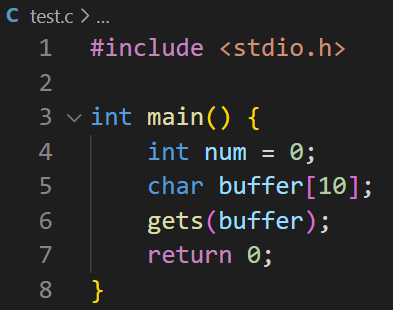
**Buffer Overflow (P1)**

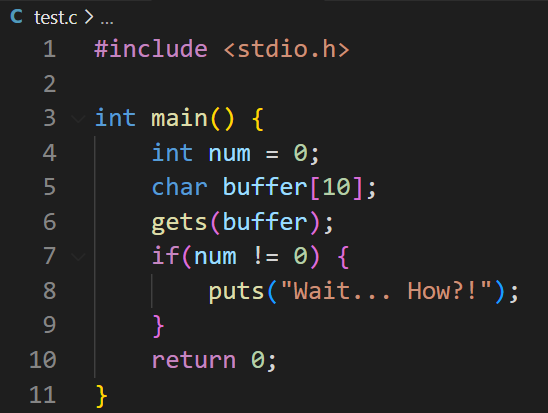
Lỗi Buffer Overflow (tràn bộ đệm) là một trong những lỗi chương trình phổ biến mà các pwner thường khai thác, và cùng là nền tảng cho những kỹ thuật pwn phức tạp như ret2shell, ret2libc, ROP,…

Vậy lỗi tràn bộ đệm có thể bị gây ra như thế nào?



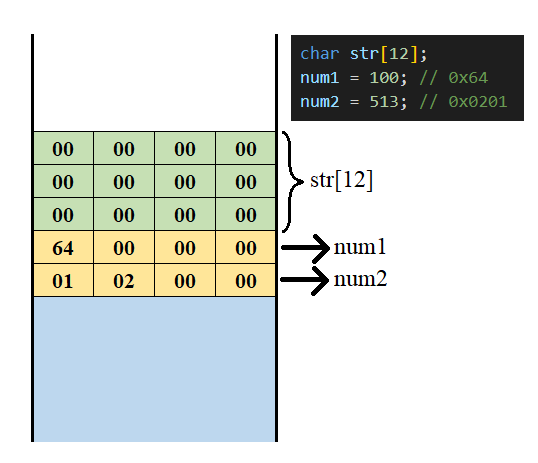
Một đoạn code C trông có vẻ vô hại. Chương trình này khai báo một biến ‘num’ kiểu int có giá trị bằng 0 và một biến ‘buffer’ là dãy 10 ký tự kiểu char. Sau đó chương trình tiến hành lấy chuỗi input từ người dùng nhập vào biến ‘buffer’ và kết thúc chương trình mà không làm gì cả.

Nhưng nếu chương trình thật sự làm gì đó thì sao?



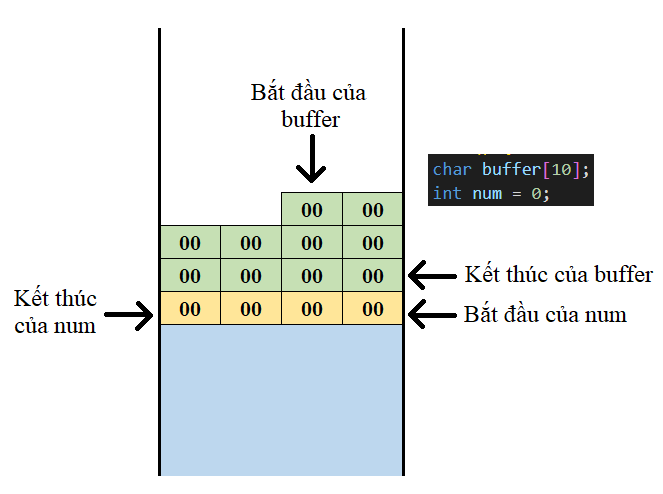
Tại đây ta thấy, sau khi chương trình nhận vào input của người dùng bỏ vào biến ‘buffer’ thì sẽ kiểm tra giá trị của biến ‘num’ xem có khác 0 hay không. Trong suốt quá trình chạy chương trình, ta không hề thực hiện bất kỳ một tác vụ nào làm thay đổi biến ‘num’ nên xét theo lẽ thường, điều kiện if không thể đạt được.

