BÁO CÁO THỰC HÀNH

**Môn học: Cơ chế hoạt động của mã độc**

**Kỳ báo cáo: Buổi 05 (Session 05)**

**Tên chủ đề: ROP Explanation**

*GVHD: Nghi Hoàng Khoa*

1. **THÔNG TIN CHUNG:**

Lớp: NT230.L21.ANTT

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** | **Email** |
| 1 | Nguyễn Công Tiến | 18520165 | 18520165@gm.uit.edu.vn |
| 2 | Nguyễn Đức Phú | 18520333 | 18520333@gm.uit.edu.vn |

* Link source code:

https://github.com/ngdph/NT230.L21.ANTT/tree/main/Lab%2005

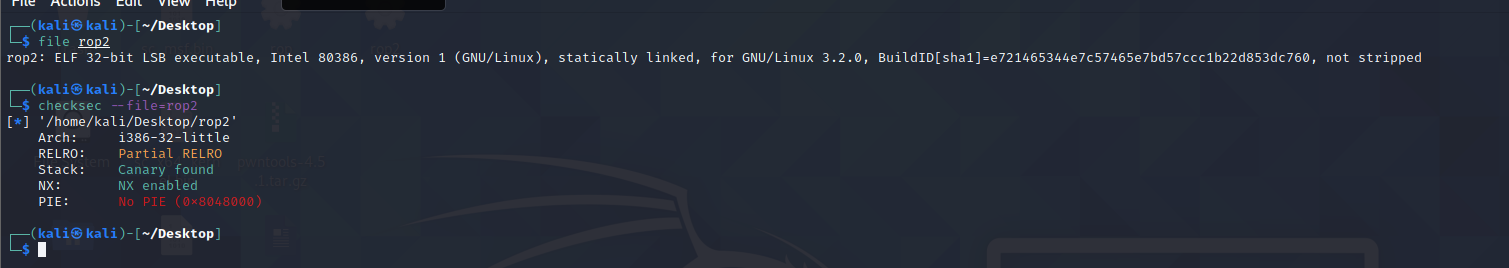
1. **NỘI DUNG THỰC HIỆN:**

**ROPBasic:**

Đã thực hiện và báo cáo tại lớp.

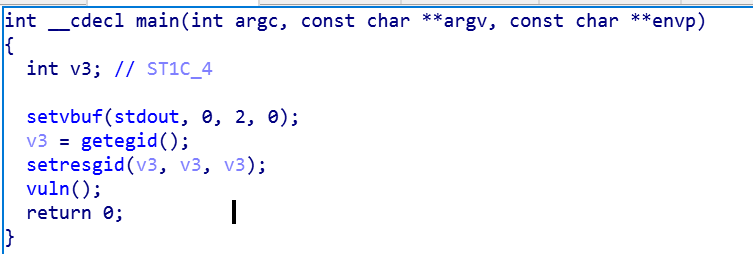
**ROP2:**

Thực hiện kiểm tra file rop2 bằng lệnh file và checksec:

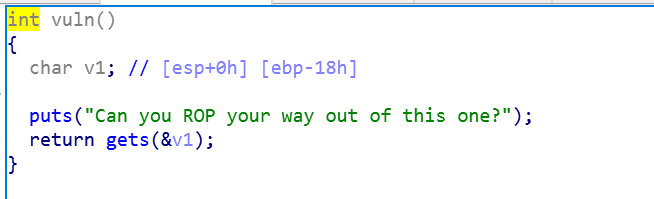


Ta thấy được flag NX (non-executetable) được bật lên 🡺 Cần thực hiện tấn công với kỹ thuật ROP.

Xem pseudo-code bằng IDA:



Ta thấy chương trình gọi tới hàm vuln(). Tiếp tục xem hàm vuln():

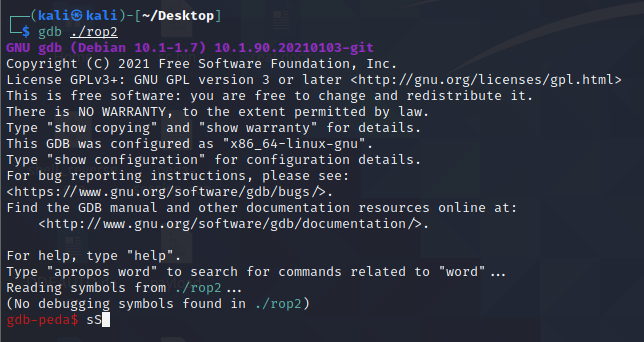


Ta thấy được chương trình sử dụng hàm gets trong vuln().

