1. **KỊCH BẢN 1:** **”Buffer Overflow 0”**
   1. **Sơ đồ**

- Mô tả A screenshot of a computer

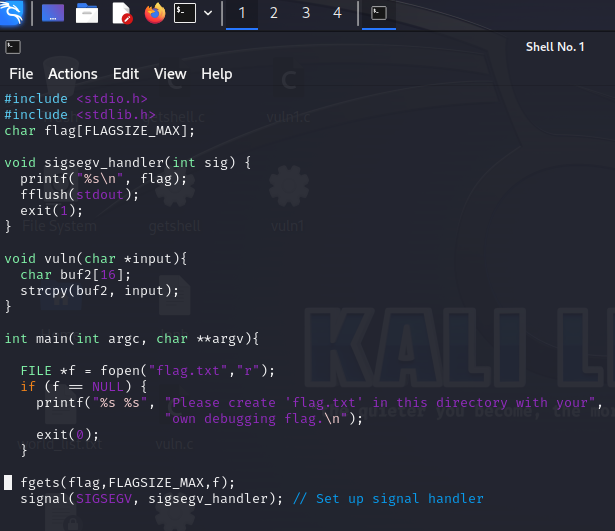
Description automatically generated

+ Input:AAAAAAAAAAAAAAAAA

+ Output: picoCTF{ov3rfl0ws\_ar3nt\_that\_bad\_9f2364bc}

A screenshot of a computer

Description automatically generated

* 1. **Nội dung thực hiện chi tiết:  
     -**

-Ta thấy đoạn code trên có một số điểm yếu tiềm ẩn liên quan đến bảo mật và có thể bị khai thác qua một cuộc tấn công tràn bộ đệm (buffer overflow)

1) Sử dụng hàm gets(): Hàm gets() không kiểm tra kích thước của bộ đệm đầu vào, điều này cho phép người dùng nhập vào một chuỗi dài hơn kích thước của bộ đệm buf1, dẫn đến tràn bộ đệm.

2)Sử dụng hàm strcpy() mà không kiểm tra kích thước: Hàm này sao chép chuỗi từ input vào buf2 mà không kiểm tra kích thước của chuỗi đầu vào so với kích thước của buf2, tiếp tục tạo điều kiện cho tràn bộ đệm.

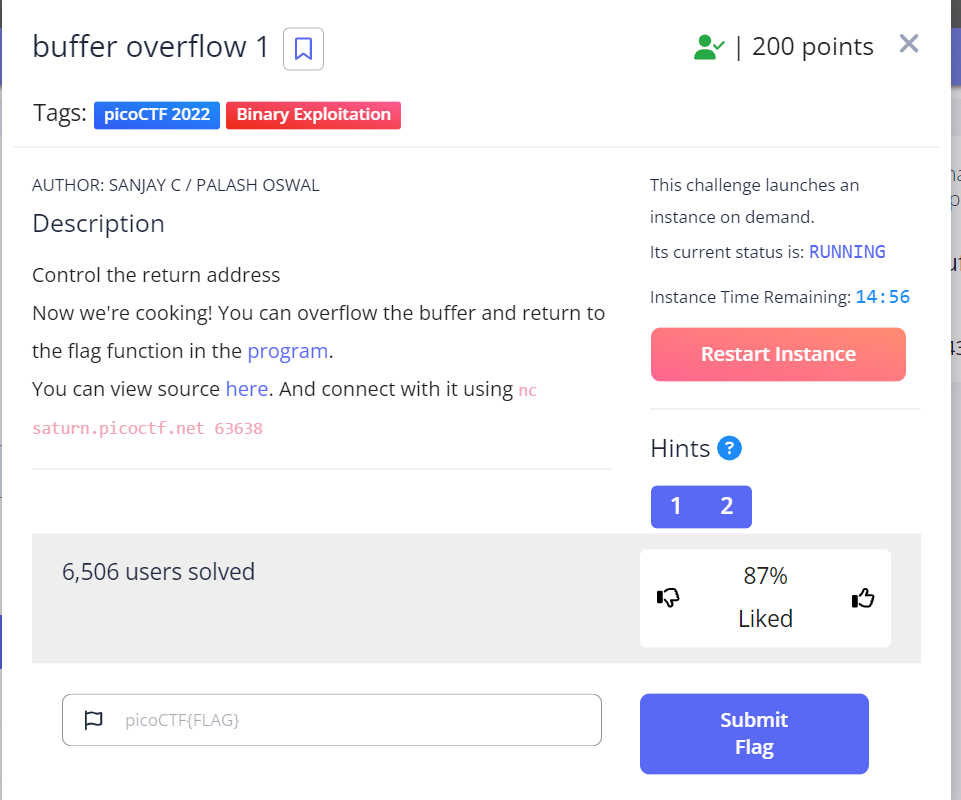
3)Xử lý tín hiệu SIGSEGV: Trình xử lý tín hiệu sigsegv\_handler được đăng ký để xử lý tín hiệu SIGSEGV (Segmentation Fault). Trình xử lý này in ra giá trị của biến flag khi một lỗi phân đoạn xảy ra, thay vì chỉ đơn thuần kết thúc chương trình.

