

Hackergame 2020 Writeups-ranwen

0x00 签到

改HTML step改成1, 提交获取flag

0x01 猫咪问答++

第一个暴力尝试

第二个查RFC1149

第三个显然是Teeworlds (多年老玩家本人)

第四个随便找个带街景的地图数一下

第五个, 官方的存档站暂时502了, 但是archive.is有, 直接找即可

0x02 2048

点开所有js, 肉眼审查一下, 显然获胜判断在html_actuator.js, 然后找到 `/getflag?my_favorite_fruit=banana` 的api得到flag

0x03 一闪而过的 Flag

用WSL运行, 就不会退出了

0x04 从零开始的记账工具人

写了好久的py自动解析大写数字 (悲) (附件1)

0x05 超简单的世界模拟器

第一问随手在Wikipedia上抄一个能平移的就好了

第二问试了好半天, 最后没办法, 写了个模拟器fuzzing了一下 (附件2)

0x06 从零开始的火星文生活

先拉出来了第一句话在百度搜索。在房天下找到了一个网页, 百度爬的是乱码, 点进去是正确的。

比对了一下, 正好是乱码的两个字对应一个字。

拿出来连续两个字, 搜到了 <https://www.v2ex.com/t/421212>, 直接运行命令得到flag

0x07 自复读的复读机

google搜索 `reverse quine python`, 稍作修改就能过第一问 (去掉回车符)

```
x='x=%r;print((x%x%x)[::-1],end="");print((x%x)[::-1],end="")'
```

第二问也是稍作修改

```
x='x=%r;print(__import__("hashlib").sha256((x%x).encode()).hexdigest(),end="")'\n;print(__import__("hashlib").sha256((x%x).encode()).hexdigest(),end="")'
```

得到flag

0x08 233 同学的字符串工具

第一问是毒瘤Unicode问题 字符 `fl(chr(64258))` 的upper是 `FL`，然后交一个 `flag` 就可以了

第二问手动学习一下UTF-7编码。手动用UTF-16构造一个Base64结果。建议只编码一个字符，注意大小端序问题

交 `+AGY-lag` 即可

0x09 233 同学的 Docker

下载Docker，查看Dockerfile，显然是利用overlayfs的一些特性

直接跑到 `/var/lib/docker` 中 `find flag.txt` 即可

0x0A 从零开始的 HTTP 链接

先挂个知乎提问 <https://www.zhihu.com/question/428175951> 各位猜猜关注的人里头有多少是组委会的

其实很简单，先找个支持0端口的开http的东西。本人用的是python的socket。

本来直接发了个raw http包，但是发现真是个web，有点麻烦，于是改成了一个反代。

注意，由于有些NAT设备的要求，非公网ip设备没法连0端口。因此我找了个vps做反代。（附件3）

0x0B 来自一教的图片

随便找个 图片频域信息提取工具，扔进去得到flag

0x0C 超简陋的 OpenGL 小程序

最开始想着改这两个文件，不过发现好像不太奏效。

大力出奇迹：下载一个OpenGL调试程序（本人下了个RenderDoc），之后随便截个帧。能点的按钮都xjb点一下：flag出现了！

0x0D 生活在博弈树上

打开IDA 用了gets 查看变量在内存的分布，第一问显然是多输入一点(大概150个)东西，溢出到success变量后将success搞成true，然后得到flag

第二问似乎是个构造rop链的pwn题，不过本人完全不会bin题，也没空学（全场bin题仅有150分）

0x0E 来自未来的信笺

显然暗示的是GitHub北极备份计划

照着 https://github.com/github/archive-program/blob/master/GUIDE_zh.md 做，将每个二维码搞成一个bin后连接，就可以得到一个.xz文件，解压repo得到flag

由于市面上大部分解二维码的软件对bin的支持很差，所以本题浪费了很长时间。建议直接搜索 `zbar bin` 以节省时间。

0x0F 狗狗银行

测试发现，所有利息计算居然是四舍五入

于是构造一下钱：每张信用卡借2050元，每张储蓄卡存167元。稍微计算一下发现这样显然是赚的。

多开几张卡，然后每隔几天倒腾一下避免信用卡利息上升，大量发包即可刷钱得到flag。

0x10 超基础的数理模拟器

显然是解析积分然后计算

尝试全本地，但是cookies处理好像有点问题，索性全用浏览器

可以用 `header editor` 浏览器插件屏蔽LaTeX渲染，然后只用解析源码就可

Chromium的Console提供了一个叫"活动表达式"的东西，可以每次页面载入完后运行JS，利用这个每次刷新提交

写了个py处理latex，之后就是计算积分了

本人直接大力使用了 <https://zh.numberempire.com/definiteintegralcalculator.php> 来做，就是算了几十个后要换下IP，很痛苦。

之后在本地用python开了个webserver，积分页面请求这个webserver，python再请求网站计算结果后返回。

(附件4)

0x11 永不溢出的计算器

好玩的数论题

看到求根号，考虑服务器是怎么计算的。

第一个想到Cipolla算法。复习了一下，发现根本没法直接把质数 p 换成 $N = pq$ ，Cipolla会错。

因此考虑另外一种做法：对两个质数作为模数求二次剩余，之后用CRT合并。

关于一个质数取模的二次剩余有两个，那么对于 $N = pq$ 就会有四个。

假如输入的 L 关于 p, q 的一个二次剩余分别是 X, Y ，那么服务器可能会得到以下四种

- X, Y
- $(p - X), Y$
- $X, (q - Y)$
- $(p - X), (q - Y)$

再考虑CRT过程，一个算法是 $A \equiv X + p(\text{inv}(p, q)(Y - X)) \pmod{N}$ ，其中 $\text{inv}(a, b) \equiv a^{-1} \pmod{b}$ 。

如果还能找到一个根 $B \equiv X + p(\text{inv}(p, q)((q - Y) - X)) \pmod{N}$ ，那么将两式相减，就可以得到含有因子 p 的一个数字。显然这对于 p, q 互换也成立，因此对于上述的4种情况，有2种都可以得到含有一个质因数的数

然后直接和 N 求gcd即可得到一个质数，然后就可以算出另外一个质数，之后就可以得到RSA私钥 d ，直接算出flag

具体操作流程是：先输入一个 2^{1024} 左右的数与0相加，然后观察一下是否相等。如果大了，就说明对 N 取模了。利用这个方法可以直接得到 N 的具体值。

之后查询sqrt(114514) (我试的较大数字中，只有这个数字是有根的) 得到一个根 X

之后枚举 K 计算 $(X + K)^2$ ，让服务器计算其二次剩余，看是否与 $X + K$ 相等。

若相等，则再挑 K ，否则直接执行上述算法，若能跑出来则已经找到了 p, q ，得到flag

0x12 超精巧的数字论证器

注意到 $\sim x$ 是很重要的单目运算，可以让 x 减1。因此可以大力区间dp只处理双目运算，然后特判这种单目运算

本地先打表，之后直接提交即可 (附件5)

