RETURN TO LIBC ATTACK LAB

Nguyễn Hoàng Dũng - Nguyễn Hồ Thăng Long - Nguyễn Quốc Trung

1 GIỚI THIỆU

Đây là một kĩ thuật khai thác lỗ hổng bằng lỗi tràn bộ đệm (Stack-based Overflow)

Mục đích là chuyển hướng thực thi đến các hàm nằm trong thư viện libc

Sử dụng các hàm trong libc để thực hiện các câu lệnh shell (VD: /bin/sh)

Có thể leo thang đặc quyền nếu chương trình lỗi được thực thi dưới quyền root

2 SETUP MÔI TRƯỜNG

Tắt cơ chế Address Space Layout Randomization

Chương trình 32 bit chứa lỗi và có SetUID quyền root

Giải Thích:

Nếu bật ASLR, mỗi lần chương trình được chạy sẽ có địa chỉ các hàm libc được random. Trong khuôn khổ của bài lab thì sẽ chưa có cách exploit để bypass được cơ chế này.

Chương trình lỗi chưa quyền root sẽ mô phỏng được trường hợp leo thang đặc quyền: chỉ cần shell được spawn bởi bởi chương trình lỗi thì dù không cần root password vẫn có thể có được root shell.

3 MÔ TẢ CHƯƠNG TRÌNH

Đọc input 1000 kí tự từ badfile và gọi hàm strcpy để copy 1000 kí tự của input vào 1 buffer 12 bytes

Có cung cấp sẵn 1 số thông tin tại thời điểm được thực thi như: địa chỉ của input, địa chỉ của buffer và địa chỉ frame pointer của hàm bof (nơi hàm strcpy được gọi)

Nhận Xét:

Copy 1000 bytes input vào 12 bytes buffer bằng hàm strcpy sẽ gây ra lỗi buffer overflow

Do chương trình nhận input bằng cách đọc 1000 bytes từ badfile nên input có thể chứa các kí tự đặc biệt như null hoặc xuống dòng.

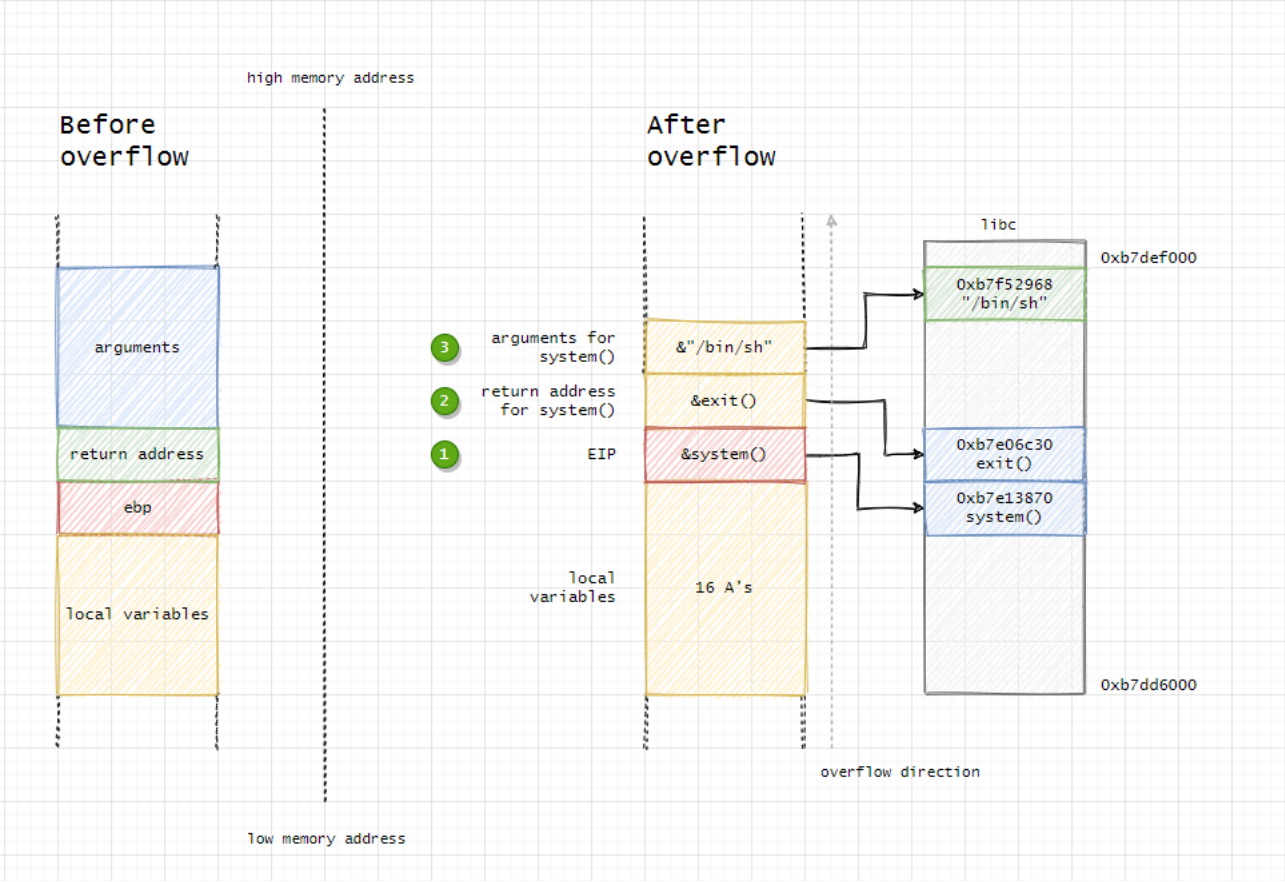
Có thể xác định địa chỉ các dữ liệu nằm trong input và xem như con trỏ đến một biến, do input cũng được đẩy vào stack và đã tắt cơ chế ASLR nên các địa chỉ này cũng sẽ không thay đổi.

Địa chỉ bắt đầu stack của các chương trình khi chạy thực tế sẽ khác so với khi debug

4 TÌM ĐỊA CHỈ CÁC HÀM LIBC

Nhóm đã mô tả lại quá trình tắt ASLR và tìm địa chỉ các hàm cần thiết trong demo

5 PHƯƠNG PHÁP CHUNG



Overview of ret-to-libc technique for 32 bit vulnerable program (source: ired.team)

6 ATTACK LAB 1

Đây là kĩ thuật exploit bằng cách return về hàm system của libc

Hàm system sẽ nhận 1 tham số là địa chỉ trỏ đến string là câu lệnh cần thực thi

Nhóm em đã mô tả lại quá trình setup tham số để lưu vào biến env, cách tìm địa chỉ của biến này và những điều cần lưu ý trong demo

Nhận Xét:

Không được thay đổi tên chương trình thực thi, do sẽ ảnh hưởng đến địa chỉ biến được lưu trong env

Shell được spawn chỉ có quyền user, không có quyền root. Điều này xảy ra do hàm system khi được thực thi sẽ gọi shell mặc định (là /bin/dash) để thực hiện câu lệnh đã được lưu tại biến env. Mặc định khi được gọi, /bin/dash sẽ drop quyền root, nếu không có thêm tham số là -p. Do đó tại thời điểm khi câu lệnh /bin/sh được thực thi, quyền root đã không còn.

Trên thực tế, việc tồn tại một biến nằm trong env, phù hợp để attacker có thể exploit và biết được luôn địa chỉ của biến này nằm trong stack khi một chương trình được thực thi là hy hữu, hoặc nếu có sẽ phụ thuộc vào ngữ cảnh của cuộc tấn công (có thể do thông tin đã được tìm thấy ở một lỗi exploit khác)

7 ATTACK LAB 2

Đây là kĩ thuật exploit bằng cách return về hàm execv của libc

Hàm execv sẽ nhận 2 tham số: địa chỉ đến câu lệnh cần thực thi và mảng chứa các con trỏ đến tham số của câu lệnh

```
int execv(const char *pathname, char *const argv[]);
```

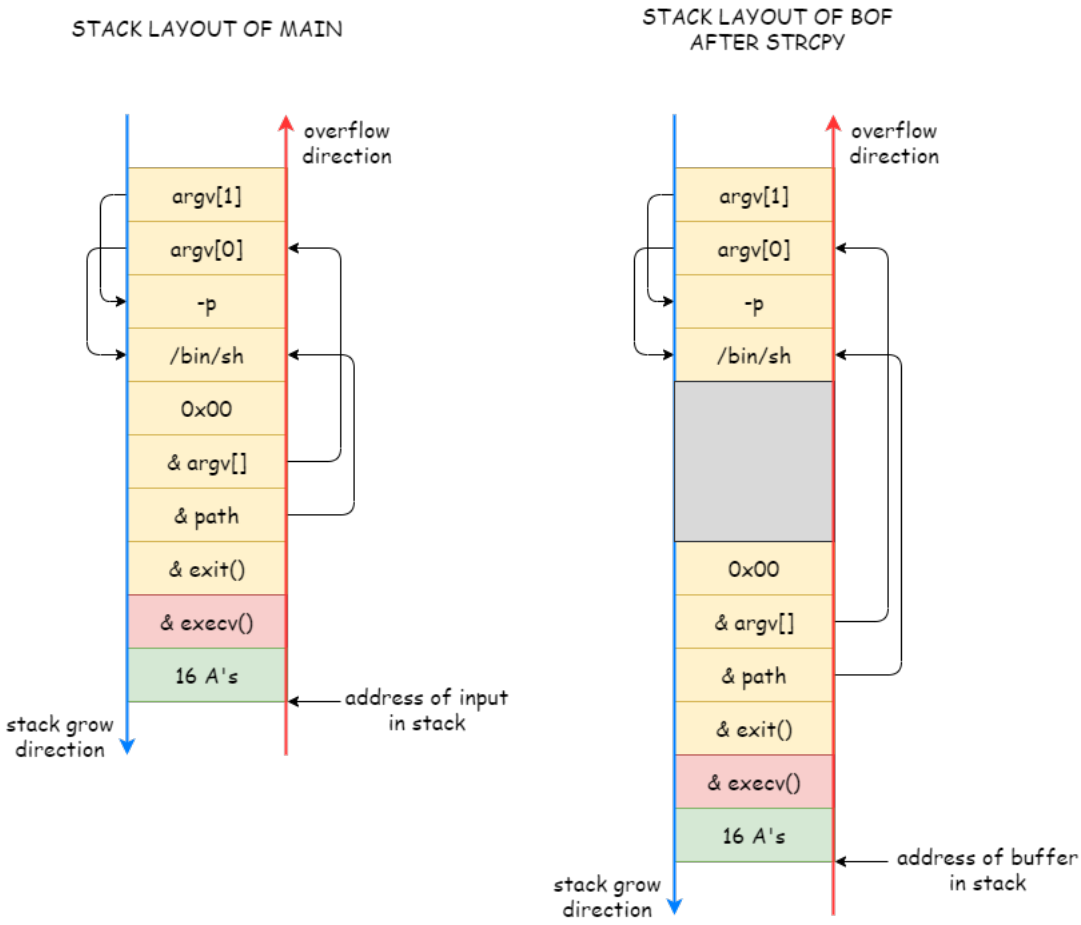
Phân Tích:

Cấu trúc của argv[] yêu cầu mỗi phần tử là con trỏ đến địa chỉ chứa chuỗi là tham số. Các chuỗi này sẽ phải kết thúc bằng NULL nên không thể export env như lab 1

Ta sẽ truyền các tham số (/bin/sh, -p) vào input để khi chương trình thực thi, các tham số này sẽ được ghi tiếp vào stack

Vì chương trình có cung cấp địa chỉ của input trong stack khi được thực thi và địa chỉ này cố định do đã tắt ASLR, nên địa chỉ của các tham số (trước đó đã được truyền vào input) cũng sẽ cố định. Từ địa chỉ bắt đầu của input, ta có thể tính được địa chỉ các tham số nằm trong input

Cấu Trúc Payload:



Nhận Xét:

Khi hàm strcpy copy payload từ input vào buffer, payload đã bị đứt đoạn (tại 0x00). Tuy nhiên địa chỉ của /bin/sh và -p thì vẫn không đổi, và chỗ đứt đoạn sẽ không làm ảnh hưởng đến việc overflow để return to execv

Lần này do câu lệnh /bin/sh được execv thực hiện trực tiếp kèm theo cờ -p nên quyền root không bị drop và ta nhận được shell root

Tuy nhiên payload lại chứa nhiều kí tự NULL và trên thực tế thì ta mong muốn hạn chế tối đa các kí tự đặc biệt như vậy

8 ATTACK LAB 3

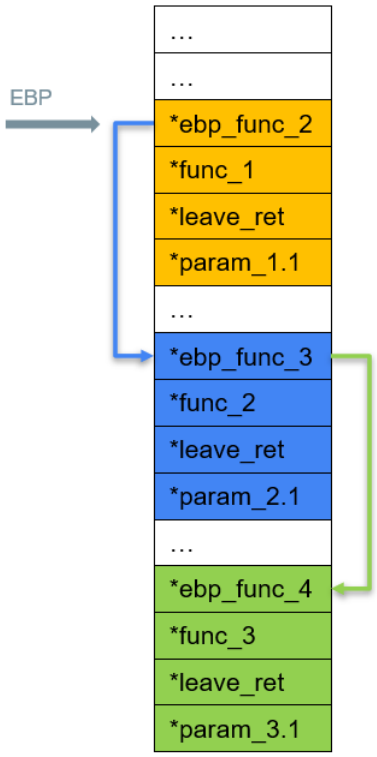
Đây là kĩ thuật exploit bằng cách chain nhiều hàm libc với nhau để đạt được mục đích

Ta mong muốn chain 2 hàm setuid và system với nhau để có thể có tham số /bin/sh là đơn giản nhất và hàm setuid sẽ ngăn chặn việc drop quyền root mặc định của /bin/dash và ta có được root shell

Tuy nhiên, tham số cho hàm setuid phải là 1 con số 4 byte có giá trị = 0, nên ta sẽ cần chain thêm hàm sprintf trước đó để copy byte 0x00 lần lượt 4 lần vào đúng vị trí chứa tham số của hàm setuid trong stack

Địa chỉ quay về sau khi 1 hàm được thực hiện sẽ là địa chỉ instruction leave.

Stack Layout:



Nhận Xét:

Do chương trình có cung cấp giá trị của thanh ghi EBP khi thực thi, và giá trị này là cố định do đã tắt ASLR nên ta có thể tính chính xác được EBP của hàm tiếp theo.

Khi đó lệnh leave sẽ giúp ta clear và setup stack layout về đúng vị trí cho hàm được chain tiếp theo.

Nếu không có thông tin về EBP, thì ta sẽ phải tự tính xem cần pop các giá trị ra khỏi stack bao nhiêu lần để đến được stack của hàm tiếp theo, và phải tìm địa chỉ của một lệnh pop nào đó có sẵn trong chương trình.