**Bài tập về nhà**

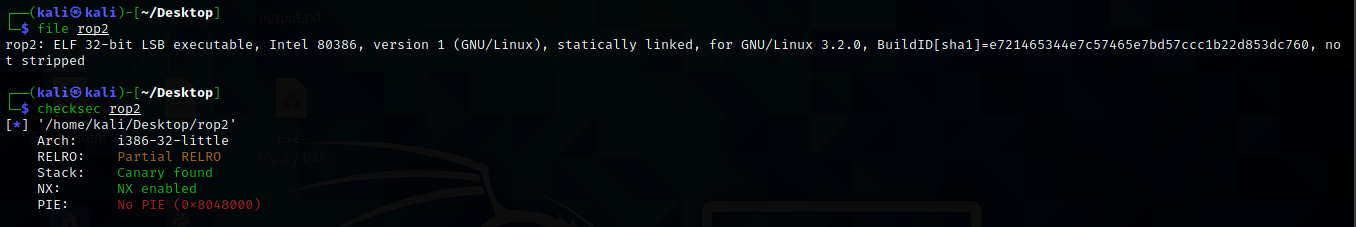
* **B. ROPBasic**

Đã làm trên lớp.

* **C. ROP2**

**Chiếm shell (/bin/sh) thông qua rop2**

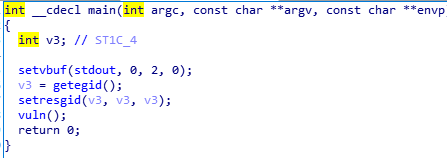
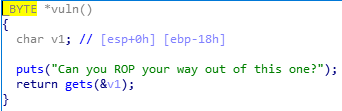
Kiểm tra file rop2:



Ta thấy được flag NX (Non Executable) có giá trị là Enabled nghĩa là code ta chèn vào stack sẽ không được thực thi

* Phải tấn công dạng return-oriented programming

Xem mã giả bằng IDA Pro

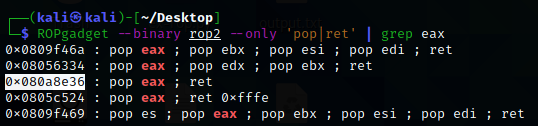
Ta thấy được chương trình dùng hàm gets trong vuln()

* Ý tưởng: tấn công buffer overflow, padding hàm gets đến hàm return của vuln(), ghi đè địa chỉ một chuỗi gadget vào stack để điều hướng chương trình như mong muốn

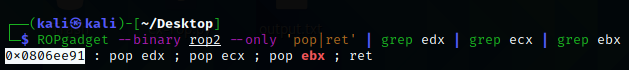
Tìm kiếm các chỗi gadget cần thiết để sử dụng:

Để gọi systemcall ta cần chỉnh giá trị của eax, ebx, ecx, edx (eax = 11, ebx chứa địa chỉ của chuỗi “/bin/sh”, ecx = edx = 0) và cần hàm interrupt (int 0x80)

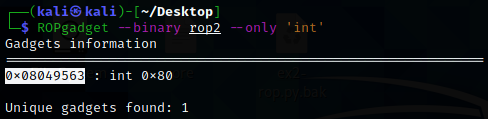
pop eax ; ret



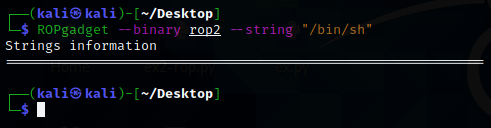
pop edx ; pop ecx ; pop ebx ; ret



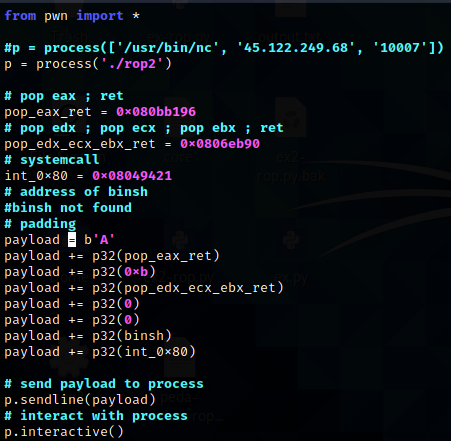
int 0x80



Chuỗi “/bin/sh” (Không có sẵn)

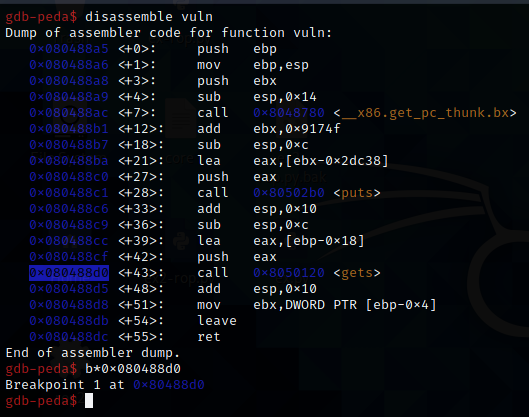


Chỉnh sửa code python đơn giản được cung cấp sẵn ở bài ROPBasic và thay các giá trị mới vào ta được đoạn code sau:



Tìm padding: (gdb rop2)

Xem code hàm vuln() (do gets() nằm trong vuln() đã biết khi xem mã giả bằng IDA)

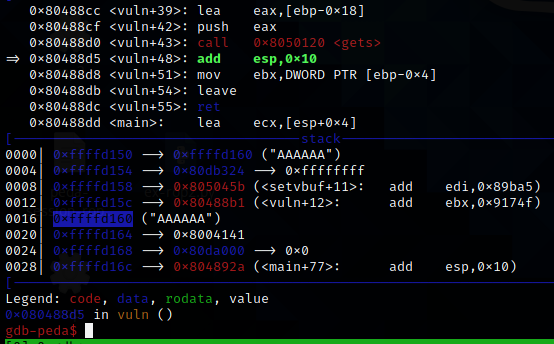


Đặt breakpoint ở hàm gets và bắt đầu chạy

Tiếp tục chọn next (n), chương trình yêu cầu cung cấp input

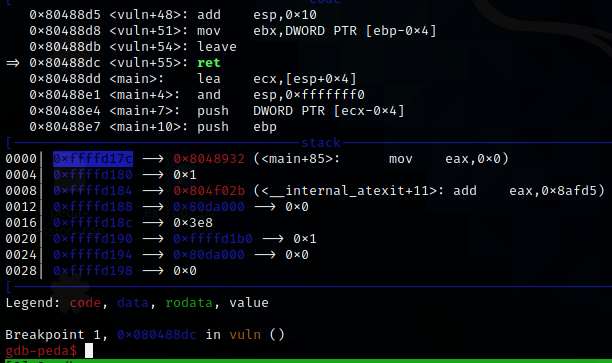


Sau khi nhập input, ta thấy input xuất hiện tại địa chỉ 0xffffd160 như hình dưới:



Đây chính là địa chỉ lưu input của ta

Nhấn next tiếp tục đến khi con trỏ chương trình trỏ tới ret

Khi chạy ret, chương trình sẽ pop ra địa chỉ tại đỉnh stack và thực hiện lệnh tương ứng địa chỉ này

Do đó đỉnh stack lúc này chính là nơi ta sẽ ghi đè địa chỉ của gadget đầu tiên vào

Trừ 2 địa chỉ đã tìm thấy



0x1c = 28 (đơn vị) (x32 => 28 bytes)

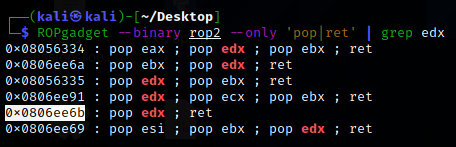
* Vậy tìm được padding là 28

Như vậy ta vẫn còn thiếu chuỗi “/bin/sh”

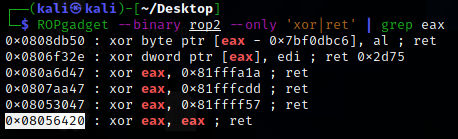
* Ý tưởng: chèn chuỗi “/bin/sh” này vào một section nào đó có flag W (Writable) rồi sử dụng

Để làm được việc trên, ta cần các gadget sau:

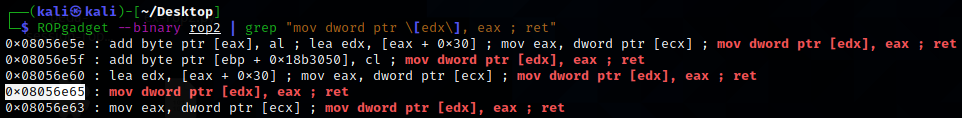
pop edx ; ret



xor eax, eax ; ret (trả eax về 0)

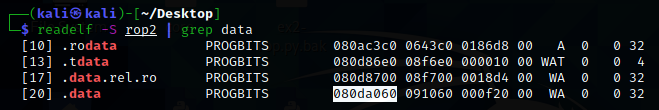


mov dword ptr [edx], eax ; ret (đưa giá trị eax vào ô nhớ có địa chỉ edx trỏ tới)

