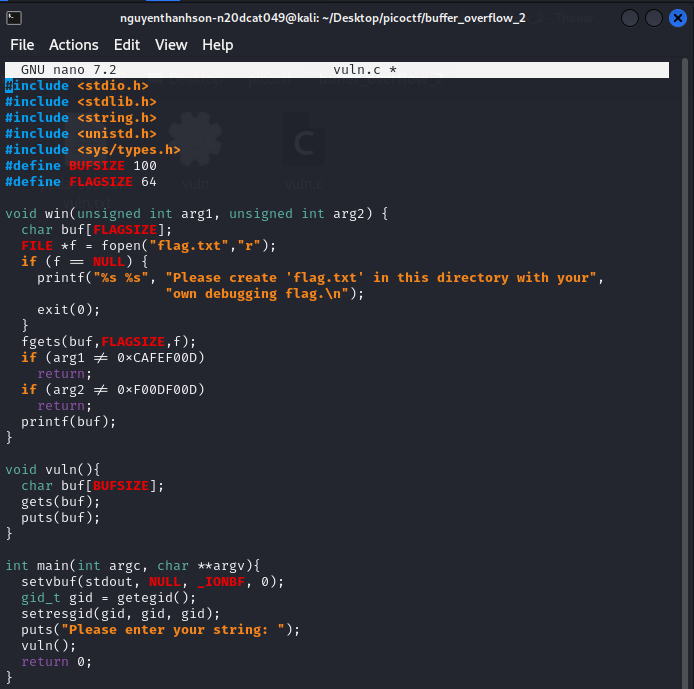
**Buffer overflow 2**

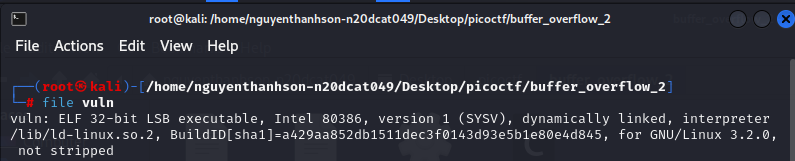
Ta được cung cấp source code của chương trình c và tệp thực thi của nó

**Phân tích:**

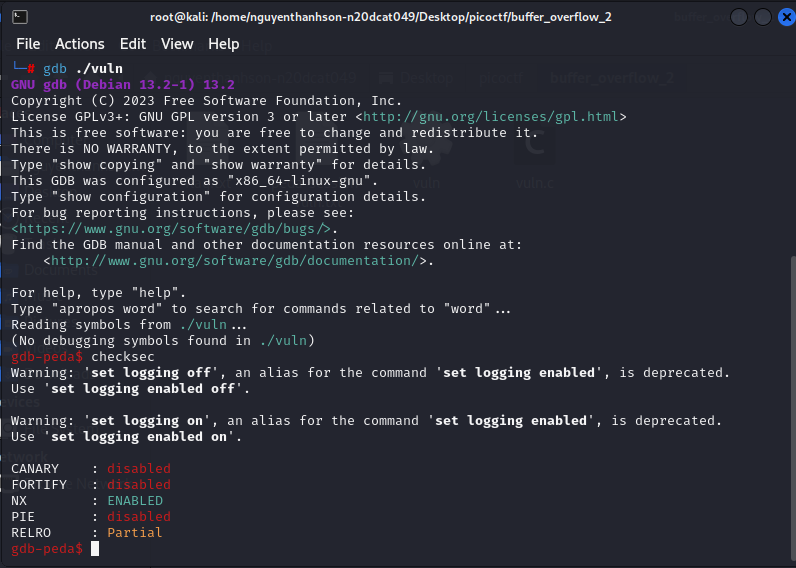
- Chương trình vuln.c.



Sau khi đọc đoạn mã chương trình vuln.c, ta thấy hàm vuln() có sử dụng hàm gets() có thể khai thác tràn bộ nhớ đệm, hàm win() sẽ là hàm đưa flag được lấy từ tệp flag.txt với điều kiện hai tham số lần lượt là 0xCAFEF00D và 0xF00DF00D.

- Tệp thực thi vuln. 

Là file thực thi 32-bit LSB



*CANARY***:** disabled (không chống các cuộc tấn công tràn bộ nhớ bằng cách thêm "canary values" vào stack frame).

*FORTIFY:* disabled (không cung cấp các hàm thư viện được bảo vệ cao hơn để ngăn chặn lỗ hổng).

*NX:* ENABLE (ngăn chặn việc thực thi mã trên các phần của bộ nhớ được đánh dấu là không thực thi).

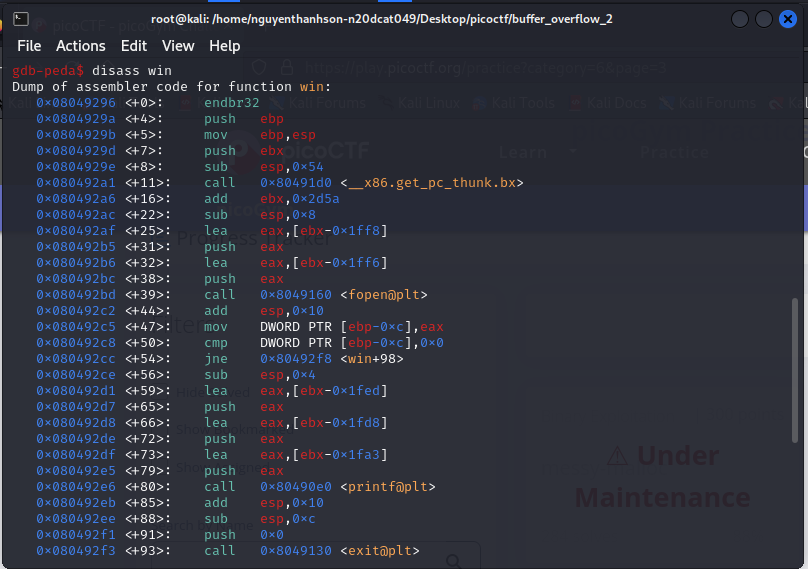
*PIE:* disabled (không Randomize địa chỉ bộ nhớ cho các phần của chương trình để làm khó khăn hơn cho kẻ tấn công tìm và tận dụng lỗ hổng).

*RELRO:* partial (chỉ bảo vệ một phần của bảng GOT (Global Offset Table) khỏi việc chỉnh sửa sau khi tải chương trình vào bộ nhớ.)

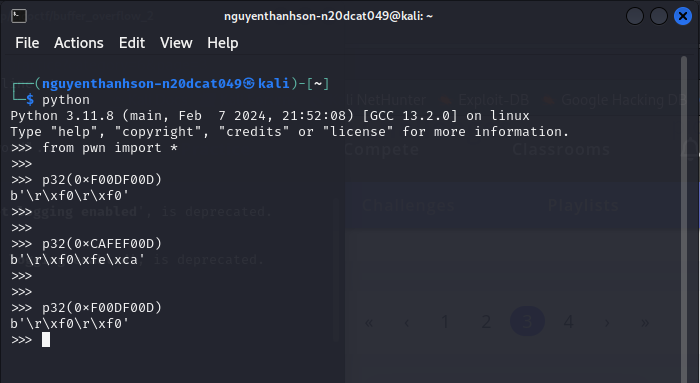
**Ý tưởng:** khai thác lỗ hỏng buffer overflow để ghi đè địa chỉ trả về sau khi thực hiện hàm vunl() bằng địa chỉ của hàm win(), ghi đè giá trị tham số lần lượt là 0xCAFEF00D và 0xF00DF00D.

