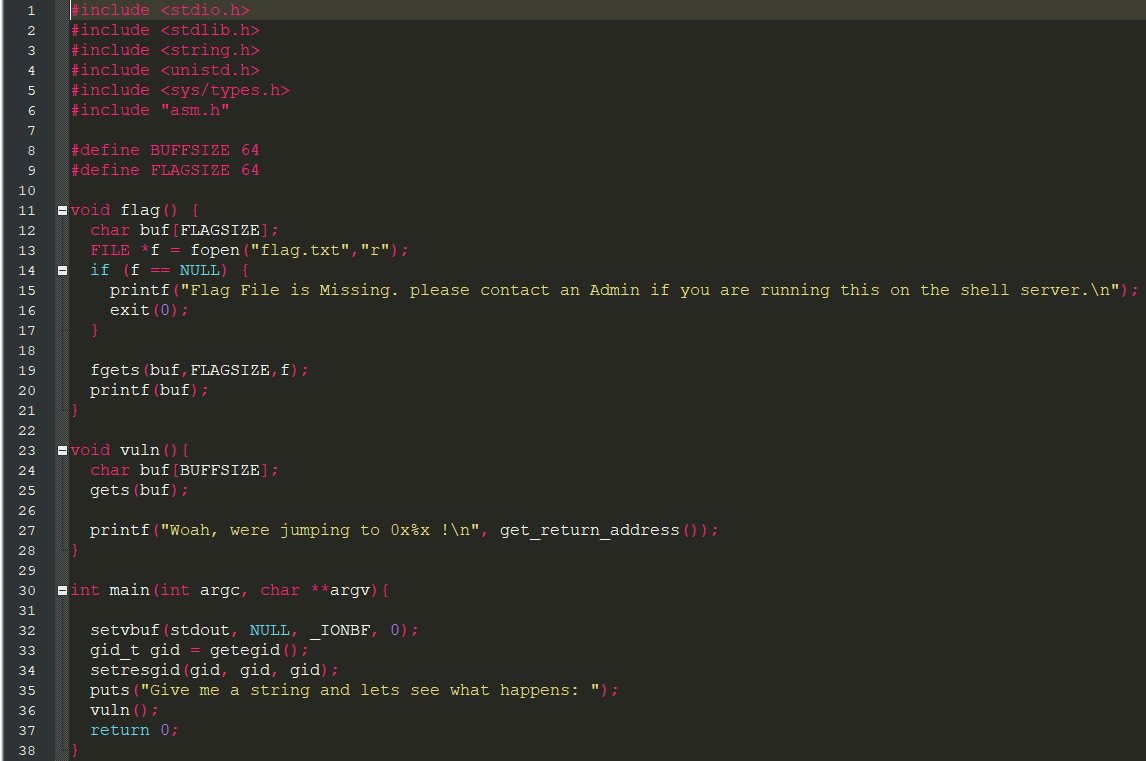
Overflow 1 solution:

1. Đọc và hiểu source code:



Chương trình có 3 hàm :

* Flag( ) mở file flag.txt, đưa dữ liệu trong file vào mảng kí tự buf và in buf.
* Vuln( ) khai báo một mảng kí tự buf[64] và cho chúng ta nhập vào bằng gets( ) . Sau đó in ra địa chỉ return của hàm vuln, lấy kết quả trả về bởi hàm get\_return\_address( ) .
* Main( ) gọi hàm vuln( ) .

1. Phát hiện lỗi:

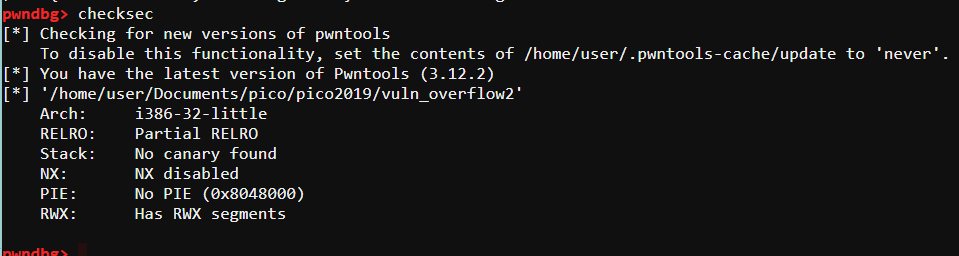
Trong hàm vuln( ) , ta thấy hàm gets ( ) cho nhập nhưng không kiểm tra số lượng kí tự mà chúng ta nhập vào cho nên gây ra lỗi buffer overflow.

1. Khai thác lỗi:
   1. Ý tưởng:

Chúng ta cần in ra flag từ chương trình, nhưng trong main( ) và vuln( ) không hề gọi về hàm flag ( ) . Chúng ta có thể lợi dụng lỗi buffer overflow ở vuln( ) để làm tràn địa stack, ghi đè lên địa chỉ trả về cũ của vuln( ) thành địa chỉ của hàm flag.

* 1. Thực hiện:

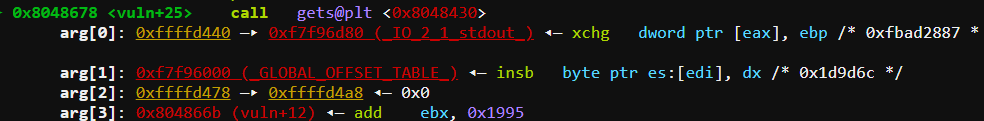
Kiểm tra các ràng buộc bảo vệ của file :



No pie và no canary là đủ cho chúng ta khai thác bằng ý tưởng trên.

B1: xác định offset từ buf đến địa chỉ trả về của vuln.

Đặt breakpoint ở gets trong vuln( ) :



Ta thấy địa chỉ buf là 0xffffd440.

Còn địa chỉ ebp của chúng ta là :



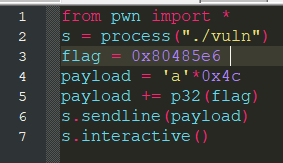
Vậy offset = ebp+4 – buf = 0x4c => Để làm tràn đến địa chỉ return của vuln thì ta cần 0x4c kí tự.

B2: xác định địa chỉ hàm flag( ) .



Đây rồi, ta đã có đủ dữ liệu để làm.

B3: viết code.



B5: lấy flag



Vậy là chúng ta đã solve xong bài này.