0x13 超自动的开箱模拟器

先来个算法链接 <https://www.zhihu.com/question/27050108/answer/35041211>

之后只要写出这个bf即可。

本人看到后，直接上StackOverflow找到了个bf中`if(a>=b)`的实现，之后手动进行了一下内存管理，大力写了一段bf，多试几次得到flag。(代码没存)

0x14 超简易的网盘服务器

进入Public下载一下Dockerfile和nginx.conf

发现关键点：Public和根目录各有一个h5ai

同时nginx.conf有缺陷：会先匹配.php而不是/的auth

看nginx.conf，找到h5ai的真正入口。因此可以直接访问/_h5ai/public/index.php

考虑怎么得到/flag.txt。进入网盘，直接点文件，发现是直接请求的文件本身，不能利用。

文件前有复选框，勾上后发现可以下一个tar文件。Network+curl修改一下h5ai的入口，把请求文件改为flag.txt，得到flag

(实际上本人是翻h5ai源码才发现存在这个download的api的，不过后续是借助浏览器完成的)

0x15 超安全的代理服务器

第一问:

```
nghttp https://146.56.228.227/
```

得到secret和flag

第二问差评：本人试了好半天H2发connect，总是被回Protocol Error。正纳闷这题怎么解释H2 Proxy，然后试了一下，发现curl似乎能直接以H1.1 CONNECT的方式连接

我：？

之后题目让干啥我就干啥：设secret，加referer，连接ip改成0.0.0.0来连localhost

```
curl http://0.0.0.0:8080 -x https://146.56.228.227/ -v -k --proxy-insecure --  
proxytunnel -H "Referer: https://146.56.228.227/" --proxy-header "Secret:  
abdad49e9a"
```

本人试了python-hyper,libcurl等不下十种直接发包工具，试了nginx,caddy,apache作反代，实在是没想到是个这。差评

0x16 超迷你的挖矿模拟器

心疼出题人土老师1s

本人似乎是唯一——一个预期解

不过大概也和本人当年天天爆破mc种子有关

翻源代码：这个地图生成方式和mc村庄几乎一模一样，显然是要爆破种子。

爆破的具体方法不写了，各位参考官方wp和以下文章吧

<https://www.mcbbs.net/thread-816891-1-1.html>

之后考虑如何获取flag：查看 `Game.java:30`，发现要想获取flag必须使前后材料不同。

想到在SHORT_DURATION的等待时间内，将不是flag的方块变成flag

(在没有AIR的情况下)显然只能是利用reset。发现reset后的种子只有3bit是完全未知的，验证了爆破种子的猜想。

最终做法是：爆破出种子，预测可能接下来的8个种子。同时发挖这8个地方的请求，然后重置游戏，必有一个能挖出来。

审查源码，发现flag分为两种，要么无论种子此位置都有，要么由种子决定位置。写代码时要注意避免。

还有要判断坐标范围的问题。

本来想本地搭个localhost测试，结果发现作者用var强行不让用j8，不得不装了个openjdk11。用cpp写还要担心数据类型问题，最后还是强行编译了一下然后用jd-gui开了一下确认数据类型无误。

(附件6)

0x17 中间人

三道题都要先拉出flag长度。方法显然：本地钦定一个flag，构造16种输入，看和远程的长度分布是否一样。

第一问：首先利用CBC模式的性质：通过改变iv可以随意裁剪出中间的任意块。

不考虑padding问题，Bob可以返回一个hash正确与不正确的信息。一个想法是：逐个字符逐个枚举猜flag，让Bob告诉这个猜的字符对不对。

因此可以考虑，通过调整name长度，将flag的最后一个字符单独成块。之后计算一个猜测的字符的SHA256接到块后边。Alice加密后，将中间这一部分裁剪出来发给Bob进行验证。

找到最后一块后，其余同理

第二、三问：看到所有人都是同时过第二问和第三问，于是我也直接看第三问。

CRC本身有个很好的性质： $\text{crc}(a) \oplus \text{crc}(b) \oplus \text{crc}(c) = \text{crc}(a \oplus b \oplus c)$

简单推导一下，可以发现：翻转原mess的一个特定位，对最终crc也是翻转某些特定位。这些特定位仅和crc多项式P和翻转位置pos有关。

再转化一下：对于mess的可预测变化，crc的变化也是可预测的，观察和预测方式均为异或。

将这些crc的性质全部套上一层HMAC，试验发现仍然成立(mess长度不能改变)

因此考虑利用Alice计算的crc校验码：猜测flag的一个字符，然后将其它可控的块进行变化，使得在猜测正确的情况下crc不变。

再观察CBC模式的性质：如果块调换顺序，对明文的影响也是可计算的(异或上操作之前和操作之后的前序块的密文)。这达到了一定程度上的可预测性，考虑利用这个构造特定crc。

具体地：将含有flag最后一个字符的块与其它地方的块交换一下。枚举这个字符，并计算在此情况下对明文的改变。上文提到，可预测的明文变化对crc变化是固定的，因此将其算出。

现在只需要，如何通过调换其它块的顺序，使得可以抵消掉明文交换所造成的crc变化。

我们发现，如果对调两个相邻密文块，只会影响三个明文块的值。

为了减少不可预料的影响，可以考虑三个块一组：每组前两个块对调，计算变化前后明文对最终crc的影响。

选择一些组进行对调操作，使得恰好能抵消影响。运用 线性基 算法，即可得到选择哪些组。

为了留一点冗余，本人构造了150组。

注意到这个线性基并不能构造完全，不过已经能覆盖所有可能的crc的变化了。

(本人并不懂抽代那一套理论)

(附件7)

0x18 不经意传输

第一问： $v=x0$ ，则 $m0_=m0$

第二问：

本人做题方向实在奇怪，以下是真实的心路历程。

具体是这样：因为RSA的d未知，因此 $m0_$ 和 $m1_$ 中加的两个数都是完全不可预测的随机信息，可以近似认为具有2048bit的信息量。

但是考虑一下，可以通过构造 v ，使得这两个数具有一定的关系。考虑到 $m0$ ， $m1$ ，总共需要获得的信息量为3072bit。

我：？明明只有2048bit的物理信道宽度，凭什么传输3072bit。

于是从头阅读代码，突然发现 $m0$ 和 $m1$ 的信息量并不是1024bit，而是512bit。

具体的，每一位都是[0-9][a-f]的可见字符。

计算一下，发现正好达到理论最高信息传输量（不考虑RSA）。

考虑如何使两个的d次方相关。一个最简单的想法是，构造 $v=(x0+x1)/2$ 。

则两个d次方恰为相反数。二者相加即可得到 $m0+m1$ ，正好1024bit。

看起来已经得到答案了，实则未必：这 $m0+m1$ ，真的有1024bit吗？

例如看第七位：必然恒为0。这显然不行，没有充分利用信道，丢失了信息。

写出所有可能数字的二进制表示。一个显然的想法是，将 $m0$ 进行移位，即计算 $(m0<<3)+m1$ 。

但这一定最优吗？未必。既然我们已经知道问题出在哪了：每一位出现0/1的概率并不相同。那么可以考虑把这个式子升级一下，即计算 $A*m0+B*m1$ 。利用程序自动找到对于每个bit/byte，信息熵最大的AB组合。

由于当AB较小的时候，答案的每个byte几乎只与 $m0,m1$ 的该位置byte和下一个位置的byte相关。因此可以暴力枚举算一下信息熵。

找到AB后，从低位到高位按bit枚举可能的 $m0$ 和 $m1$ ，并进行简单的大小范围和求和的检验。

之后对于所有可能的情况，利用RSA的性质进行检验（即验证 $(m0_ - m0)^e \equiv v - x0 \pmod{N}$ ）

最终对于AB采用了(61,43)，这个组合，跑的比第一和官方都快（雾）。到最后有约1/2的概率情况数只有不到一万种，纯python可两分钟出解。因此其实这题也可以再增大bit数。

那么问题只剩：如何得到 $A*m0+B*m1$ 了。愉快的推公式时间来了（最终程序定义略有差别）。

定义： $D = x0 - x1$ $V = v - x0$

则有： $m0_ \equiv m0 + V^d \pmod{N}$ $m1_ \equiv m1 + (V + D)^d \pmod{N}$

因此： $Am0 + Bm1 + AV^d + B(V + D)^d \equiv m0_ + m1_ \pmod{N}$

如果能抵消关于V的项，枚举右侧加上整数倍的N即可得到 $A*m0+B*m1$ 的可能值。这种可能性其实非常少，几乎是唯一的。

考虑抵消，有：

$$AV^d \equiv -B(V + D)^d \pmod{N}$$

同时开 d 次根，有：

$$(A(-B)^{-1})^{\frac{1}{d}} V \equiv V + D$$

欧拉公式，有：

$$((A(-B)^{-1})^e - 1)V \equiv D$$

因此已经可以直接计算出V的值了，进而推出v。

提交后本地跑，快速得到flag。

(附件8)

0xFE 感想

题出的好！难度适中，覆盖知识点广，题目又着切合实际的背景，解法比较自然。给出题人点赞！

明年一定再来。

0xFF 附件

附件1

```
def readfile(filename, encode="utf-8"):
    f=open(filename, encoding=encode)
    sr=f.read()
    f.close()
    return sr

def savefile(filename, content):
    f=open(filename, 'w')
    f.write(content)
    f.close()

def dect(t):
    #print(t)
    if len(t)==0:
```

```

        return 0
jbl="零壹贰叁肆伍陆柒捌玖"
dw="拾佰"
ans=0
if "佰" in t:
    a,b=t.split("佰")
    ans+=dect(a)*100
    ans+=dect(b)
    return ans
if "拾" in t:
    a,b=t.split("拾")
    if len(a)==0:
        ans+=10
    else:
        ans+=dect(a)*10
    ans+=dect(b)
    return ans
sp=t[-1]
for i in range(10):
    if jbl[i]==sp:
        return i
print("ERR")
print(t)

def toi(x):
    if len(x)==0:
        return 0
    ans=0
    if "元" in x:
        a,b=x.split("元")
        ans+=dect(a)
        ans+=toi(b)
        return ans
    if "角" in x:
        a,b=x.split("角")
        ans+=dect(a)*0.1
        ans+=toi(b)
        return ans
    if "分" in x:
        return dect(x[:-1])*0.01
    if x=="整":
        return 0
    print("ERR")
    print(x)

g=readfile("bills.csv").split('\n')
jb=0
for x in g:
    if len(x)==0:
        break
    a,b=x.split(',')
    b=int(b)
    xx=toi(a)
    jb+=xx*b
    #print(xx,a)

print(jb)

```


附件2

```
//I
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<queue>
#include<set>
#include<map>
#include<cstdlib>
#include<ctime>
using namespace std;
template<typename __T>
inline void read(__T &x)
{
    x=0;
    int f=1;char c=getchar();
    while(!isdigit(c)){if(c=='-') f=-1;c=getchar();}
    while(isdigit(c)) {x=x*10+c-'0';c=getchar();}
    x*=f;
}
typedef pair<int,int>pi;
pi mpi(int a,int b)
{
    return make_pair(a,b);
}
const int mod=1000000007;
long long qpow(long long a,long long b=mod-2)
{
    long long ans=1;
    while(b)
    {
        if(b&1) ans=ans*a%mod;
        b>>=1;
        a=a*a%mod;
    }
    return ans;
}

long long rand(long long l,long long r)
{
    long long a=rand();
    long long b=rand();
    long long c=rand();
    a=((a<<30)|(b<<15)|c);
    long long siz=(r-l+1);
    return l+a%siz;
}

int n=50;
char inimp[100][100];
char mp[100][100];
```

```

void print()
{
    for(int i=0;i<n;i++)
    {
        cout<<mp[i];
        cout<<endl;
    }
    cout<<endl;
}

int fxx[8]={-1,-1,-1,0,1,1,1,0};
int fxy[8]={-1,0,1,1,1,0,-1,-1};
int getcnt(int x,int y)
{
    int sum=0;
    for(int i=0;i<8;i++)
    {
        int nx=x+fxx[i];
        int ny=y+fxy[i];
        if(nx<0 || nx>=n || ny<0 || ny>=n) continue;
        sum+=(mp[nx][ny]=='#');
    }
    return sum;
}

char tmp[100][100];
void gen(int rnd=200)
{
    for(int rd=0;rd<rnd;rd++)
    {
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
            {
                int gg=getcnt(i,j);
                if(gg<2) tmp[i][j]='.';
                else if(gg>3) tmp[i][j]='.';
                else
                {
                    if(mp[i][j]=='#') tmp[i][j]=='#';
                    else
                    {
                        if(gg==3) tmp[i][j]=='#';
                        else
                            tmp[i][j]='.';
                    }
                }
            }
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                mp[i][j]=tmp[i][j];
        //print();
    }
}

void ccmp()
{
    for(int i=0;i<n;i++)
        for(int j=0;j<n;j++)

```

```

        mp[i][j]=inimp[i][j];
    }
    bool ck()
    {
        if(mp[5][45]=='#') return 0;
        if(mp[5][46]=='#') return 0;
        if(mp[6][45]=='#') return 0;
        if(mp[6][46]=='#') return 0;
        if(mp[25][45]=='#') return 0;
        if(mp[25][46]=='#') return 0;
        if(mp[26][45]=='#') return 0;
        if(mp[26][46]=='#') return 0;
        return 1;
    }

    bool dot()
    {
        int rat=200;// per 1k
        cmp();
        for(int i=0;i<15;i++)
            for(int j=0;j<15;j++)
            {
                int sb=rand(0,999);
                mp[i][j]=(sb<rat)?'#': '.';
            }
        print();
        gen();
        print();
        return ck();
    }

    int main()
    {
        long long seed=time(NULL);
        srand(seed);
        freopen("imp","r",stdin);
        for(int i=0;i<n;i++)
        {
            cin>>inimp[i];
        }
        while(dot()==0);
        return 0;
    }

```

附件3

```

import logging
import select
import socket
import struct
import socks
from socketserver import ThreadingMixIn, TCPServer, StreamRequestHandler

logging.basicConfig(level=logging.DEBUG)
SOCKS_VERSION = 5

```

```

class ThreadingTCPServer(ThreadingMixIn, TCPServer):
    pass

class SocksProxy(StreamRequestHandler):
    def handle(self):
        logging.info('Accepting connection from %s:%s' % self.client_address)
        remote = socket.socket()
        remote.connect(("202.38.93.111",0))
        self.exchange_loop(self.connection, remote)

    def exchange_loop(self, client, remote):
        while True:
            r, w, e = select.select([client, remote], [], [])
            if client in r:
                data = client.recv(4096)
                if remote.send(data) <= 0:
                    break
            if remote in r:
                data = remote.recv(4096)
                if client.send(data) <= 0:
                    break

if __name__ == '__main__':
    with ThreadingTCPServer(('0.0.0.0', 8080), SocksProxy) as server:
        server.serve_forever()

```

附件4

- web.js

```

function post (data) {
    let url = 'http://localhost:5000/que',
        xhr = new XMLHttpRequest();

    xhr.open('post', url, false);
    xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
    xhr.send(data);
    console.log(xhr.responseText);
    return xhr.responseText;
}

function toget()
{
    cont=document.querySelector("body > div > div > div > center > p").innerText
    return post("txt="+encodeURIComponent(cont))
}

ans=toget()
document.getElementsByName("ans")[0].value=String(ans)
document.getElementsByTagName("button")[1].click()

```

- web.py

```
import os
import json
import math
import time
import re
from shutil import copyfile
from flask import Flask, request
import queue

import logging
log=logging.getLogger("mid")
log.setLevel(logging.WARNING)

app = Flask("mid")
def after_request(resp):
    resp.headers['Access-Control-Allow-Origin'] = '*'
    return resp

app.after_request(after_request)


import re
import requests as rqs
import json
import time

proxyList={
    "http":"socks5h://localhost:1081",
    "https":"socks5h://localhost:1081"
}

jb=rqs.Session()
jb.headers.update({"Cookie":"PHPSESSID=114514; session=aaabbbccc"})

nms=rqs.Session()
nms.headers.update({'Cookie': "__cfduid=1; _ga=2; _gid=3; __gads=4; _gat=5;
_game_success_count=6; _game_points=7",
"user-agent": "Mozilla/5.0 (Windows NT 10.0; win64; x64) AppleWebKit/114.514
(KHTML, like Gecko) Chrome/114.514 Safari/1919.810 Edg/114.514"})

def locbk(txt,pos):
    sb=0
    for i in range(pos,len(txt)):
        if txt[i]=='(':
            sb+=1
        if txt[i]==')':
            sb-=1
        if sb==0:
            return i
def tosstex(txt):
    txt=re.sub(r"\$", "", txt)
    txt=re.sub(r"\}\{", "/", txt)
    txt=re.sub(r"\frac", "", txt)
```

```

txt=re.sub(r"\\left", "", txt)
txt=re.sub(r"\\right", "", txt)
txt=re.sub(r"ln", "log", txt)
txt=re.sub(r",", "*", txt)
txt=re.sub(r"\{", "(", txt)
txt=re.sub(r"\}", ")", txt)
txt=re.sub(r"\\", "", txt)
txt=re.sub(r"([\+\-\*/]) ([\+\-\*/])", "\\1*\\2", txt)
txt=re.sub(r"([\+\-\*/]) ([\+\-\*/])", "\\1*\\2", txt)
return txt

def tossint(txt):
    txt=tosstex(txt)
    txt=txt[4:-6]
    pm=txt.find('^')
    p1=txt[0:pm]
    eps=locbk(txt, pm+1)
    p2=txt[pm+1:eps+1]
    txt=txt[eps+1:]
    if txt[0]=='*':
        txt=txt[1:]
    return (eval(p1), eval(p2), txt)

def getpage():
    r=jb.get("http://202.38.93.111:10190/")
    tx=r.text
    p1=tx.find("<p>")
    p2=tx.find("$</p>")+1
    tx=tx[p1+4:p2]
    lol=tossint(tx)
    print(lol)
    return lol

def queryans(tup):
    r=nms.post("https://zh.numberempire.com/definiteintegralcalculator.php", data=
{"function":tup[2], "var": "x", "a":tup[0], "b":tup[1], "_p1":2110}, proxies=proxyList
)
    tx=r.text
    tx=tx[tx.find("result1>")+8:]
    tx=tx[:tx.find("</span>")]
    print(tx)
    return tx

@app.route('/que', methods=["POST"])
def que():
    txt=(request.form["txt"])
    tu=tossint(txt)
    print(tu)
    an=queryans(tu)
    #time.sleep(1)
    return an

app.run(host="0.0.0.0", port=5000, debug=False, threaded=True)

```



```

dp[a][b][mv]=ss

upl={}
for v in dp[a][b]:
    for t in range(1,50):
        ss="("+ "~"*t+dp[a][b][v]+")"
        mv=v+t
        if len(ss)>256:
            continue
        if not mv in upl:
            upl[mv]=ss
        else:
            if len(ss)<len(upl[mv]):
                upl[mv]=ss
    for t in range(1,50):
        ss="("+ "~"*t+dp[a][b][v]+")"
        mv=v-t
        if len(ss)>256:
            continue
        if not mv in upl:
            upl[mv]=ss
        else:
            if len(ss)<len(upl[mv]):
                upl[mv]=ss
for mv in upl:
    if mv>114514*5:
        continue
    ss=upl[mv]
    if len(ss)>256:
        continue
    if not mv in dp[a][b]:
        dp[a][b][mv]=ss
    else:
        if len(ss)<len(dp[a][b][mv]):
            dp[a][b][mv]=ss

print("FIN",a,b,len(dp[a][b]))

```

```

li=[]
for x in dp[0][5]:
    it=dp[0][5][x]
    li.append((x,it))

li.sort()

for x in li:
    print(x)

```

附件6

流程：编译ustc.cpp后，执行getinfo.py，将信息输出到pos文件。

手动运行cpp程序，手动复制9个坐标到send.py，得到flag

- ustc.cpp


```

#include<iostream>
#include<cstring>
#include<cstdio>
#include<algorithm>
#include<queue>
#include<bitset>
#include"anti.h"
//extern "C"
//{
// #include "finders.h"
// #include "generator.h"
//}
using namespace std;
template<typename T>
inline void read(T& s)
{
    s=0;int f=1;char c=getchar();
    while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
    while(c>='0' && c<='9'){s=s*10+c-'0';c=getchar();}
    s*=f;
}
long long rmod(long long a,long long b)
{
    long long c=a%b;
    if(c<0) return c+b;
    return c;
}
int tpxx[20],tpyy[20];
typedef bitset<(1<<20)> bs2;
bitset<(1<<20)> anti(int x,int y)
{
    bitset<(1<<20)> ret;
    int fx=x;
    int fy=y;
    if(fx<0)    fx-=31;
    if(fy<0)    fy-=31;
    fx/=32;
    fy/=32;
    int px=x-32*fx;
    int py=y-32*fy;
    for(int s=0;s<(1<<20);s++)
    {
        mclib mc(s);
        for(int i=4;i<=4;i++)
        {
            if(ret[s]==1)
                break;
            random rd(mc.getsed(i,fx,fy));
            for(int j=0;j<16;j++)
            {
                int rdx=(rd.nexti()*((1<<j)-1)+fx+1)&15;
                int rdy=(rd.nexti()*((1<<j)-1)+fy+1)&15;
                if(rdx!=(px&15))    continue;
                if(rdy!=(py&15))    continue;
                ret[s]=1;
                break;
            }
        }
    }
}

```

```

    }
    return ret;
}

int getison(int x,int y,u1l sed)
{
    mclib mc(sed);
    int ckx=x/32;
    int cky=y/32;
    rrandom rd(mc.getsed(4,ckx,cky));
    for(int i=0;i<16;i++)
    {
        int rdx=rmod((rd.nexti()*((1<<i)-1)+ckx+1),48);
        int rdy=rmod((rd.nexti()*((1<<i)-1)+cky+1),48);
        //cout<<rdx<<' '<<rdy<<endl;
        if(rdx!=(x&31)) continue;
        if(rdy!=(y&31)) continue;
        cout<<"HIT "<<i<<' '<<rdx<<' '<<rdy<<endl;
        return 1;
    }
    return 0;
}

int n=0;
int xx[100]={-248,-177,148,3,71,111,44,22,134,303,320,370};
int yy[100]={-18,-73,9,38,132,142,139,179,245,229,245,256};
int fxx[100],fyy[100];
bs2 q;
vector<u1l>a1ns;
int main()
{
    //OBS
    // cout<<getison(1,34,-578018873004245340811)<<endl;
    // cout<<getison(1,34,18139168757820811)<<endl;
    freopen("pos","r",stdin);
    cin>>n;
    n=min(n,64);
    for(int i=0;i<n;i++)
    {
        cin>>xx[i]>>yy[i];
    }
    for(int i=0;i<n;i++)
    {
        fxx[i]=xx[i];
        if(fxx[i]<0) fxx[i]==31;
        fyy[i]=yy[i];
        if(fyy[i]<0) fyy[i]==31;
        fxx[i]/=32;
        fyy[i]/=32;
        // cout<<i<<' '<<xx[i]<<' '<<yy[i]<<' '<<fxx[i]<<' '<<fyy[i]<<endl;
    }
    q=~q;
    for(int i=0;i<n;i++)
    {
        q&=anti(xx[i],yy[i]);
        cout<<"FIN"<<i<<endl;
        flush(cout);
    }
    //u1l x=0;

```

```

for(int i=0;i<(1<<20);i++)
    if(q[i])
    {
        // x=i;
        aIns.push_back(i);
        cout<<"LOCATED "<<i<<endl;
        flush(cout);
    }
//return 0;
// aIns.push_back(684640);
int cnt=0;
ull fsd=0;
for(auto x:aIns)
{
    for(ull i=0;i<(1<<28);i++)
        //for(ull i=172988593;i<=172988593;i++)
        {
            if ((i&((1<<24)-1))==0)
            {
                cout<<"NT"<<i<<endl;
                flush(cout);
            }
            mclib mc(x+(i<<20));
            // cout<<(x+(i<<20))<<endl;;
            bool afg=1;
            for(int j=0;j<n && afg;j++)
            {
                bool fg=0;
                for(int ti=4;ti<=4;ti++)
                {
                    if(fg==1)
                        break;
                    // cout<<"FF"<<j<<' '<<xx[j]<<' '<<yy[j]<<' '<<fkx[j]<<'
'<<fky[j]<<endl;
                    rrandom rd(mc.getsed(ti,fkx[j],fky[j]));
                    for(int tj=0;tj<16;tj++)
                    {
                        // cout<<"FF"<<j<<' '<<xx[j]<<' '<<yy[j]<<' '<<fkx[j]<<'
'<<fky[j]<<endl;

                        int rdx=rmod((rd.nexti()*((1<<tj)-1)+fkx[j]+1),48);
                        int rdy=rmod((rd.nexti()*((1<<tj)-1)+fky[j]+1),48);
                        // cout<<"RD: "<<tj<<' '<<rdx<<' '<<rdy<<endl;
                        if(rdx!=(xx[j]&31)) continue;
                        if(rdy!=(yy[j]&31)) continue;
                        fg=1;
                        break;
                    }
                }
                // cout<<j<<' '<<xx[j]<<' '<<yy[j]<<' '<<fg<<endl;
                afg&=fg;
            }
            if(afg==1)
            {
                cout<<"FOUND"<<x+(i<<20)<<endl;
                fsd=x+(i<<20);
                flush(cout);
            }
        }
    }
}

```

```

        if(fsd==0)
            continue;
        int ck1=16777216;
        int md1=ck1+ck1/2;
        cout<<"SEED "<<fsd<<endl;
        for(int lst=0;lst<9;lst++)
        {
            ull jsd=(fsd<<3)|lst;
            if(lst==8)
                jsd=fsd;
            mc1ib mc(jsd);

            for(int cx=0;cx<5;cx++)
            {
                for(int cy=0;cy<5;cy++)
                {
                    for(int i=5;i<=5;i++)
                    {
                        rrandom rd(mc.getsed(i,cx,cy));
                        for(int j=0;j<2;j++)
                        {
                            int rdx=rmod((rd.nexti()*((1<<j)-1)+cx+1),md1);
                            int rdy=rmod((rd.nexti()*((1<<j)-1)+cy+1),md1);
                            if(j==0) continue;
                            if (rdx<ck1 && rdy<ck1)
                            {
                                cout<<"FLAG: "<<lst<<' '<<cx*ck1+rdx<<'
'<<cy*ck1+rdy<<endl;

                                tpxx[lst]=cx*ck1+rdx;
                                tpyy[lst]=cy*ck1+rdy;
                            }
                        }
                    }
                }
            }
            flush(cout);
        }
        for(int i=0;i<9;i++)
            printf("(%d,%d),\n",tpxx[i],tpyy[i]);
    }
    return 0;
}

```

- anti.h

```

typedef unsigned long long ull;
struct random
{
    ull multip=0x5DEECE66D;
    ull addp=0xB;
    ull last=0;
    ull mask=(1ll<<48)-1;
    void init(ull sed)
    {
        last=(sed^multip)&mask;
    }
    int nexti()

```

```

    {
        last=(last*multip+addp)&mask;
        return last>>(48-32);
    }
    int next()
    {
        last=(last*multip+addp)&mask;
        return last>>(48-31);
    }
    int nextint(int x)
    {
        return next()%x;
    }
    rrandom(u11 sed=0)
    {
        init(sed);
    }
};
struct mclib
{
    u11 sed;
    //u11 mu1=341873128712;
    //u11 mu2=132897987541;
    void init(u11 x)
    {
        sed=x;
    }
    mclib(u11 x=0)
    {
        init(x);
    }
    u11 getsed(int p,u11 a,u11 b)
    {
        return sed^(p+0x6E5D5AF15FA1280Bu11*a+0xE9716B1CE6339E6Cu11*b);
    }
};

```

- getinfo.py

```

import requests as rqs
import json

jb=rqs.Session()
jb.headers.update({"Cookie": "session=114514:1919810"})

def mkreq(met,dat):
    dat["token"]="114514:1919810"
    r=jb.post("http://202.38.93.111:10169/api/"+met,params=dat)
    #r=jb.post("http://localhost:8088/api/"+met,params=dat)
    return r.text

def reset():
    mkreq("reset",{})
    #print()

def getchunk(ckx,cky):
    rt=json.loads(mkreq("state",{ "x":ckx*32, "y":cky*32}))

```

```

        return rt["materials"]

def domine(mx,my):
    rt=json.loads(mkreq("damage",{ "x":mx, "y":my, "material":"FLAG"}))
    reset()

gl=[]
for a in range(0,6):
    #print(len(gl))
    for b in range(0,6):
        mif=getchunk(a,b)
        for x in range(32):
            for y in range(32):
                #print(mif[x][y])
                if mif[x][y]=="OBSIDIAN":
                    gl.append((a*32+x,b*32+y))

print(len(gl))
for a,b in gl:
    print(a,b)

```

- send.py

```

import requests as rqs
import json
import _thread
import time

jb=rqs.Session()
jb.headers.update({"Cookie": "session=114514:1919810"})

def mkreq(met,dat):
    dat["token"]="114514:1919810"
    r=jb.post("http://202.38.93.111:10169/api/"+met,params=dat)
    #r=jb.post("http://localhost:8088/api/"+met,params=dat)
    return r.text

def reset():
    print(mkreq("reset",{}))

def getchunk(ckx,cky):
    rt=json.loads(mkreq("state",{ "x":ckx*32, "y":cky*32}))
    return rt["materials"]

def domine(mx,my):
    print(mx,my)
    rt=json.loads(mkreq("damage",{ "x":mx, "y":my, "material":"FLAG"}))
    print(rt)

mn1=[(61746934,68779701),
      (75170139,3690429),
      (76791398,2344681),
      (68439752,49624564),
      (82400664,27808754),
      (68220730,66468796),
      (70938717,82993007),

```

```

(54979585,67433954),
(78880385,47709289)]

for i in range(0,8):
    _thread.start_new_thread( domine, mnl[i] )

time.sleep(0.5)
reset()

while 1:
    pass

```

附件7

- MITM1.py

```

from pwn import *
from Crypto.Cipher import AES
from hashlib import sha256
import os
from hashlib import sha256
from utils import *
import time

#r=process("python3 MITM1.py",shell=True)

r=remote("202.38.93.111",10041)
print(r.recv())
r.send(b"114514:1919810\n")
time.sleep(1)
r.send(b"1\n")
time.sleep(1)

def genmess(nm,et,fg=b"flxgagasdfddd0123456789abcdef0123456789abcdef"):
    return
    nm=bytes.fromhex(nm)
    et=bytes.fromhex(et)
    msg = b"Thanks " + nm + b" for taking my flag: " + fg + et
    plaintext = msg + sha256(msg).digest()
    plaintext=pad(plaintext)
    print(plaintext)
    print(plaintext.hex())
    print(len(plaintext))

def doreq(P,M):
    if P=='A':
        r.send(b"Alice\n")
        r.send(M[0].encode()+b"\n")
        r.send(M[1].encode()+b"\n")
        while True:
            x=r.recv().decode()
            #print(x)
            if not "encrypted" in x:
                continue
            x=x.strip().split(':')[1].strip()
            if len(x)==0:
                x=r.recv().decode().strip()

```

```

        #print(x)
        return x
    if P=='B':
        r.send(b"Bob\n")
        #print(M)
        r.send(M.encode()+b"\n")
        while True:
            x=r.recv().decode()
            #print(x)
            if (not "Thanks" in x) and (not "problem" in x):
                continue
            if "Thanks" in x:
                return 1
            return 0

#flag:32+13 = 45

#32-127
def genext(suf):
    ret="00"*(48-len(suf))
    ret+=sha256((suf+"\x00"*(48-len(suf))).encode()).digest().hex()
    ret+="10"*16
    return ret

def genname(nt):
    return "00"*(80-28+nt+3)

def tryn(nowc,okc):
    gl=len(okc)
    quen=genname(gl+1)
    qext=genext(nowc+okc)
    genmess(quen,qext)
    mess=doreq('A',(quen,qext))
    #print("MESSA: ",mess)
    res=doreq('B',mess[8*32:15*32])
    print(res)
    return res

act=[chr(x) for x in range(1,128)]

nowpx=""
for i in range(45):
    for x in act:
        #print(x,end='')
        #print(x)
        r1=tryn(x,nowpx)
        if r1==1:
            nowpx=x+nowpx
            break
    print("\n")
    print("OK",nowpx)

```

- MITM3(2).py 其中2的差异在代码中注释已写明。

```

from pwn import *
from Crypto.Cipher import AES
from hashlib import sha256

```



```
import os
from hashlib import sha256
from utils import *
import time
import random
import os

#r=process("python3 MITM3.py",shell=True)

r=remote("202.38.93.111",10041)
print(r.recv())
r.send(b"11451:1919810\n")
time.sleep(1)
r.send(b"3\n") #MITM2.py: r.send(b"3\n")
time.sleep(1)

BLS=150

def gencrc128(crc):
    crc^=((1 << 128) - 1)
    crc = (crc >> 1) ^ (0xB595CF9C8D708E2166D545CF7CFDD4F9 & -(crc & 1))
    return crc^((1 << 128) - 1)

def sbcsc(msg,key=b"aq2rvxPq9adcUICm"):
    cz=hmac_crc128(key,b"\x00"*len(msg))
    return xor(hmac_crc128(key,msg),cz)

def
genmess(nm,et,fg=b"flagflxgflxgflxgflxgflxgflxgflxgflxgflxgffffffl1111lxg"):
    return
nm=bytes.fromhex(nm)
et=bytes.fromhex(et)
msg = b"Thanks " + nm + b" for taking my flag: " + fg + et
plaintext = msg + sha256(msg).digest()
plaintext=pad(plaintext)
print(plaintext)
print(plaintext.hex())
print(len(plaintext))

def doreq(P,M):
    if P=='A':
        r.send(b"Alice\n")
        r.send(M[0].encode()+b"\n")
        r.send(M[1].encode()+b"\n")
        while True:
            x=r.recv(409600).decode()
            #print(x)
            if not "encrypted" in x:
                continue
            x=x.strip().split(':')[0].strip()
            if len(x)==0:
                x=r.recv(409600).decode().strip()
                #print(x)
            while True:
                xx=r.recv(409600,0.1).strip()
                if len(xx)==0:
                    break
                x+=xx.decode().strip()
```

```

        #print(x)
        if "whom" in x:
            x=x[:x.find('\nwhom')]
        return x
    if P=='B':
        r.send(b"Bob\n")
        #print(M)
        r.send(M.encode()+b"\n")
        while True:
            x=r.recv(409600).decode()
            #print(x)
            if (not "Thanks" in x) and (not "problem" in x):
                continue
            if "Thanks" in x:
                return 1
            return 0

#flag:32+13 = 45
# 48+10=58

#32-127
def genext(suf):
    ret="00"*(64-len(suf))
    ret+="00"*16
    return ret

PB=b""

def genblo():
    global PB
    #PB=os.urandom(3*16*BLS)
    PB=b"\x00"*(3*16*BLS)
    return PB.hex()+"00"*16
    #myl=""
    #for i in range(BLS):
    #    myl+=("%02x"%((123*i+7)%256))*16+("%02x"%((114*i+3)%256))*16+("%02x"%
    ((514*i+6)%256))*16)
    #return myl

def genname(nt):
    return "00"*(96-28+nt+6)#MITM2.py: return "00"*(96-28+nt+24)

def gbk(bts,bid):
    return bts[bid*16:(bid+1)*16]

def getxxj(vl,msl,cv):
    rms=0
    #print(cv)
    #print(bin(cv))
    for i in range(128):
        if ((cv>>i)&1)==0:
            continue
        #print(vl[i])
        cv^=vl[i]
        rms^=msl[i]
    #print(cv)
    assert cv==0
    return rms

```

```

def getrb(mess,i,v):
    if v==0:
        return gbk(mess,15+3*i)+gbk(mess,16+3*i)+gbk(mess,17+3*i)
    return gbk(mess,16+3*i)+gbk(mess,15+3*i)+gbk(mess,17+3*i)

def tryn(okc):
    gl=len(okc)
    quen=genname(gl+1)
    qext=genext("g"+okc)
    quxx=genblo()
    genmess(quen,qext+quxx)
    mess=doreq('A',(quen,qext+quxx))
    #print(mess)
    mess=bytes.fromhex(mess)
    civ=mess[:16]
    mess=mess[16:]
    yx=[]
    for i in range(BLS):
        #15+3*i 16+3*i 17+3*i
        fum=gbk(PB,0+3*i)+gbk(PB,1+3*i)+gbk(PB,2+3*i)
        gueorc=sbcsc(fum+b"\x00"*(16*(3*BLS+15-17-3*i)))
        bl1m=xor(gbk(PB,1+3*i),xor(gbk(mess,14+3*i),gbk(mess,15+3*i)))
        bl2m=xor(gbk(PB,0+3*i),xor(gbk(mess,14+3*i),gbk(mess,16+3*i)))
        bl3m=xor(gbk(PB,2+3*i),xor(gbk(mess,15+3*i),gbk(mess,16+3*i)))
        fkm=bl1m+bl2m+bl3m+b"\x00"*(16*(3*BLS+15-17-3*i))
        myx=int.from_bytes(gueorc,"big")^int.from_bytes(sbcsc(fkm),"big")
        #print(bl1m+bl2m+bl3m,myx)
        yx.append(myx)
    rmsk=[]
    rv=[]
    #doreq('B',(civ+mess[:15*16]+gbk(mess,16)+gbk(mess,15)+mess[17*16:]).hex())
    #print(yx)
    #print(int.to_bytes(yx[0],16,"big").hex())
    for i in range(BLS):
        rmsk.append((1<<i))
        rv.append(yx[i])
    for i in range(128):
        myj=-1
        for j in range(i,BLS):
            if ((rv[j]>>i)&1)==1:
                myj=j
                break
        #print(i,myj)
        #if myj==-1:
        #    print("ERR",i)
        if i!=myj:
            rv[i],rv[myj]=rv[myj],rv[i]
            rmsk[i],rmsk[myj]=rmsk[myj],rmsk[i]
        for j in range(i+1,BLS):
            if ((rv[j]>>i)&1)==1:
                rv[j]^=rv[i]
                rmsk[j]^=rmsk[i]
        #for j in range(0,BLS):
        #    print((rv[j]>>74)&1,end='')
        #print()
        #print(rv[i])
    #for i in range(BLS):

```

```

# print(rv[i])
for myc in act:
    gec=(myc+okc).encode()
    fum=gec+b"\x00"*(64-len(gec))
    gueorc=sbcsc(fum+b"\x00"*((3*BLS+1+1)*16))
    b11m=xor(xor(gbk(fum,3),gbk(mess,12)),gbk(mess,9))
    b12m=xor(xor(gbk(fum,0),gbk(mess,9)),gbk(mess,13))
    b13m=gbk(fum,1)
    b14m=gbk(fum,2)
    b15m=xor(gbk(mess,12),gbk(mess,13))
    guefrc=sbcsc(b11m+b12m+b13m+b14m+b15m+b"\x00"*(16*(3*BLS+1)))
    ndc=int.from_bytes(gueorc,"big")^int.from_bytes(guefrc,"big")
    uxj=getxxj(rv,rmsk,ndc)

bms=mess[:10*16]+gbk(mess,13)+gbk(mess,10)+gbk(mess,11)+gbk(mess,12)+gbk(mess,14)

    assert len(bms)%16==0
    for i in range(BLS):
        bms+=getrb(mess,i,(uxj>>i)&1)
    bms+=gbk(mess,15+3*BLS)
    bms+=gbk(mess,16+3*BLS)
    bms+=gbk(mess,17+3*BLS)
    ffv=(civ+bms).hex()
    res=doreq('B',ffv)
    #print(res)
    if res:
        return myc
    print(myc,end="")
#print("MESSA: ",mess)
#res=doreq('B',mess[8*32:15*32])
#print(res)
assert False

act=[chr(x) for x in range(1,128)]

nowpx=""
for i in range(58):#MITM2.py: for i in range(40):
    r1=tryn(nowpx)
    nowpx=r1+nowpx
    print("\n")
    print("OK",nowpx)

```

附件8

- Ml.py

```

import math

sb=[]
jxl="0123456789abcdef"
def dfs(nows,nowl):
    global sb
    if nowl==0:
        sb.append(nows.encode())
        return
    for i in jxl:

```

```

        dfs(i+nows,nowl-1)

def genstr1(nl):
    global sb
    sb=[]
    dfs("",nl)
    return sb

def calcs(ma,mb):
    ssl=1
    jl=genstr1(ssl+1)
    soso=[0]*256
    for sa in jl:
        for sb in jl:
            va=int.from_bytes(sa,"big")
            vb=int.from_bytes(sb,"big")
            vn=ma*va+mb*vb
            tiv=((vn>>(8*ssl))&255)
            soso[tiv]+=1
    xxs=0
    qj=len(jl)*len(jl)
    for x in soso:
        if x==0:
            continue
        xxs+=(x/qj)*math.log(qj/x)
    return xxs

#print(calcs(14,1))

miv=0
mip=(0,0)

ss=[]

for xa in range(1,64):
    for xb in range(1,64):
        vv=calcs(xa,xb)
        print(xa,xb,vv)
        ss.append((vv,xa,xb))
        if vv>miv:
            miv=vv
            mip=(xa,xb)

print(mip,miv)

ss.sort()
#print(ss)

```

- sol.py

```

from pwn import *
import time
from gmpy2 import *

#r=process("python3 OT.py",shell=True)
r=remote("202.38.93.111",10031)

```

```

print(r.recv())
r.send(b"114514:1919810\n")

CCA=43
CCB=61

def sendv(x):
    r.send(str(x).encode()+b"\n")

def check(vl,sa,sb,ll=0):
    va=int.from_bytes(sa,"big")
    vb=int.from_bytes(sb,"big")
    ckv=(va*CCA+vb*CCB)%(2**(8*ll))
    ckx=vl%(2**(8*ll))
    return ckv==ckx

def facheck(vl,va,vb):
    ckv=(va+vb)>>8
    ckv&=255
    return ckv==vl

def tryans(smv):
    lasl=[(b"",b"",0,0)]
    nowl=[]
    stl="0123456789abcdef"
    print("CALC",smv//n)
    for i in range(128):
        for nowv in lasl:
            nowa,nowb,nowia,nowib=nowv
            ndckv=(smv>>(8*i))&255
            for ca in stl:
                na=ca.encode()+nowa
                fav=ord(ca)*256*CCA+nowia
                for cb in stl:
                    fbv=ord(cb)*256*CCB+nowib
                    if facheck(ndckv,fav,fbv):
                        nb=cb.encode()+nowb
                        nowl.append((na,nb,fav>>8,fbv>>8))

        lasl=nowl
        nowl=[]
        #print(lasl)
        if len(lasl)==0:
            break
        if i>30:
            print(i,len(lasl))
    if len(lasl)==0:
        return []
    print(len(lasl))
    for x in lasl:
        nowa,nowb,_,_=x
        if check(smv,nowa,nowb):
            nowl.append((int.from_bytes(nowa,"big"),int.from_bytes(nowb,"big")))
    print(len(nowl))
    hoho=[]
    for xxx in nowl:
        vm0,vm1=xxx
        dp=(c0-vm0)%n
        cvv=pow(dp,e,n)

```

```

        if cvv==vv:
            print("FOUND:")
            print(vm0)
            print(vm1)
            hoho.append((vm0,vm1))
        return hoho

time.sleep(1)
x=r.recv().decode()
print(x)
xl=x.split('\n')
e=65537
for l in xl:
    if len(l)==0:
        continue
    a,b=l.split('=')
    a=a.strip()
    b=b.strip()
    if len(b)==0:
        continue
    if a=='n':
        n=int(b)
    if a=='x0':
        x0=int(b)
    if a=='x1':
        x1=int(b)

print(n,e,x0,x1)

D=x0-x1

vv=(n-D)*invert(1+pow(CCA*invert(CCB,n)%n,e,n),n)%n
v=(vv+x0)%n
sendv(v)
time.sleep(1)

x=r.recv().decode()
print(x)
xl=x.split('\n')
for l in xl:
    if len(l)==0:
        continue
    a,b=l.split('=')
    a=a.strip()
    b=b.strip()
    if len(b)==0:
        continue
    if a=='m0_':
        c0=int(b)
    if a=='m1_':
        c1=int(b)
    jbl=(CCA*c0+c1*CCB)%n

print(jbl)

for i in range((CCA+CCB)//4,CCA+CCB):
    ss=tryans(jbl+i*n)

```

```
r.interactive()
```