-Ta thấy mảng ký tự buf2 được khai báo với kích thước 16 byte.Vì buf2 chỉ có kích thước 16 byte, việc sao chép 21 byte vào buf2 sẽ ghi đè lên vùng nhớ nằm ngay sau buf2. Trong một số trường hợp, điều này có thể ghi đè lên các giá trị quan trọng khác trên stack, như địa chỉ trả về của hàm hoặc giá trị của các biến khác.

**-Biện pháp phòng ngừa:** Sử dụng các hàm an toàn hơn như fgets() thay cho gets(), và kiểm tra kích thước đầu vào trước khi sao chép chuỗi.

1. **KỊCH BẢN 2**

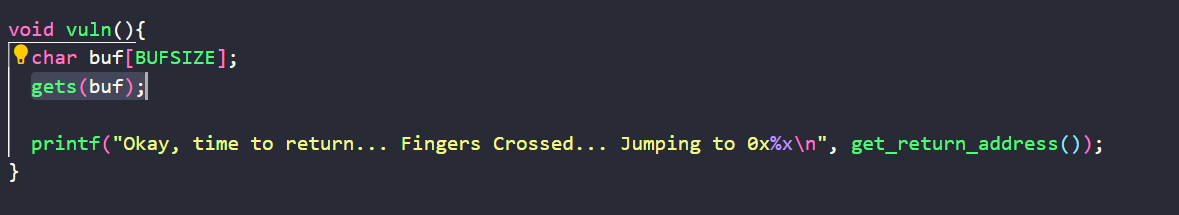
**Buffer Overflow 1**

****

Bước 1: Để có thể lấy được flag để hoành thành challenge ta cần gọi được hàm win() nhưng trong chương trình chỉ khởi tạo hàm nhưng không sử dụng hàm nên ta phải dùng cách khác để gọi hàm này



Kiểm tra source của server ta thấy hàm vuln() sử dụng hàm gets không phải fgets (gets chứa lỗ hổng buffer overflow do không kiểm tra kích

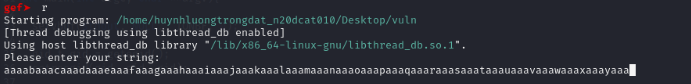
thước của giá trị đi vào hàm.

Bước 2: Đầu tiên ta cần tìm và xác định địa chỉ của esp chưa return address của hàm vuln(). Sử dụng gdb

A computer screen shot of a program

Description automatically generated

Ở bước này sau khi run sẽ yêu cầu bạn nhập một chuỗi ký tự và ở đây tôi dùng lệnh “pattern create 100” để tạo ra chuỗi 100 ký tự sau đó nhập vào vị trí được yêu cầu.



Đây là kết quả hiện ra sau khi chạy được

A screenshot of a computer program

Description automatically generated

Bước 3: Tìm phần bù

A screen shot of a computer program

Description automatically generated

"[ebp-0x28]" đề cập đến một vị trí cụ thể trong bộ nhớ được tính từ giá trị của thanh ghi cơ sở của frame stack (EBP) trừ đi một giá trị cụ thể là 0x28, tương đương với 40 byte trong hệ thập lục phân.

Ta thấy Dword PTR [ebp-0x4] đề cập tới một từ 4 bytes trong bộ nhớ do đó nó sẽ tham chiếu tới một từ ở vị trí cách 4 byte so với vị trí được lưu trữ trong ebp.

Vậy ta cần một từ có độ dài 44 byte là đủ để tràn bộ đệm.

Bước 4: Khai thác hàm gets() để trả về đia chỉ của hàm win()

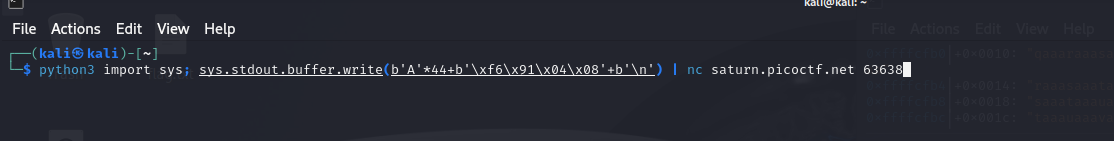
A screen shot of a computer program

Description automatically generated

Đia chỉ breakpoint là 0x080491f6, lúc khai thác hàm gets() chúng ta cần trả về địa chỉ này

Địa chỉ này được định dạng little-endian là: \xf6\x91\x04\x08

Bước 5: Tấn công



Đoạn mã Python này có chức năng gửi một payload đặc biệt tới một địa chỉ IP và cổng cụ thể sử dụng giao thức TCP. Cụ thể:

* "import sys; sys.stdout.buffer.write(b'A'\*44+b'\xf6\x91\x04\x08'+b'\n')": Đây là mã Python được chạy. Nó là một dòng lệnh đơn giản thực hiện các thao tác sau:
* sys.stdout.buffer.write(b'A'\*44+b'\xf6\x91\x04\x08'+b'\n'): Ghi một payload vào standard output. Payload này được tạo ra bằng cách nối chuỗi:
  + b'A'\*44: Một chuỗi gồm 44 ký tự 'A'.
  + b'\xf6\x91\x04\x08': Một địa chỉ trong định dạng little-endian. Đây là địa chỉ của hàm win()
  + b'\n': Ký tự xuống dòng thay cho hành động nhấm phím enter.
* "| nc saturn.picoctf.net 63638": "|Trong trường hợp này, nó sử dụng nc (netcat) để kết nối tới máy chủ có địa chỉ IP saturn.picoctf.net trên cổng 63638.

Kết quả:



A screenshot of a computer

Description automatically generated

* + Đầu tiên ta đọc source của challenge này ta thấy để lấy được flag ta cần làm sao để cho biến code bằng giá trị GOAT để lấy flat

A computer screen with text

Description automatically generated

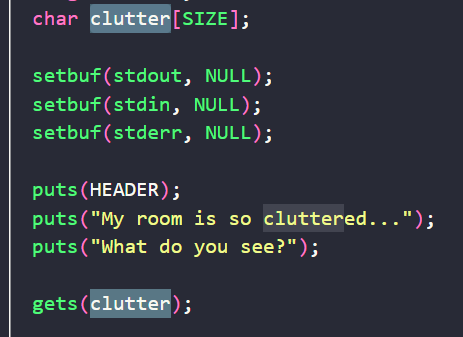
* + Tuy nhiên chương trình lại không cho chúng ta tương tác với biến code và mặc định biến code bằng 0

A computer screen shot of a program code

Description automatically generated

* + Ta nhận thấy chương trình sử dụng hàm gets() để lấy dữ liệu nhập vào và hàm gets() chứa lỗ hổng không kiểm tra kích thước dữ liệu nhập vào so với kích thước được cấp cho biến clutter là 0x100 tức là 256 kí tự.

Ta có thể buffer overflow hàm gets này



* + Khi chạy hàm main các biến cục bộ sẽ được push theo thứ tự vào stack ta sẽ lợi dụng việc này kết hợp với lỗ hổng của hàm gets() để tiến hành ghi đè giá trị từ biến clutter đến code

A black background with white text and colorful letters

Description automatically generated

 Sau khi được push vào stack

A white rectangular sign with black text

Description automatically generated

* + Đầu tiên ta cần xác định địa chỉ của 2 biến trên trong chương trình
  + Sử dụng pwndbg và đặt break point tại địa chỉ hàm gets ta sẽ lấy được địa chỉ của biến clutter

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA computer screen with colorful text

Description automatically generated

* + Tiếp theo ta đặt breakpoint tại chỉ thị cmp

A close-up of a word

Description automatically generatedA computer screen with text

Description automatically generated

* + Ta sẽ thấy chỉ thị cmp sẽ so sánh 2 giá trị tại địa chỉ của rbp-8 và rax
  + Kiểm tra rax ta thấy đây chính là giá trị GOAT



* + Vậy rbp-8 là sẽ là địa chỉ của code



* + Kiểm tra lại 2 địa chỉ clutter và code

A screenshot of a computer

Description automatically generated

 Ta chỉ cần ghi 264 kí tự bất kì cùng với 0xdeadbeef ở little endian ta sẽ thay đổi giá trị của code thành 0xdeadbeef

* + Thử nghiệm vào chương trình

A computer screen shot of a computer screen

Description automatically generated

* + Thử nghiệm vào server challenge để lấy flag

A computer screen shot of a computer screen

Description automatically generated