Yêu cầu 1. Giả sử cần chuẩn bị chuỗi định dạng cho printf(). Sinh viên tìm hiểu và hoàn thành các chuỗi định dạng cần sử dụng để thực hiện các yêu cầu bên dưới.

|  |  |
| --- | --- |
| **Yêu cầu** | **Chuỗi định dạng** |
| 1. In ra 1 số nguyên hệ thập phân | %d |
| 1. In ra 1 số nguyên 4 byte hệ thập lục phân, trong đó luôn in đủ 8 số hexan. | %08x |
| 1. 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. | %05d |
| 1. In tối đa chuỗi 8 ký tự, nếu dư sẽ cắt bớt. | %.8s |
| 1. In ra 1 số thực, trong đó đầu ra sẽ chiếm ít nhất 7 ký tự và luôn hiển thị 3 chữ số thập phân. Nếu số chữ số không đủ, nó sẽ đệm khoảng trắng ở phần nguyên. |  |
| 1. In ra 1 số thực, trong đó đầu ra sẽ chiếm ít nhất 7 ký tự và luôn hiển thị 3 chữ số thập phân. Nếu số chữ số không đủ, nó sẽ đệm ký tự 0 ở phần nguyên. | %7f |

B.2.1 Đọc dữ liệu trong ngăn xếp – stack

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Source code file app-leak.c

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Khi built thì trình có báo lỗi ở dòng 7 printf(s) sẽ xảy ra lỗi format string nhưng ta vẫn build được

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Chạy chương trình và nhập một chuỗi bình thường

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Khi ta nhập một chuỗi định dạng thay vì in lại chuỗi ta đã nhập thì chương trình lại in ra một giá trị nào đó
* Trong đó:

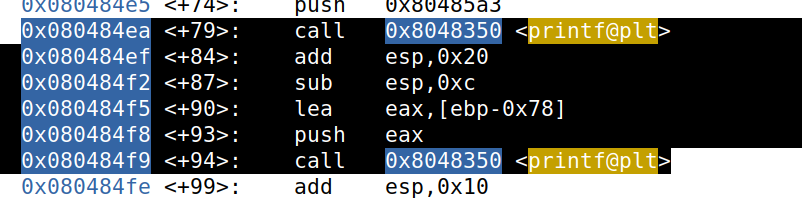
+ 8 là in ra 8 ký tự

+ 0 là nó sẽ thêm 0 vào thay vì khoảng trống nếu không đủ 8 ký tự

+ x là ta sẽ in ở lower0case hexadecimal



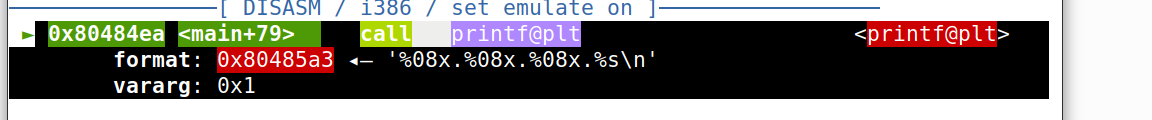
* Dùng gdb để phân tích chương trình

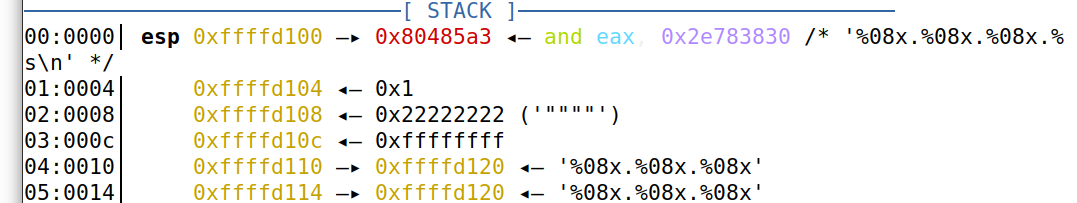


* Ta thấy hàm printf nằm ở 0x080484eavà 0x080484f9

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Đặt break point và run thì chương trình sẽ dùng ở hàm scanf để người dùng nhập
* Ta nhập chuỗi định dạng %08x.%08x.%08x
* 



* Ta thấy tham số đầu tiên của hàm printf là địa chỉ của chuỗi định dạng %08x.%08x.%08x.%s\n
* Tham số thứ 2 là giá trị của a
* Tham số thứ 3 là giá trị của b
* Tham số thứ 4 là giá trị của c
* Tham số thứ 5 là địa chủ tương ứng của chuỗi ta đã nhập

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta tiếp tục chạy tới lệnh printf thứ 2 bằng lệnh c

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Tại printf thứ 2 chỉ ó 1 tham số là chuỗi s đã nhập. Tuy nhiên, do ở đây ta cố tình nhập chuỗi s giống như chuỗi định dạng. Printf() sẽ coi đó là chuỗi định dạng và đi tìm các giá trị cụ thể để in ra tương ứng
* Thông thường, chuỗi định dạng sẽ là tham số thứ nhất của printf(Dòng esp đầu tiên), các giá trị cần in sẽ ở vị trí tham số thứ 2 và thứ 3(2 dòng esp tiếp theo)
* Với stack trên, đối với printf() thứ 2, các tham số của nó được lưu từ địa chỉ 0xffffd110. Tham số đầu tiên là địa chỉ chuỗi s(0xffffd120)
* Vậy theo logic hàm printf() sẽ hiểu các giá trị nằm sau đó ở các địa chỉ 0xffffd114, 0xffffd118, 0xffffd11c lần lượt là các tham số 2, 3, 4 và là những giá trị cần in
* Với định dạng %x, chương trình sẽ coi các giá trị tại địa chỉ chỉ này như kiểu int và in chúng ra. Do đó tiếp tục chạy ta được kết quả là các giá trị tại các ô nhớ nằm ngay phía sau tham số đầu tin của printf

Ảnh có chứa văn bản

Mô tả được tạo tự động

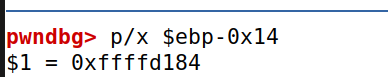
* Các kết quả sẽ không giống nhau mọi lúc vì dữ liệu trên ngăn xếp sẽ khác nhau do các stack được cấp phát mỗi lần

Yêu cầu 2. Sinh viên khai thác và truyền chuỗi s để đọc giá trị biến c của main. Giải thích ý nghĩa của chuỗi định dạng và lý do có thể in được giá trị cần thiết. Bonus: chuỗi s không dài hơn 10 ký tự

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta dùng gdb thấy biến c nằm ở vị trí ebp-0x14



* Địa chỉ cụ thể là 0xffffd184

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta xem vị trí đặt tham số của hàm printf thứ 2. Ta thấy tham số đầu tiên, là địa chỉ chuỗi định dạng được đặt ở vị trí 0xffffd110

Ảnh có chứa bàn

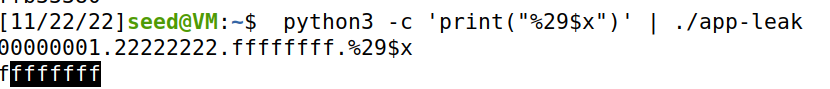
Mô tả được tạo tự động

* Ta xem các giá trị lưu gần địa chỉ 0xffffd110 thấy vùng màu đen là giá trị của các biến a, b, c
* 0xffffd120 là tham số thứ nhất của printf, vốn cần 1 chuỗi định dạng, các khối dữ liệu phía sau nó sẽ lần lượt được đọc theo các ký hiệu. Ví dụ, 1 ký thệu %x sẽ đọc 4 byte dữ liệu từ các ô nhớ phía sau
* Như vậy cần 29 ký hiệu %x để có thể đọc được đến dữ liệu tại phần biến a, b, c

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta đọc được giá trị 3 biến a, b, c



* Giá trị k và m đều như nhau ở 2 lệnh
* Nếu k=29 thì nó sẽ in hết các giá trị trong stack tới giá trị ta cần in còn m=29 thì nó sẽ chỉ in giá trị tại vị trí ta chỉ định

B.2.2 Đọc chuỗi trong ngăn xếp

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Chạy chương trình và nhập chuỗi “%s%s%s%s” ta thấy chương trình bị segmentation fault

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dùng gdb để debug và đặt break point tại printf thứ 2

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Ta thấy với lệnh printf thứ 2 thì 3 giá trị tạo 0xffffd0f4, 0xffffd0f8, 0xffffd0fc lần lượt là tham số dành cho 3 format %s, được mong đợi là địa chỉ chứa chuỗi cần in

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

* %s thì các đối số được sử dụng như một con trỏ trỏ đến một chuỗi mà chuỗi này thay thế cho đầu ra

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Khi ta thử dùng %p để in địa chỉ thì ta thấy tại %p thứ 4 in ra một địa chỉ không hợp lệ có trong stack nên sẽ gây crash chương trình làm chương trình bị lỗi

B.2.3 Đọc dữ liệu từ địa chỉ tùy ý

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Đặt break potin tại hàm scanf và run
* Khi chạy đến hàm scanf để đọc chuỗi từ người dùng có thể thấy tham số thứ 2 là 0xffffd990 là địa chỉ lưu của s

Ảnh có chứa bàn

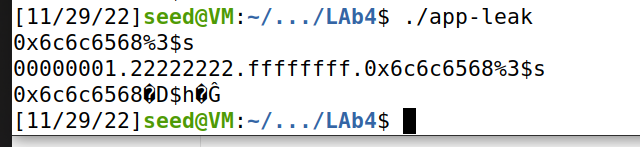
Mô tả được tạo tự động

* Ta xem vị trí đặt các tham số của hàm printf thứ 2
* Ta thấy tham số đầu tiên được đặt ở địa chỉ 0xffffd0f0

Ảnh có chứa bàn

Mô tả được tạo tự động

* Xem các giá trị đang được lưu gần địa chỉ 0xffffd0f0
* Giả sử địa chỉ cần đọc dữ liệu ở đầu chuỗi s(Phần tô đen) địa chỉ cần đọc sẽ nằm ở tham số thứ 4



Yêu cầu 4. Sinh viên khai thác và truyền chuỗi s đọc thông tin từ Global Offset Table (GOT) và lấy về địa chỉ của hàm scanf. Giải thích ý nghĩa của chuỗi định dạng và lý do có thể in được giá trị cần thiết.

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Dòng 16 sẽ loại bỏ địa chỉ của \_\_isoc99\_scanf@got
* Sử dụng %n có thể xem được nội dung của got trong hàm libc

Ảnh có chứa văn bản

Mô tả được tạo tự động

* Kết quả sau khi exploit ta lấy được địa chỉ hàm scanf