**Challenge: stackpivot**

- Stack pivot là kỹ thuật thay đổi giá trị của thanh ghi rsp/rbp (trên hệ thống x86\_64) để nó trỏ đến một vùng nhớ khác, thường là vùng bạn có thể điều khiển nội dung. Mục tiêu: Sau khi pivot, bạn dùng các ROP gadget ở vùng nhớ mới để điều khiển luồng thực thi.

- Các chế độ bảo vệ ảnh hưởng đến stackpivot:

A screenshot of a black screen

AI-generated content may be incorrect.

- Với file chall:

A screen shot of a computer program

AI-generated content may be incorrect.

- No canary -> có thể buffer overflow ghi dè đc rip

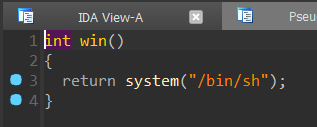
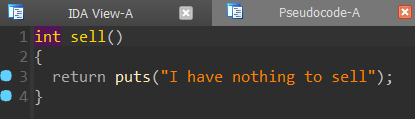
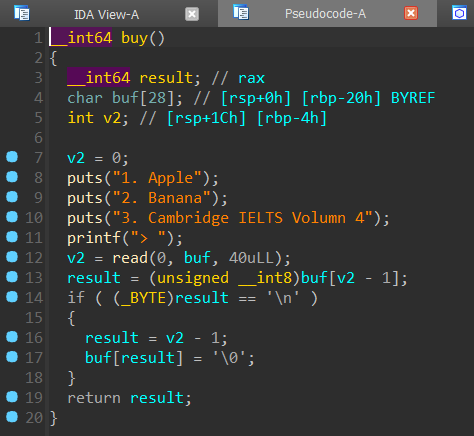
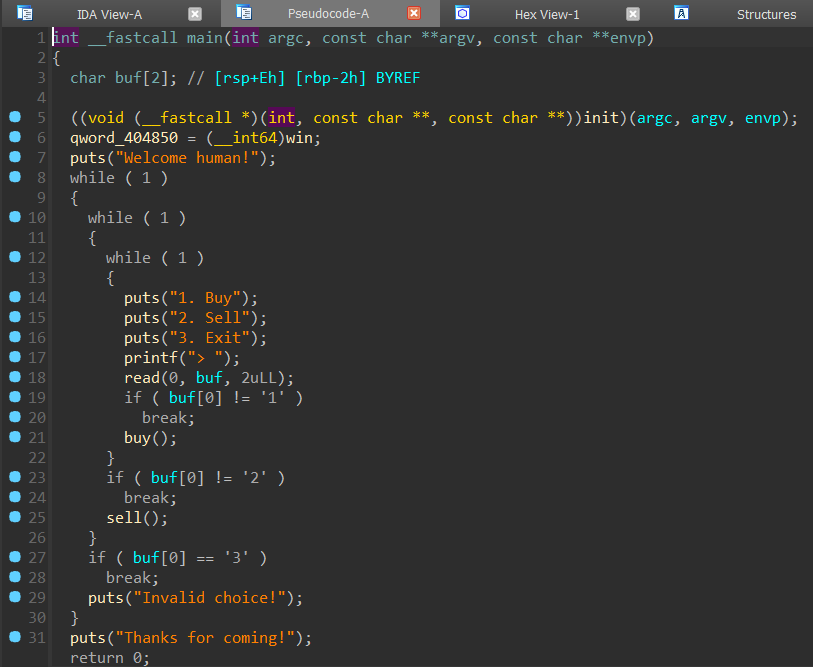
- NX -> ko thực thi đc shellcode trên stack nhưng không ảnh hưởng đến stackpivot

- No PIE -> địa chỉ tĩnh ko cần leak base

**\* Xem pseudo code ở IDA xem, chương trình hoạt động ntn:**

- Chương trình gồm 4 hàm chính: main(), buy(), sell(), win().

Mô tả chương trình hoạt động: chương sẽ cho nhập vô hạn 3 lựa chọn. nếu ta nhập 1 thực hiện hàm buy, trong hàm buy cũng có 3 lựa chọn 1 2 3, cho nhập 1 chuỗi 28 kí tự rồi kết thúc hàm. Ta nhập 2 sẽ thực hiện hàm sell() in ra 1 chuỗi “I have nothing to sell” rồi kết thúc hàm. Nếu ta chọn 3 thì kết thúc hàm main.



-Mục tiêu: ta phải chiếm được hàm win()

**\* Nhận diện stackpivot:**

**-** Nhận thấy trong hàm buy() chuỗi buf khai báo 28 kí tử mà cho nhập 40 kí tự -> lỗi buffer overflow

- Thử nhập tối đa 40 byte rác vào buf thử xem có bị overwrite cái nào ko

A computer screen shot of a program code

AI-generated content may be incorrect.

-> Ta thấy rbp bị thay đổi còn rip thì không

- Do rbp không phải 1 địa chỉ hợp lệ vì khi lấy biến ra để để ghi dữ liệu thì bị lỗi

- Vậy ta phải thay đổi rbp thành 1 giá trị hợp lệ thì ta sẽ dùng bất kì địa chỉ nào mà ta biết do No PIE nên các địa chỉ là tĩnh nên ta có thể thay thành các địa chỉ trong file binary.

**\* Viết script:**

- Đầu tiên ta sẽ tìm offset từ buf đến rbp bằng cyclic:  
**A screenshot of a computer

AI-generated content may be incorrect.**

- Sau đó ta sẽ tìm 1 địa chỉ trong file có quyền write. Ta thấy có địa chỉ 0x404000 có thể read và write đc nên ta dùng lệnh x/50xg 0x406000 để tìm 1 địa chỉ trống chưa có giá trị để write vào.