**Chuẩn bị:**

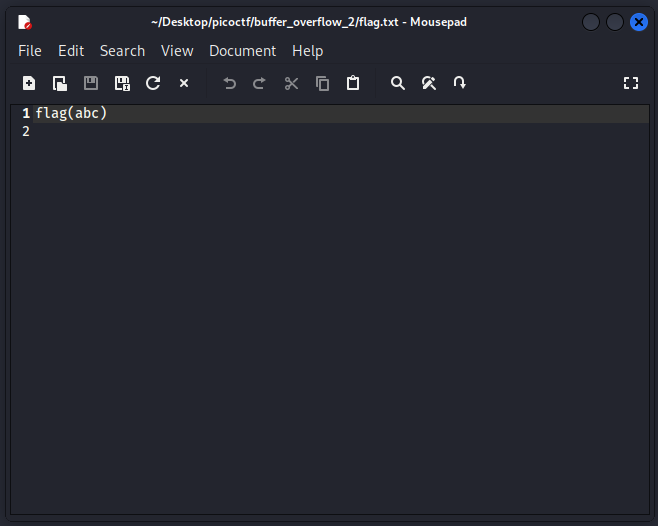
**-** tìm địa chỉ bắt đầu của hàm win()



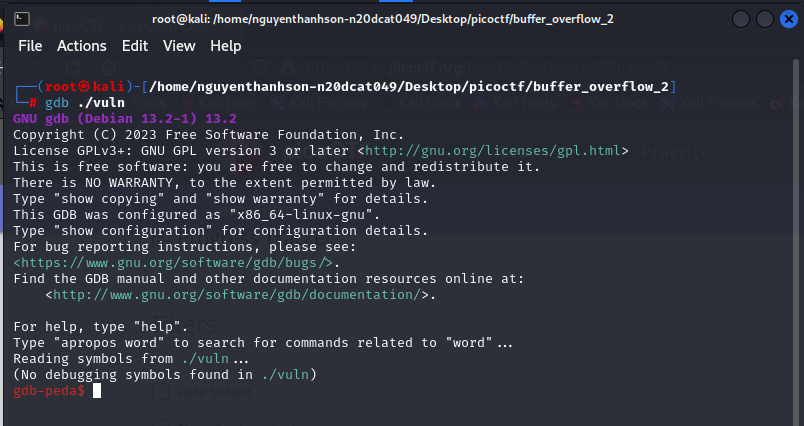
Địa chỉ bắt đầu của hàm win() là 0x08049296, vì chương trình là tệp thực thi 32 bit LSB nên ta phải thực hiện chuyển đổi địa chỉ này về dạng 32 bit (4byte). Tương tự với hai tham số 0xCAFEF00D và 0xF00DF00D.



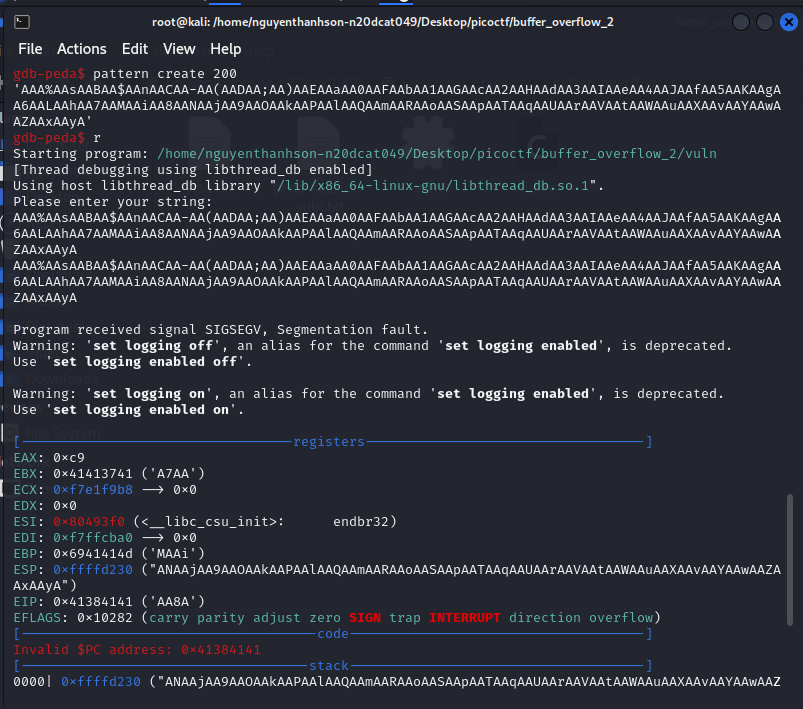
Tạo file flag.txt để lưu flag nhằm kiểm tra kết quả trong lúc khai thác