Vấn đề đặt ra ở đây là liệu có cách nào để thay đổi biến ‘num’ khi chạy chương trình trên hay không? Câu trả lời là có.



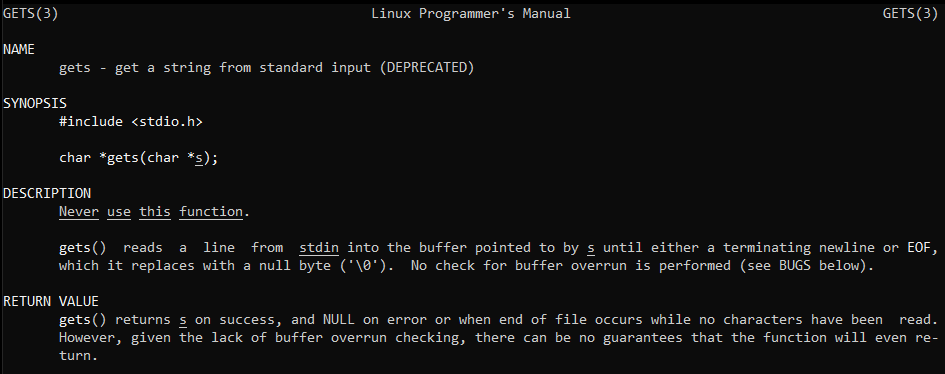
Thông thường, trong kiểu kiến trúc intel x86, các biến local của hàm khi khai báo sẽ được lưu vào trong vùng nhớ stack, và khi ta compile chương trình bằng gcc, các biến có kiểu dữ liệu kích thước cố định sẽ được ưu tiên xếp ở dưới.

Thêm một điều nữa, vì intel x86 là kiểu kiến trúc Little Endian nên ngoại trừ các chuỗi ký tự, khi ta tiến hành đọc hay lưu dữ liệu tại địa chỉ của vùng nhớ bất kỳ thì các byte giá trị sẽ được xếp ngược lại. Tức là nếu địa chỉ chứa một biến có kiểu dữ liệu int (4 byte) đang có giá trị lần lượt là 01 02 03 04 thì giá trị thực của biến đó sẽ là 0x04030201.



Do đó ta có thể thấy ở ảnh trên, khác với địa chỉ bắt đầu của chuỗi ‘buffer’ là nằm ở trên cùng và đọc dần xuống dưới cuối, khi ta đọc biến ‘num’ thì chương trình sẽ tự hiểu và đọc ngược các byte từ cuối về đầu, tuy nhiên địa chỉ bắt đầu của biến ‘num’ vẫn nằm ở vị trí ‘kết thúc của num’ như trên hình, chỉ có việc đọc là được tiến hành ngược lại.

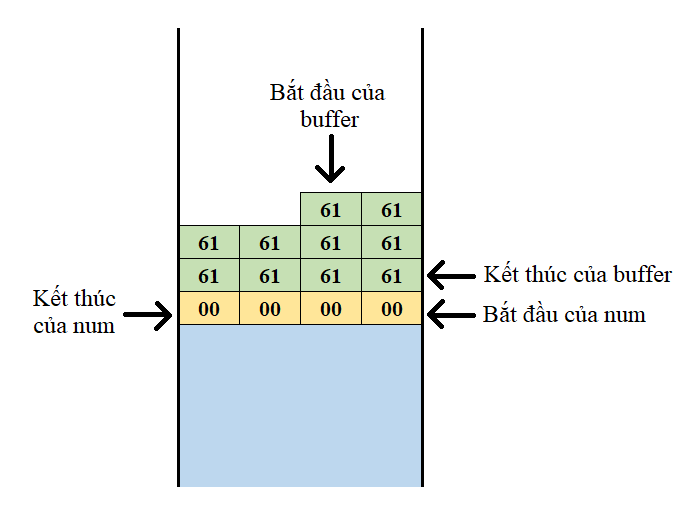
Điều này đòi hỏi các pwner phải nắm một ít kiến thức nền tảng về kiến trúc máy tính đó :’)



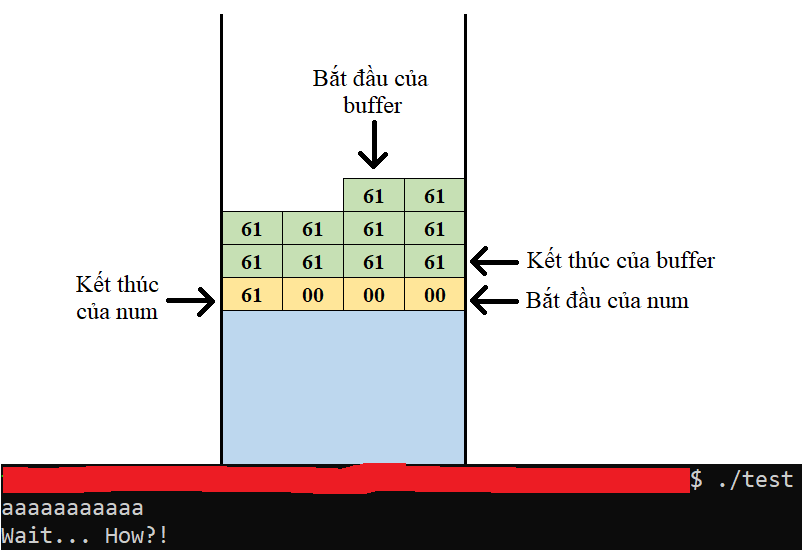
Quay trở lại vấn đề chính, đầu tiên hãy nhìn vào Linux Programmer’s Manual dành cho hàm gets(). Ta biết được rằng hàm gets() đọc một dòng từ stdin vào bộ đệm (buffer) được trỏ tới bởi tham số con trỏ s và dừng lại cho đến khi gặp ký tự xuống dòng (‘\n’) hoặc ký tự EOF (End of File). Không những thế, ta con nhận được một dòng cảnh báo được gạch chân rất rõ ràng: “Đừng bao giờ sử dụng hàm này.”

Tại sao lại như vậy?

Bản chất của hàm gets() là đọc vào chuỗi ký tự từ người dùng hoặc từ file, tuy nhiên nó lại không hề kiểm tra số lượng ký tự mà nó nhận vào. Chính do đó mà cho đến khi chưa gặp ký tự xuống dòng hoặc EOF thì nó vẫn sẽ tiếp tục đọc, và điều này dẫn đến một lỗi vô cùng phổ biến đó chính là tràn bộ đệm.



Giả sử ta nhập vào một chuỗi 10 ký tự ‘a’ (có mã hex 0x61). Hàm gets() đọc vào chuỗi này và tiến hành lưu vào trong vùng nhớ từ bắt đầu của biến ‘buffer’ cho đến kết thúc của biến ‘buffer’, vừa vặn 10 byte. Mọi thứ diễn ra như bình thường, chương trình chạy đúng như những gì chúng ta dự đoán lúc trước và không có điều gì kỳ lạ xảy ra cả.



Nhưng nếu ta nhập vào chuỗi 11 ký tự ‘a’ thì sao?

Hàm gets() vẫn sẽ nhận vào chuỗi 11 ký tự đó, tiến hành lưu vào vùng nhớ từ bắt đầu của biến ‘buffer’ cho đến kết thúc của biến ‘buffer’, nhưng độ dài vùng nhớ của biến này chỉ có 10 byte. Câu hỏi đặt ra là: byte thứ 11 sẽ đi về đâu?

Câu trả lời chính là nó sẽ tràn xuống vùng nhớ ở dưới, không đâu khác chính là vùng nhớ của biến ‘num’. Vậy khi này nếu như chương trình tiến hành đọc dữ liệu của biến ‘num’ thì khi ấy nó sẽ không còn mang giá trị 0 như ban đầu nữa mà lại là 0x61, tức bằng 97. Ta đã có thể thay đổi được giá trị của biến ‘num’ và điều chỉnh được luồng thực thi của chương trình.



Nếu nhập vào nhiều ký tự hơn nữa, ta không những có thể ghi đè lên biến ‘num’ mà còn tiếp tục ghi đè xuống được vùng nhớ ở dưới biến ‘num’ đó. Nhưng vấn đề này sẽ được đè cập tới ở phần tiếp theo.

Một lưu ý nho nhỏ, là đối với các phiên bản gcc mới hơn, compiler sẽ luôn ưu tiên đặt các biến buffer ở vùng địa chỉ cao hơn các biến riêng lẻ khác (tức là nằm ở dưới trong stack) vậy nên việc ghi đè lên biến ‘num’ như ví dụ trên sẽ không còn khả thi nữa.

Trên thực tế, không chỉ mỗi hàm gets() mà còn có nhiều những hàm khác có thể gây nên lỗi tràn bộ đệm. Và việc tràn bộ đệm này có thể giúp các attacker thay đổi được luồng thực thi của chương trình, và hơn nữa là chiếm được shell của hệ thống.