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**->** Chọn bừa địa chỉ 0x404880 để debug xem như nào

- Ta có script:  
A computer screen shot of text

AI-generated content may be incorrect.

- Giờ thử debug động quan sát:  
A screen shot of a computer code

AI-generated content may be incorrect.A screenshot of a computer code

AI-generated content may be incorrect.

-> Khi vẫn còn ở trong hàm buy khi mình gửi liệu rồi thì cái rbp nó lại chính là địa chỉ rbp của hàm main -> tức là chương trình cho phép ta overwrite đc rbp của chính hàm main luôn

A screenshot of a computer screen

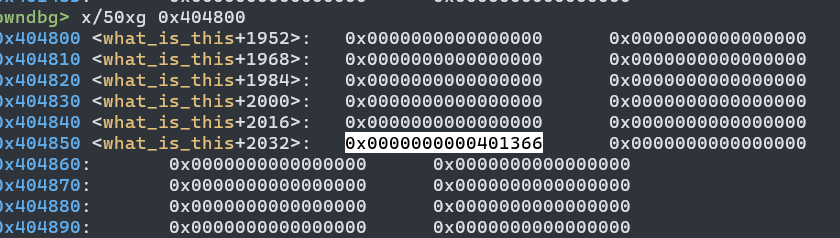
AI-generated content may be incorrect.

- khi read xong thì ta thấy địa chỉa rbp của hàm main bởi overwrite giống địa chỉ mà mình gửi trong payload

\* A black background with yellow and orange text

AI-generated content may be incorrect.-

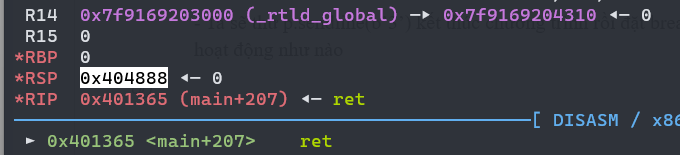
Nhìn lại ở đầu hàm main trong ida ta thấy hàm win đc khai báo vào trong con trỏ qword\_404850. Địa chỉ 0x404850 cũng là 1 địa chỉ của file binary luôn. Giờ ta sẽ đi kiểm tra



-> 0x404850 chính là địa chỉ của hàm win.

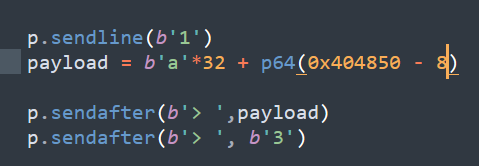
-> Ý tưởng: mình sẽ tìm cách ret ở hàm main vào đúng địa chỉ của hàm win

- Ta sẽ thử p.sendline(b’3’) kết thúc chương trình rồi đặt breakpoint ở đúng return xem chương trình hoạt động như nào



-> ta thấy địa chỉ tại rsp đã bị đẩy lên 8 byte (0x404880 -> 0x404888) và do ở xunh quanh địa chỉ 0x404880 ko có dữ liệu gì hết nên chương trình sẽ exit mà ko thực hiện gì

-> nên giờ ta sẽ lấy cái địa chỉ hàm win ta tìm đc kia sau đó trừ đi 8 byte để sau khi chạy qua leave đến ret thì ta sẽ ret hàm đúng địa chỉ của hàm win, nên ta có script cuối cùng

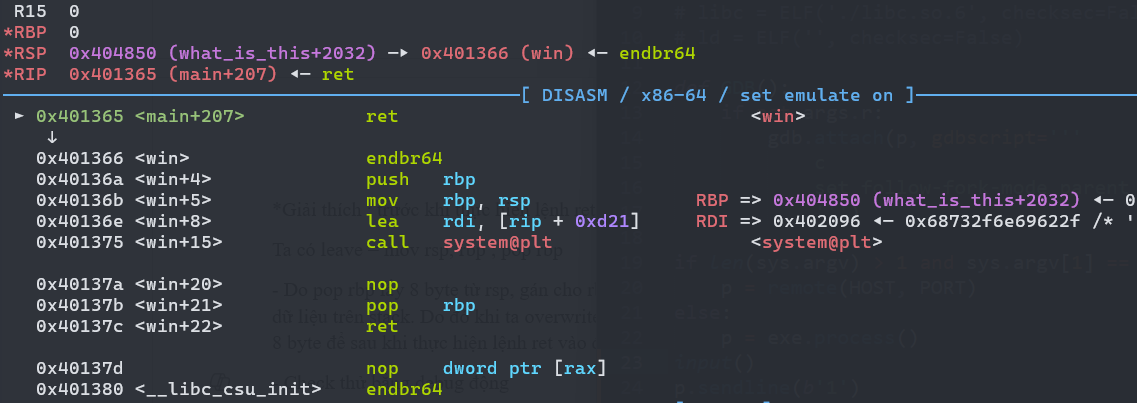


\*Giải thích : trước khi thực hiện lệnh ret ta thực hiện lệnh leave.

Ta có leave = mov rsp, rbp ; pop rbp

- Do pop rbp lấy 8 byte từ rsp, gán cho rbp và sau đó rsp += 8, nên tổng cộng hàm leave mất 8 byte dữ liệu trên stack. Do đó khi ta overwrite đc rbp của main thì ta sẽ pass địa chỉ chỉ của hàm win – đi 8 byte để sau khi thực hiện lệnh ret vào đúng hàm win.

- Check thử bằng debug động

-> Chính xác

- Giờ gửi script xem chiếm đc shell trên LOCAL chưa:

A screen shot of a computer

AI-generated content may be incorrect.

-> Đã chiếm đc shell