**Khai thác:** sử dụng extension peda cho gdb thực hiện khai thác chương trình vuln



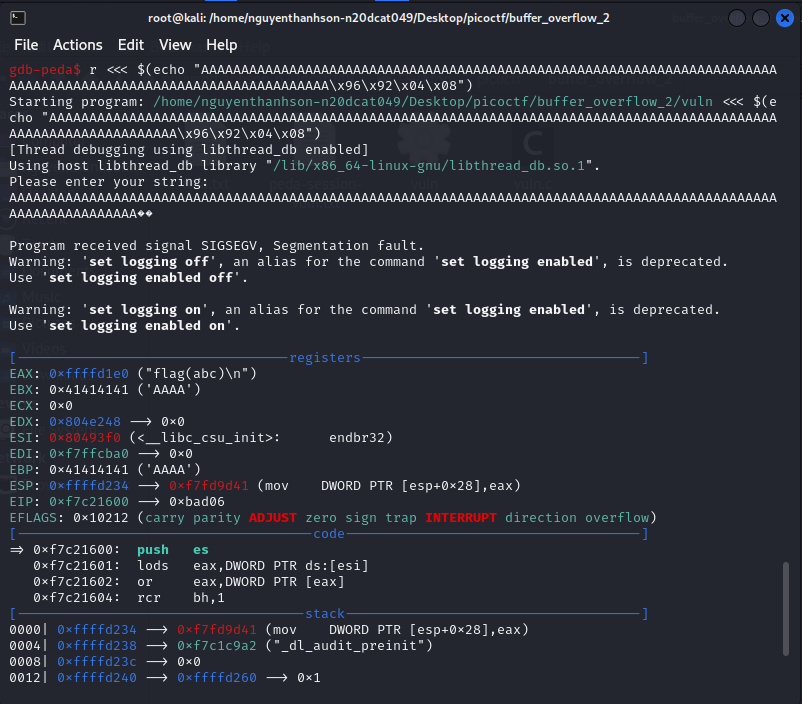
Tạo chuỗi 200 kí tự ngẫu nhiên để nhập thông tin cho hàm vuln() để tìm ra vị trí lưu địa chỉ trả về sau khi thực hiện hàm vuln()



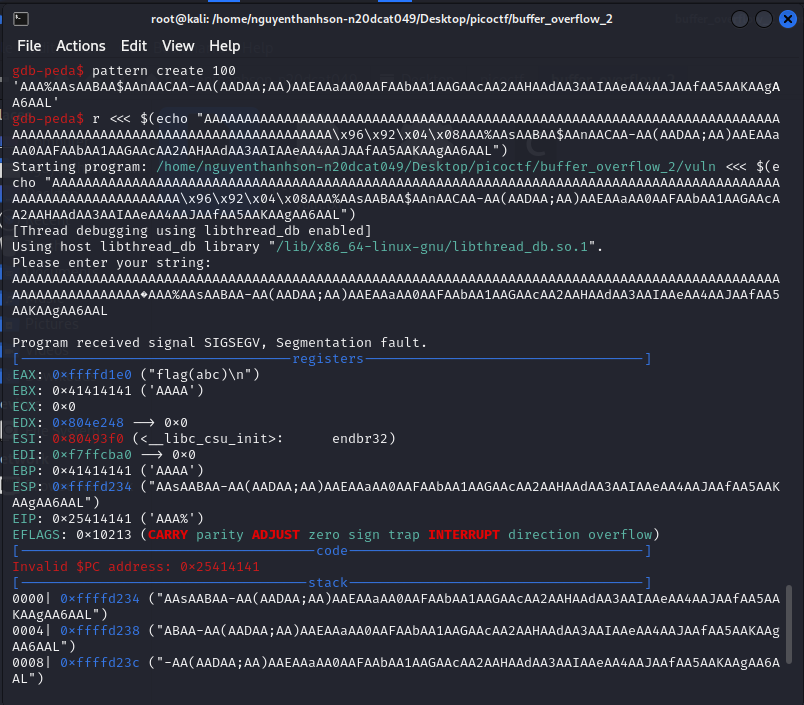
Ta thấy giá trị địa chỉ trả về đã được ghi đè bởi chuỗi ‘AA8A’ mà chúng ta đưa vào. Tìm vị trí của chuỗi ‘AA8A’ trong chuỗi 200 kí tự được nhập vào ban đầu ta có thể xác định được cụ thể vị trí để chèn địa chỉ hàm win().



