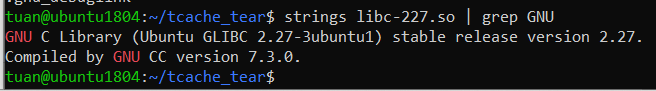
TCACHE\_TEAR

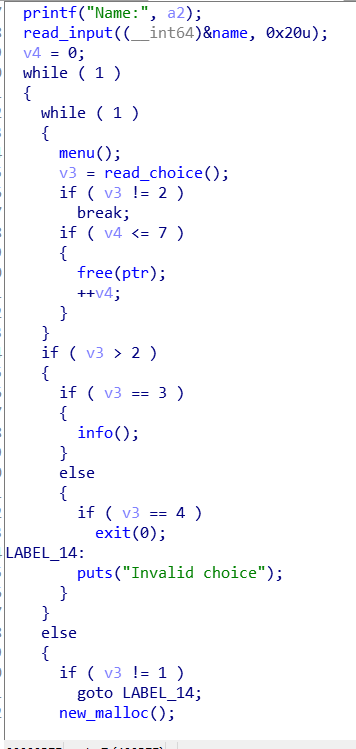
1. Analyze

glibc version:



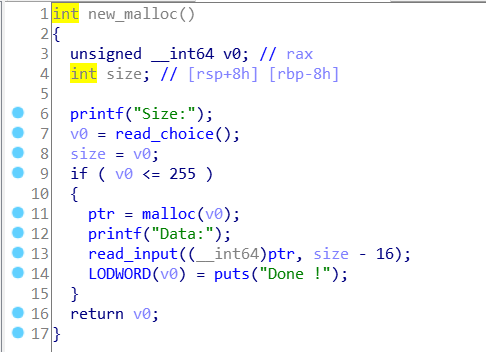
* Glibc phiên bản 2.27

Reversing code:



Flow chương trình khá đơn giản, gồm 3 chức năng: malloc,free,info

1. Malloc :



ở hàm này ta thấy chương trình có lỗi overflow, hàm cho ta nhập size, malloc và sau đó cho nhập size-16 bytes. Nếu size<16 thì chúng ta sẽ nhập được 1 cực kì nhiều.

1. Free

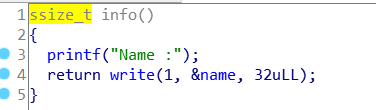
Đối với hàm free, ta thấy chương trình chỉ free(ptr) nhưng lại không gán ptr=null, tức là ta vẫn còn con trỏ của malloc chunk

* Double freed

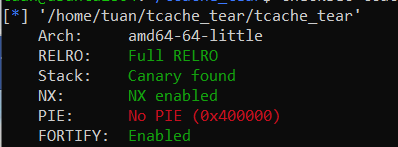
Tuy nhiên ta cũng bị hạn chế chỉ được free tối đa 8 lần.

1. Info:

Hàm này chỉ in ra info ở name:



Checksec:

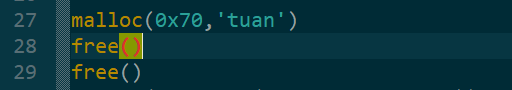
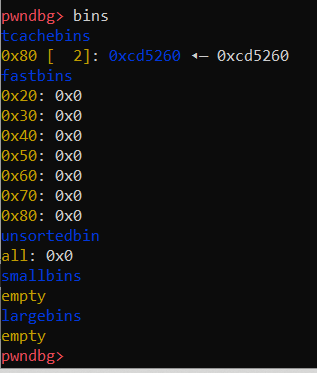


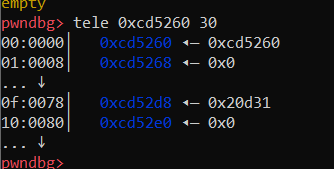
Chỉ có no pie. Tức là ta có địa chỉ name và ptr là cố định.

1. Tìm hướng exploit:

* Đối với bài này, ta thấy không thể thực thi shellcode, cũng không có gì giúp ta call shellcode có sẵn. Nên ta chỉ có thể leak libc và call system.
* Vậy đầu tiên ta phải tìm hướng leak libc, ta có double freed, nếu ta freed 1 unsorted chunk thì có khả năng leak được địa chỉ main\_arena+42, nhưng glibc 2.27 sẽ có tcache, sau khi freed ,chunk sẽ được push vào tcache\_bin. Rất may là tcache\_bin chỉ có thể chứa những chunk có size < 0x408 .

=> Vậy giải sử ta free 1 chunk có size 0x500 thì nó sẽ được push vào unsorted bin.

* Nhưng ta không thể malloc 1 chunk 0x500 được, vì chương trình hạn chế < 0xff. Vậy ta có thể lợi dụng double freed để tạo 1 fake chunk có size là 0x500 ở địa chỉ bss ( name) .
* Step 1: tạo 1 double freed chunk
  + 
  + 
* Step 2: nếu tiếp theo ta malloc 1 chunk có size 0x70 thì sẽ được trả về con trỏ tới 0xcd5260 và được nhập vào 0x70-16 bytes



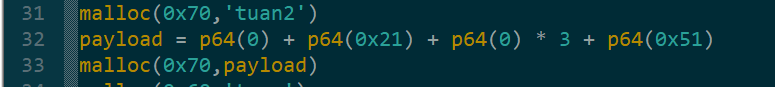
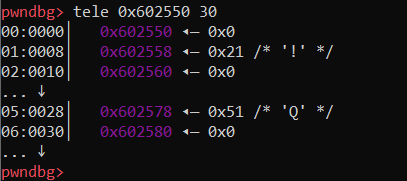
Ta sẽ nhập vào địa chỉ của bss.