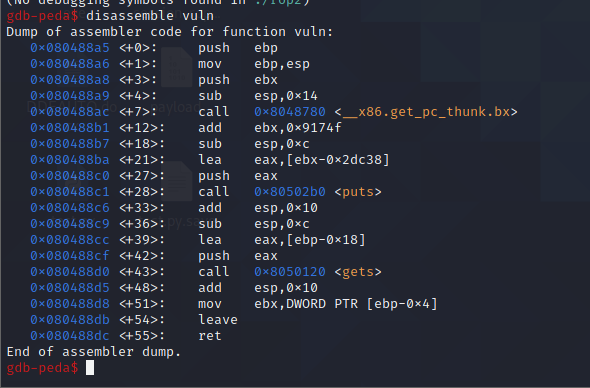
Ý tưởng cho cuộc tấn công: tấn công buffer overflow, padding làm tràn hàm gets đến hàm return của vuln().

* Tìm padding:

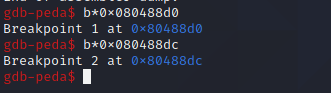
Sử dụng gdb để có thể debug file rop2:

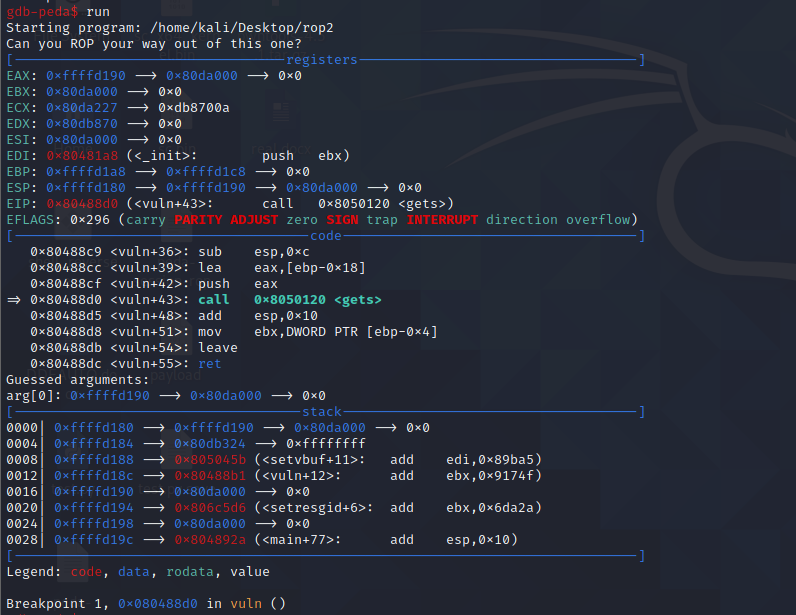


Vì hàm gets nằm trong hàm vuln() nên ta sẽ thực hiện disassemble hàm vuln():

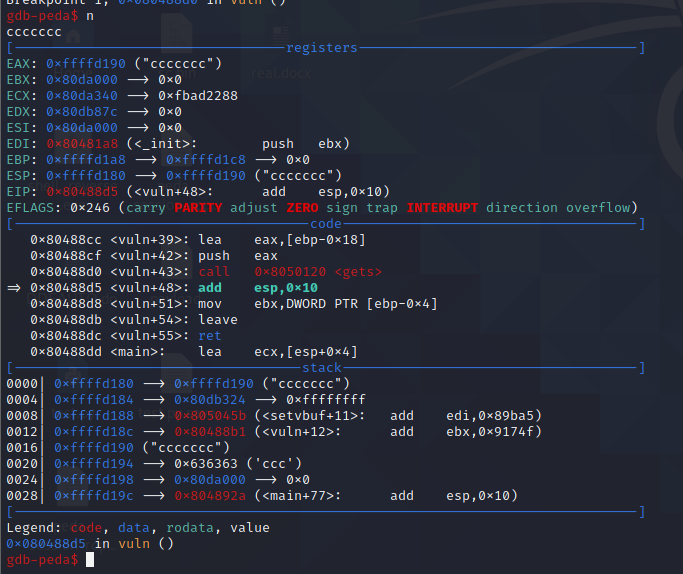


Đặt breakpoint ở gets và ret rồi thực hiện chạy debug:



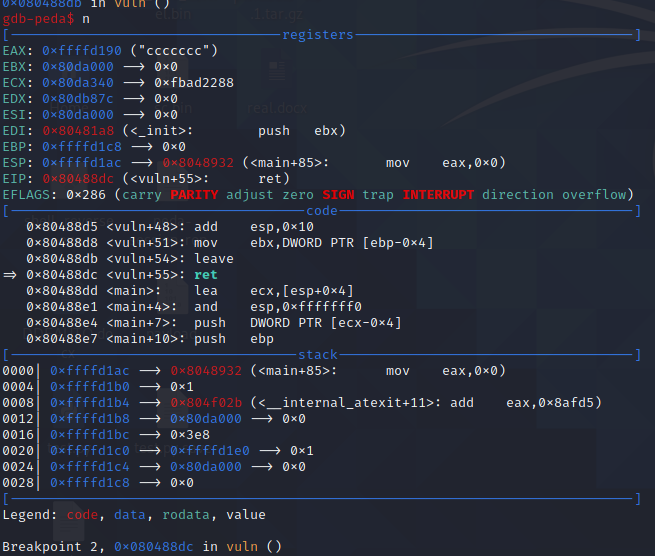


Gõ “n” để tiếp tục chạy debug. Lúc này ta cần nhập input:



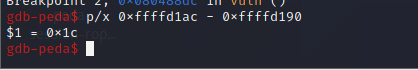
Ta thấy được địa chỉ của hàm gets là 0xffffd190.

Tiếp tục gõ “n” cho đến khi con trỏ chương trình trỏ tới ret:



Ta thấy được địa chỉ của hàm ret là 0xffffd1ac.

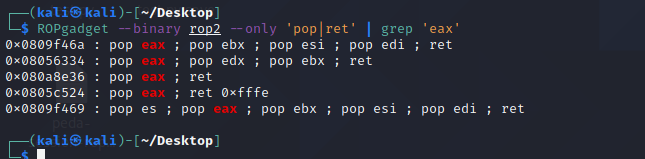
Thực hiện trừ 2 địa chỉ đã tìm thấy để tìm lượng cần padding:



Ta cần phải 0x1c tức 28 bytes cần được padding.

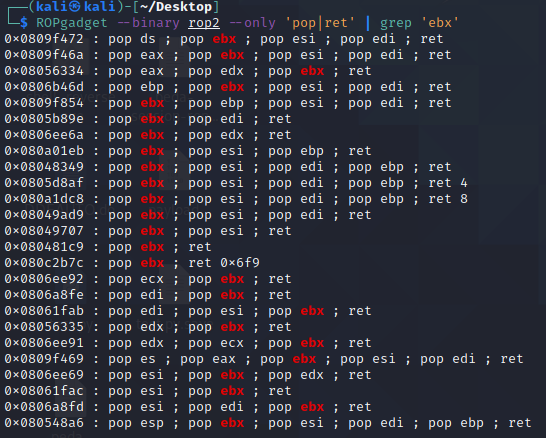
Payload thực hiện khai thác: execve("/bin/sh",NULL,NULL) 🡺 execve eax ebx ecx edx. Ta cần xác định các giá trị của eax, ebx, ecx và edx. Trong đó, ta có eax = 0x0b, ecx và edx là NULL. Ngoài ra ta cần xác định hàm interrupt (int 0x80)

Kiểm soát giá trị cho eax:

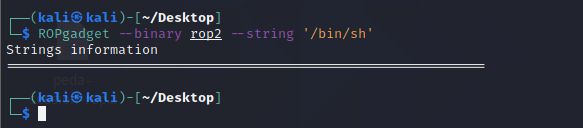


Khi ret trỏ đến địa chỉ 0x080a8e36 : pop eax ; ret nó sẽ thực thi lệnh pop lấy giá trị tại esp trỏ tới gán cho eax sau đó tiếp tục thực hiện ret. Các gadget khác cũng tìm kiếm theo cách tương tự.

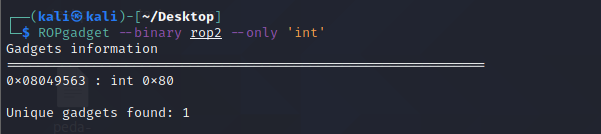
Trong lúc xác định ebx, tình cờ ta có đủ cho edx và ecx: 0x0806ee91 : pop edx ; pop ecx ; pop ebx ; ret



Chuỗi “/bin/sh” không có sẵn trong file:



Gadget quan trọng nhất int 0x80:

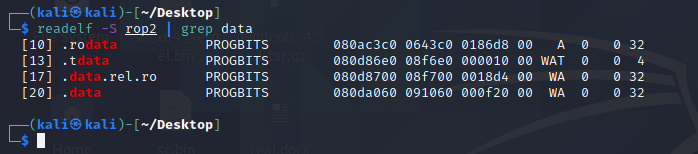


Ta còn thiếu chuỗi “/bin/sh”.

Để có được chuỗi “/bin/sh”, ta chèn chuỗi “/bin/sh” này vào một section nào đó có thể ghi được (Writable).

Nhóm tham khảo cách thực hiện tại đây: <https://failingsilently.wordpress.com/2017/12/14/rop-chain-shell/>

Đầu tiên ta cần các định section có thể writable. Ta biết được rop2 là một file elf. Dùng lệnh readelf để xác định section:

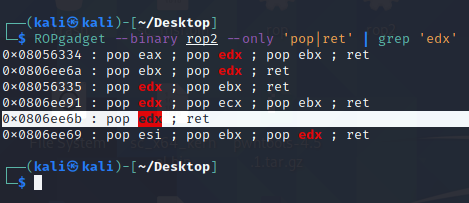


* Thực hiện chèn chuỗi “/bin/sh”:

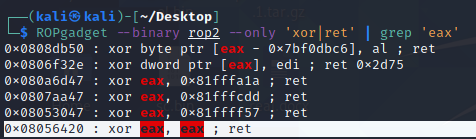
**Ý tưởng:** Đầu tiên pop thanh ghi edx rồi gán địa chỉ của section đã chèn chuỗi “/bin/sh” ở đây là section .data. Để thực hiện việc chèn chuỗi “/bin/sh”, ta pop thanh ghi eax rồi thực hiện lưu giá trị eax là “/bin/sh”. Tuy nhiên mỗi lần ta chỉ thực hiện lưu được 4 byte nên ta sẽ tách chuỗi cần chèn thành “/bin” và “//sh”. Sau khi lưu được “/bin” ta cần đưa chuỗi tới địa chỉ mà thanh ghi edx đang trỏ tới ở đây chính là section .data. Sau khi lưu chuỗi “/bin”, ta tiếp tục ghi chuỗi”//sh” tương tự. Sau khi lưu xong chuỗi “/bin/sh”, ta sẽ khi thêm giá trị NULL để báo hiệu kết thúc chuỗi.

**Với ý tưởng trên, ta thực hiện tìm các gadget sau:**

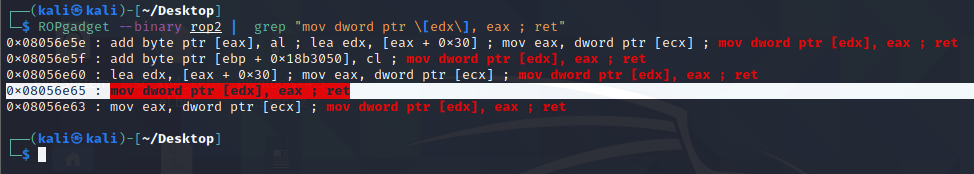
Pop edx ; ret ( Load thanh ghi edx từ stack để gán địa chỉ section .data)



xor eax, eax ; ret (để lấy giá trị 0)



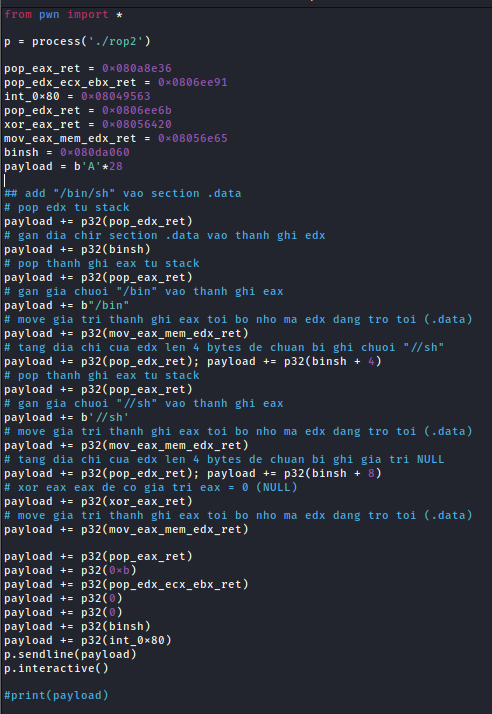
mov dword ptr [edx], eax ; ret (đưa giá trị eax vào ô nhớ có địa chỉ edx trỏ tới):



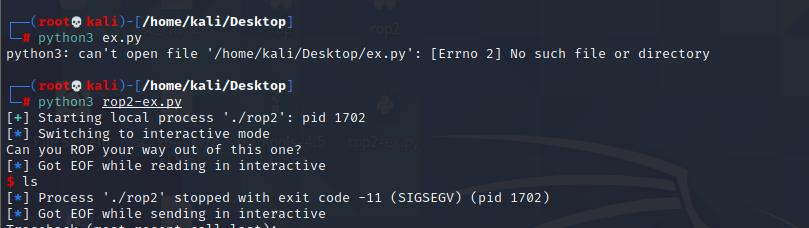
**Như vậy payload của ta sẽ như sau:**

"c"\*28 + [pop edx ; ret] + "\x60\xa0\x0d\x08" + [pop eax ; ret] + "/bin" + [mov dword ptr [edx], eax ; ret] + [pop edx ; ret] + "\x64\xa0\x0d\x08" + [pop eax ; ret] + "//sh" + [mov dword ptr [edx], eax ; ret] + "\x68\xa0\x0d\x08" + [xor eax, eax ; ret] + [mov dword ptr [edx], eax ; ret] + [pop eax ; ret] + "\x0b" + [pop edx ; pop ecx ; pop ebx ; ret] +"\x00\x00" + ["\bin\sh"] + [int 0x80]

**Ta có được code python để khai thác như sau:**

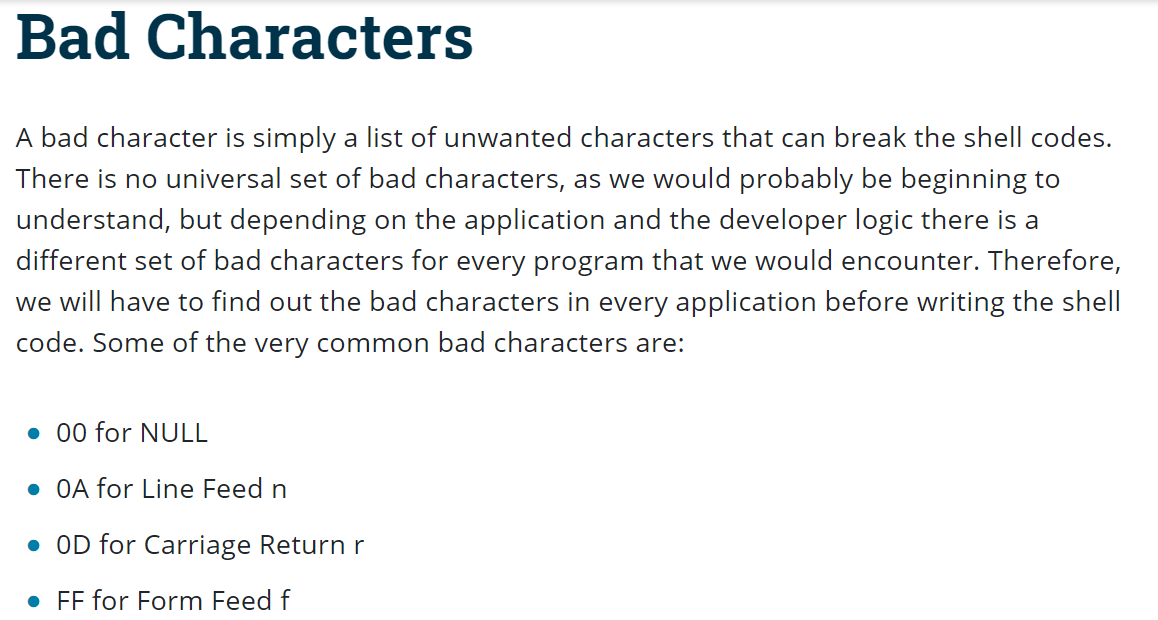


**Tuy nhiên khi chạy lại bị lỗi:**



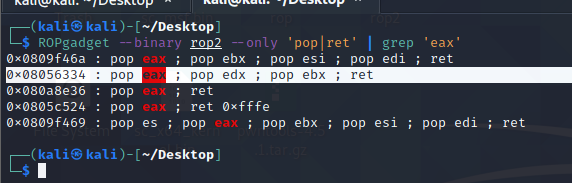
Sau khi tìm hiểu thì có thể shellcode đã bị dừng do bad character. Tham khảo tại bài viết sau: <https://resources.infosecinstitute.com/topic/stack-based-buffer-overflow-in-win-32-platform-part-6-dealing-with-bad-characters-jmp-instruction/>

**Các bad character cần tránh:**



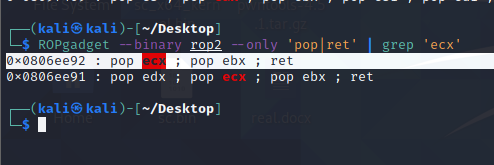
Sau khi xem kĩ lại các gadget thì thấy gadget 0x080a8e36 : pop eax ; ret có chứa /x0a. Vì vậy ta cần tìm một gadget khác không chứa bad character.

Gadget có pop eax mà nhóm tìm được:

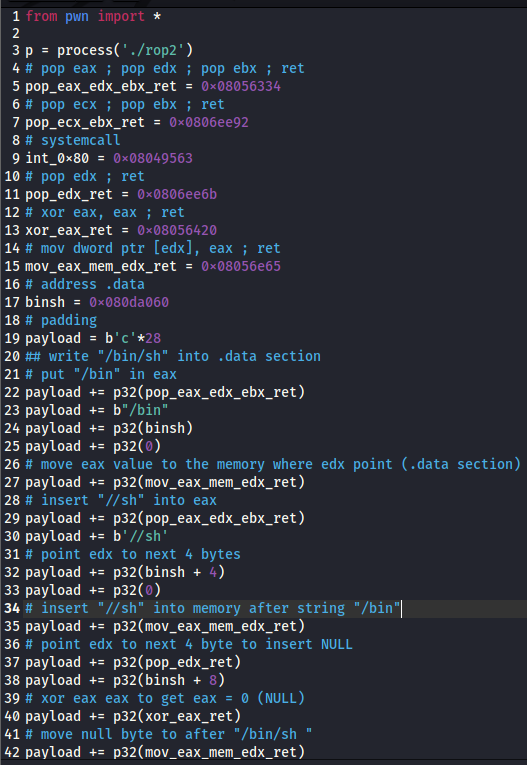


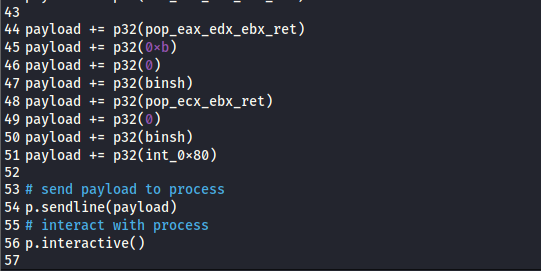
Lúc này gadget có cả thanh ghi edx và ebx.

Tìm gadget có pop ecx:

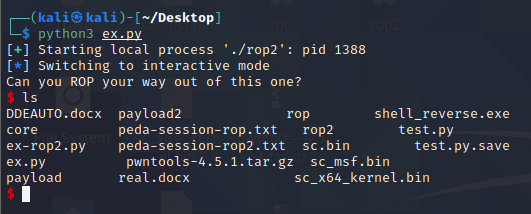


Lúc này, ta cần phải chỉnh sửa shell code như sau:



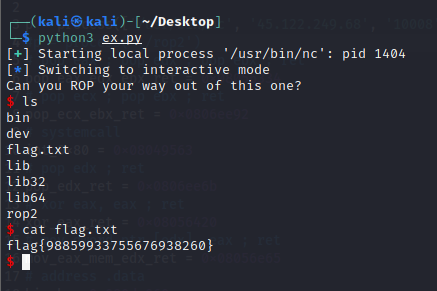


Chạy code thì ta thu được kết quả như sau:



🡺 Thực hiện chiếm shell thành công.

**Thực hiện lấy flag từ trang 45.122.249.68:10008**

****

Nhóm thu được flag như sau: flag{98859933755676938260}