Thử chạy lại chương trình và nhập vào chuỗi gồm 112 kí tự + địa chỉ hàm win()



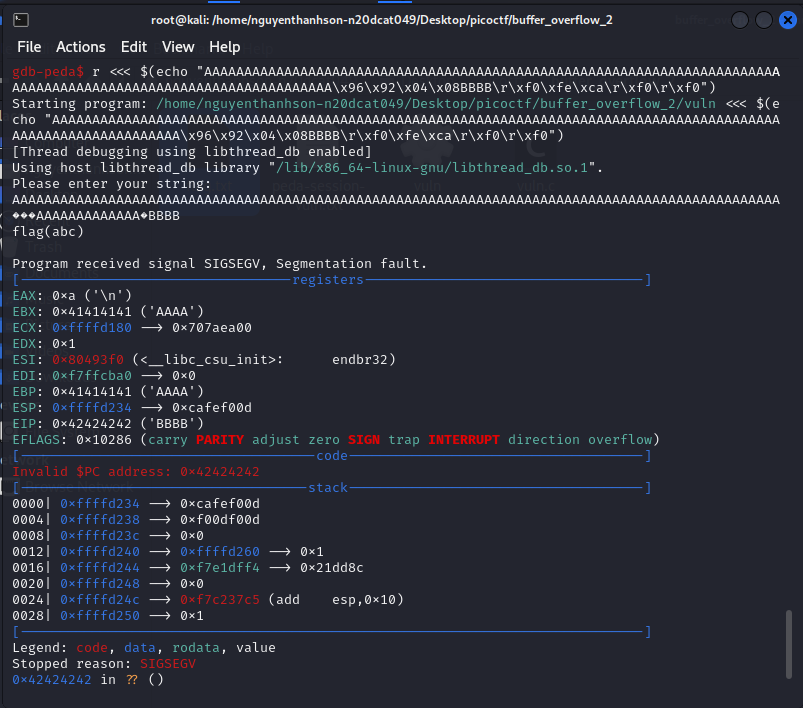
Ta thấy flag đã được lấy và lưu vào thanh ghi EAX tuy nhiên vì tham số vẫn chưa thõa mãn điều kiện nên nó chưa được đưa ra màn hình, vì vậy vẫn chưa thể lấy nó từ server. Tiếp tục tìm kiếm vị trí lưu giá trị của hai tham số của hàm win() trong bộ nhớ bằng cách nhập chuỗi 112 kí tự + địa chỉ hàm win() + 100 kí tự ngẫu nhiên.



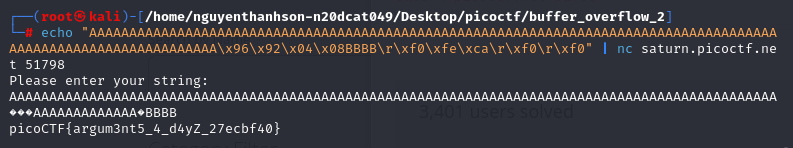
Tìm kiếm vị trí chuỗi ghi đè lên thanh ghi ESP để biết vị trí muốn tìm (thường thì tham số của hàm sẽ được lưu trên stack đầu tiên khi một stack frame được tạo)



Cuối cùng, thử nhập chuỗi 112 kí tự + địa chỉ hàm win() + 4 kí tự + hai giá trị tham số ( ở dạng 32 bit)



Đã lấy được flag tự tạo, thực hiện khai thác trên server để lấy flag của thử thách.



Submit flag, hoàn thành thử thách.