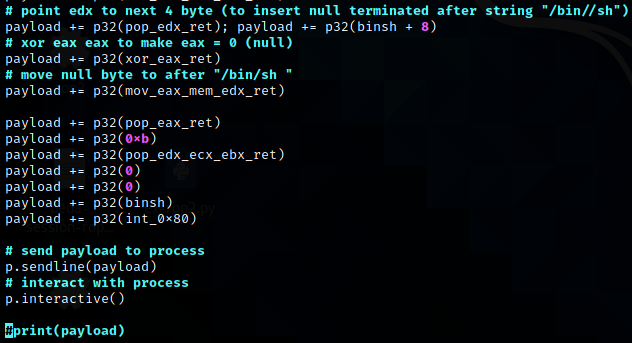
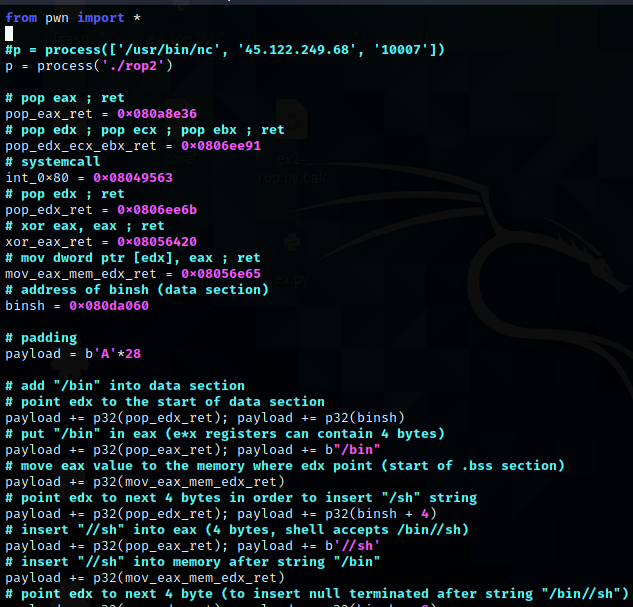


Ý tưởng: trỏ edx đến ô nhớ muốn lưu chuỗi “/bin/sh” -> lưu một phần của chuỗi này vào eax -> đưa giá trị eax vào ô nhớ có địa chỉ chứa trong edx -> dịch edx tới vị trí tiếp theo và tiếp tục đến hết chuỗi.

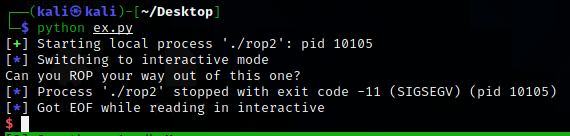
Ta sẽ lưu chuỗi này vào section data tại địa chỉ bên dưới (do data có flag W)



Như vậy ta được code sau (ex.py):

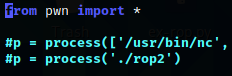
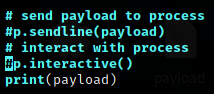


Chạy thử bị lỗi



Bắt đầu debug

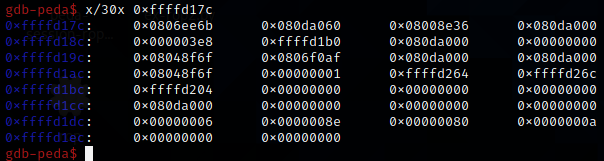
Sửa lại code, code chỉ in ra payload không tạo ra cũng như interact with new process. Dùng payload này debug bằng gdb

Mở gdb rop2, đặt breackpoint tại hàm ret của vuln() rồi chạy lệnh sau



Dùng lệnh dưới để xem 30 mục trong stack tính từ địa chỉ 0xffffd1ac (đây chính là địa chỉ sau phần padding, xem lại phần tìm padding)



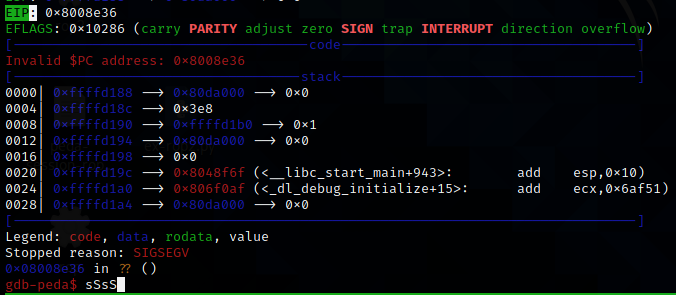
Ta thấy các lệnh sau

0x0806ee6b pop edx ; ret (với 0x080da060 là giá trị được pop ra từ stack)  
0x080da060

0x08008e36 Không tìm thấy giá trị này trong payload, ta không hề chèn giá trị này vào payload. Xem kĩ lại payload thấy được 0x080a8e36 là gadget pop eax ; ret

* Như vậy trong quá trình truyền dữ liệu đã xảy ra lỗi làm sai lệch dữ liệu
* Dự đoán: 0x0a chính là kí tự “\n” nhưng khi được chèn vào trong process, có thể process nhận ký tự này thành null (giá trị 0x00)

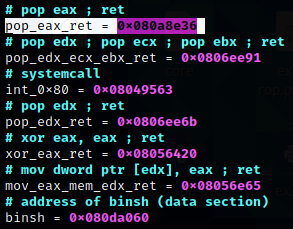
Tiếp tục nhấn next (n)



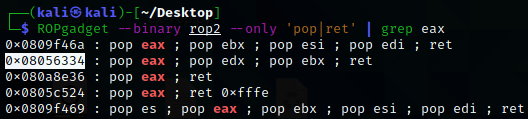
Địa chỉ 0x8008e36 có giá trị 0 => ngắt chương trình

Vậy để khắc phục, ta tránh dùng các địa chỉ gadget có chứa byte 0x0a

Kiểm tra lại payload, ta thấy pop\_eax\_ret là gadget duy nhất chứa byte 0x0a

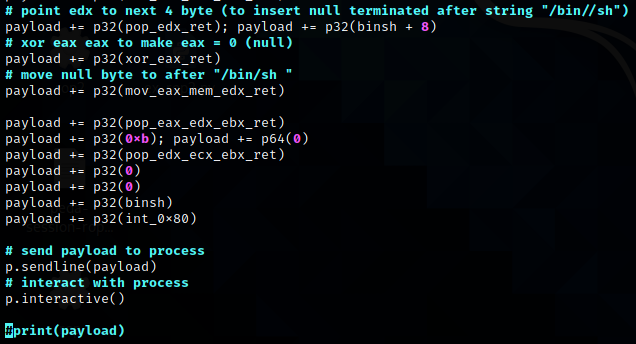
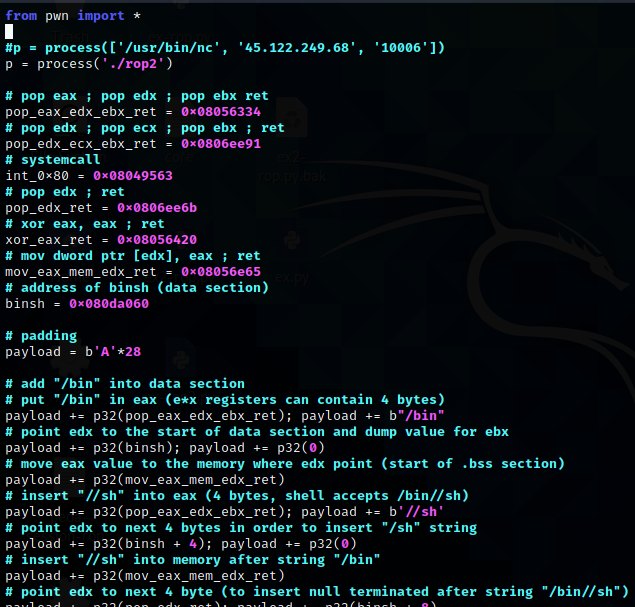


Tìm một gadget khác thay thế:

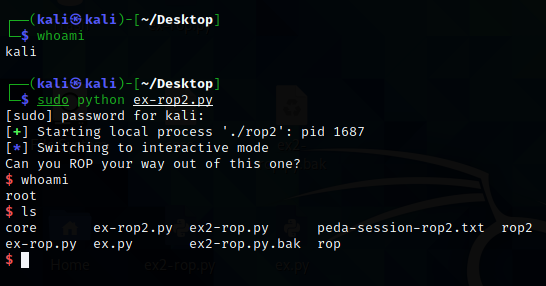


Đây là gadget phù hợp nhất

Như vậy code cần được sửa lại như sau (ex-rop2.py):



Chạy lại chương trình:



Thành công chiếm được shell

Do khi nhóm em làm, server tại 45.122.249.68:10006 không hoạt động nữa nên không thể kiếm flag