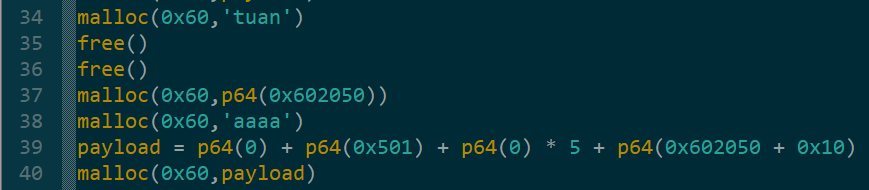
như này.

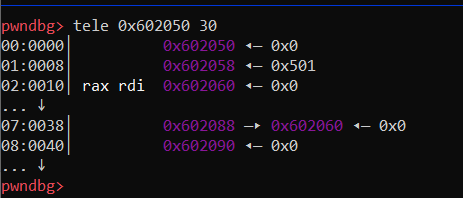
Như vậy,nếu tiếp theo ta malloc 2 lần, thì lần thứ 2 ta sẽ được trả về con trỏ tại bss.

ở đây ta nhập 0x602550 mà không phải là 0x602050 vì ta muốn tạo 1 fake chunk có size là 0x500. Nó sẽ trông như thế này:

|  |  |
| --- | --- |
| 0x602050 |  |
| 0x602060 | Size = 0x501( prev in use) |
|  |  |
|  |  |
|  |  |
|  |  |
| 0x602550 | 0 |
| 0x602558 | Size = 0x21 ( prev in use) (chunk X này để khi free thì không bị nhập vào top chunk) |
|  | 0 |
|  | 0 |
|  | 0 |
| 0x602550+0x20 | Size = 0xx1 ( bypass security check) (fake top chunk) |

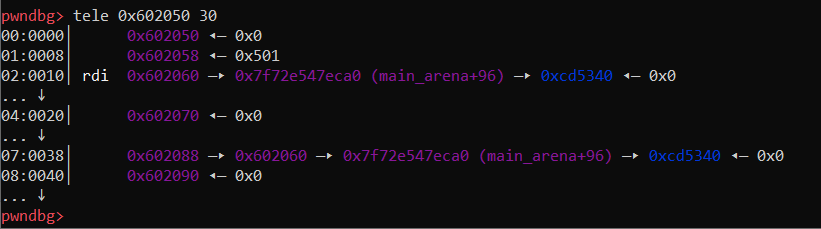
* Step 3: set up fake top chunk và chunk X
  + 
  + Sau khi set up:
  + 
* Step 4: set up fake chunk size 0x500, chúng ta tiếp tục tạo 1 double freed như lần trước.



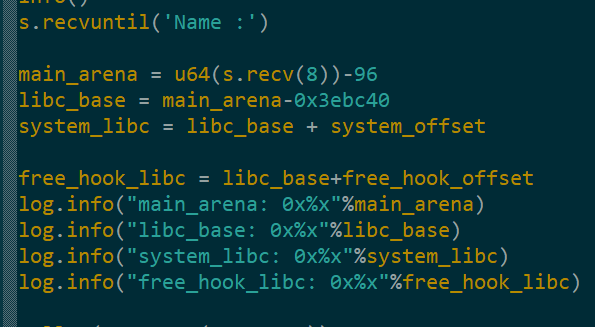
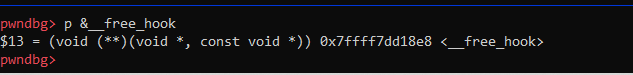
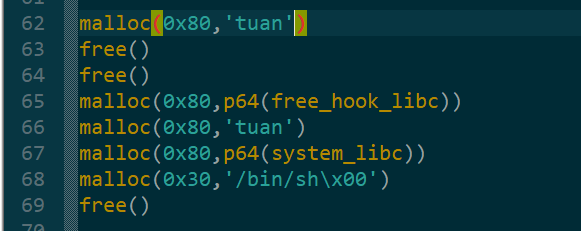
* + 
  + Chúng ta muốn free(0x602060) thì phải ghi địa chỉ đó vào &ptr = 0x602088

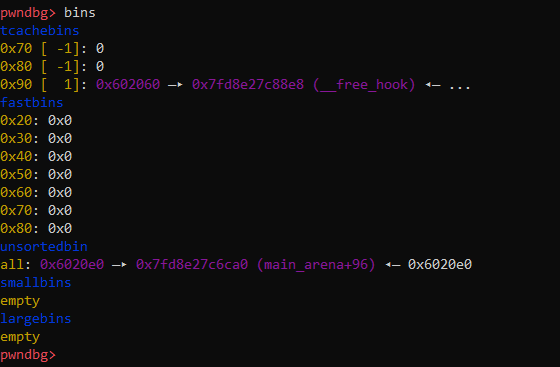


* Step 5: set up xong rồi thì free nó nào.



Yeah, chúng ta đã có được địa chỉ main\_arena+96 và &name, nên chúng ta có thể dùng hàm info để in ra.

* Step 6: leak libc(đơn giản mà)
* Step 7: tính địa chỉ libc, system, \_\_free\_hook
  + 
  + 
* Step 8: ghi đè địa chỉ system vào \_\_free\_hook caller.
  + 
  + Ta cũng tiếp tục tạo 1 case double freed, sau khi malloc 1 lần và ghi địa chỉ free\_hook vào thì

 sẽ như này, ta free thêm 1 lần, và lần cuối cùng thì ta sẽ được malloc trả về địa chỉ của free\_hook. Lúc đó thì thoải mái ghi vào địa chỉ system.

* Step 9:
  + Malloc 1 chunk mới có chứa chuỗi /bin/sh vào free chunk đó.