao) theo stack trên.



Dùng GDB để lấy địa chỉ hàm smoke:



Dùng IDA pro 7.7 để lấy địa chỉ hàm smoke

A screen shot of a computer

Description automatically generated

Địa chỉ hàm smoke là 0x804B21DB



Tạo file smoke.txt chứa chuỗi exploit như sau:

A screenshot of a computer screen

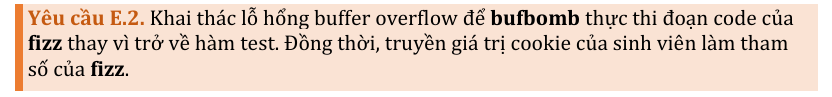
Description automatically generated



A black screen with blue text

Description automatically generated

Level 1: (Mục tiêu: Thực thi hàm fizz)

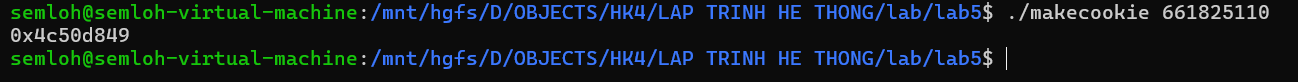


Đầu tiên ta xem code của hàm fizz:

A screen shot of a computer code

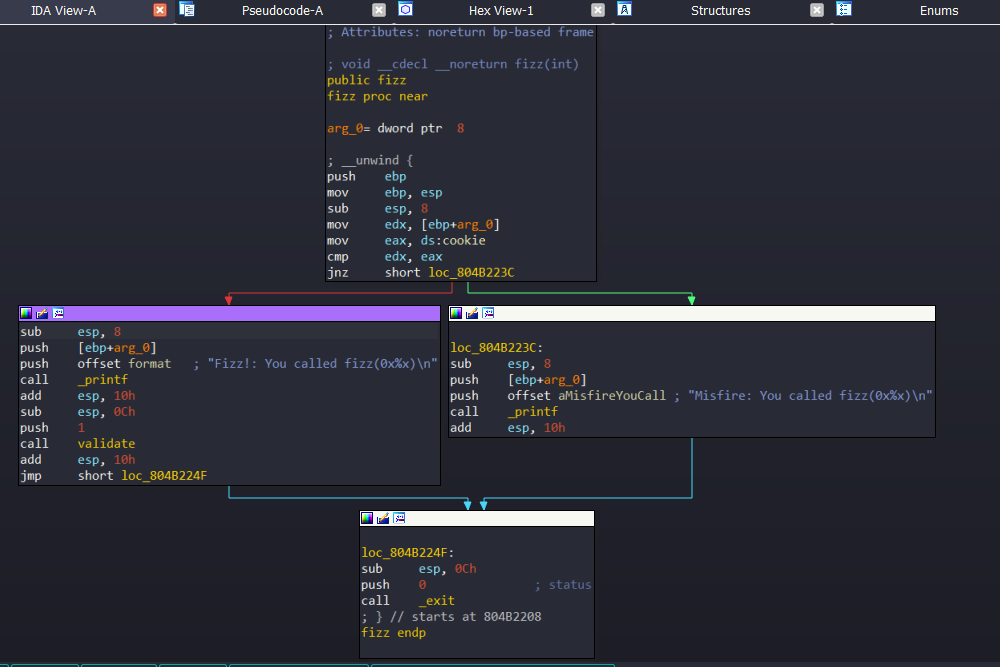
Description automatically generated

Ta thấy để thực thi đúng yêu cầu đề bài, nhảy vào hàm fizz và in ra message “Fizz!: You called fizz…” thì phải thỏa điều kiện biến a1 = cookie. Mà cookie được generate dựa vào file makecookie được cung cấp sẵn với đầu vào là format MSSV theo quy định file lab:



Vậy nên ta cần ghi đè giá trị đối số a1 trên thành giá trị cookie hợp lệ là thành công.

Xem mã assembly của hàm fizz:



Ở đây có một điều thú vị là nếu ta ghi đè địa chỉ return address như level trên thì sau khi thực hiện instruction ret , chương trình sẽ nhảy thẳng đến đoạn code thực thi của hàm fizz bắt đầu bằng lệnh “push ebp” như ảnh trên.

Nhảy chương trình với cách như thế sẽ khác với dùng call instruction một chút. Bình thường khi dùng call, stack sẽ tự động thêm return address vào. Còn nếu dùng phương pháp ghi đè ở level 0 thì không có

Với stack getbuf() ở trên sau khi kết thúc chương trình với instruction ret và leave, stack sẽ được dọn dẹp sạch sẽ bao gồm cả return address trước khi nhảy đến địa chỉ return address đang chứa chính là địa chỉ của câu lệnh ngay sau lệnh call.

Sau đó thì nhảy trực tiếp đến hàm fizz, thực hiện các instruction cơ bản như “push ebp” “mov ebp, esp” -> Khởi tạo stack như thông thường, điểm duy nhất khác biệt là không có return address. Vậy stack lúc này so với stack trước như sau:

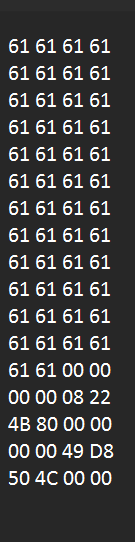
Có thể thấy rằng do không có return address nên ebp lưu vào trong stack của hàm fizz sẽ có vị trí cao hơn ebp của hàm buf 4 byte. Vì để hàm fizz đọc được tham số truyền vào ở vị trí ebp + 8 thì sẽ tương ứng với vị trí ebp +12 của hàm buf.

A diagram of a data flow

Description automatically generated

Vậy payload của chúng ta như sau:

50 bytes buffer + 4 bytes (Saved ebp) + 4 bytes địa chỉ fizz(ebp +4) + 4 bytes (tại <ebp+8>) + giá trị cookie (Little-endian)(ebp +12)



Với địa chỉ của hàm fizz là: 0x804b2208

A screen shot of a computer

Description automatically generated

Thực hiện kết quả:

