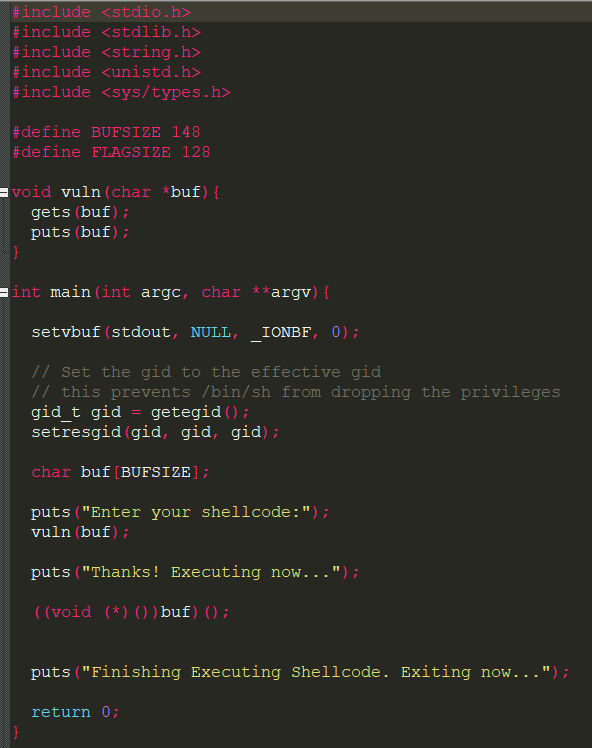
**Handy shellcode:**

Ta thấy đề bài cho 2 file vuln.c và file vuln . Download về file vuln.c để phân tích xem bài này làm gì ?



1. Tìm lỗi và định hướng :

Phân tích hàm main:

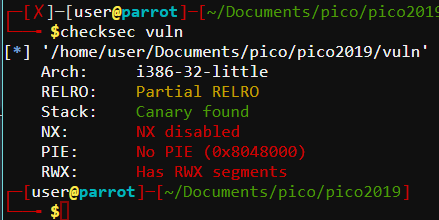
* 3 dòng đầu tiên không quan trọng với bài này, ta bắt đầu với char buf[Bufsize], chương trình khai báo 1 mảng kí tự có bufsize = 148, sau đó gọi hàm vuln và cho buf làm tham số.
* Sau khi ra khỏi vuln, nó lại gọi cái buf của mình như một hàm, tức là nó thực thi cái chuỗi buf đấy. Vậy nếu chuỗi buf là 1 chuỗi gọi shell thì có thể chúng ta có thể gọi một cái shell để cat flag.txt. < hướng 1>

Phân tích hàm vuln:

* Đầu tiên cho gets(buf) , nhập vào buf.
* Sau đó puts(buf) in ra buf.
* Để ý rằng khi nhập vào, thì hàm gets không kiểm gia độ dài được nhập, tức là hàm này gây ra lỗi bufferoverflow. Lỗi này có thể khiến chúng ra sửa đổi địa chỉ return của hàm vuln hoặc thậm chí hàm main để quay về một địa chỉ mà chúng ta mong muốn. <hướng 2 >

1. Loại trừ hướng thừa và khai thác lỗi:

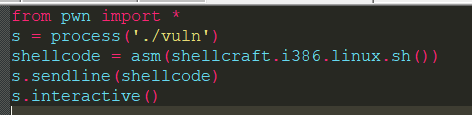
Để loại trừ hướng thừa, ta kiểm tra các ràng buộc bảo vệ của file.



* Chúng ta có not excutable disable, có no pie, và có has rwx segments, có vẻ là điều kiện cho tốt cho hướng 2.
* Chúng ta có canary found, tức là bài này đã bị chặn chống tràn canary, nếu chúng ta không thể làm tràn thì chúng ta không dùng hướng 2 được.
* Quay lại hướng 1. Rất đơn giản, chương trình gọi buf, vậy chỉ cần đưa 1 đoạn shellocde vào buf, chương trình sẽ thực thi và gọi 1 shell lên cho chúng ta.

1. Bắt đầu khai thác :

* Code:



1. Kết quả:

