校赛writeup

原创

Ni9htMar3 于 2016-12-26 17:28:30 发布 9037 收藏 3

分类专栏： WriteUp 文章标签： writeup

本文链接：https://blog.csdn.net/Ni9htMar3/article/details/53887845

第一届校赛，总的来说对自己所处的名次还是比较满意的，这次还是比较占便宜，做出来几道简单的逆向，要不然一个弱弱的web狗肯定要被吊打，这次的crypto的题比较难，也多，尝试做了几道，后来由于难度上升，也就没怎么看这些题。希望能从各种比赛中学习到真正的知识，毕竟自己还是跟其他人有很大的差距，总要赶上大牛学长的脚步！~

sign

得到一串字符

`4C4A575851324332474E5A5445544B584B4A5446534D545947464D57594F4A544C4A4C585132544347495957595A5352485536513D3D3D3D`

明显是16进制转化，写个脚本

```
str = '0123456789ABCDEF'

def to(s):
    s1 = ''
    i = 0
    while i < len(s):
        c1 = s[i]
        c2 = s[i+1]
        i += 2
        b1 = str.find(c1)
        b2 = str.find(c2)
        s1+=chr((b1 << 4)+b2)
    return s1

s = '4C4A575851324332474E5A5445544B584B4A5446534D545947464D57594F4A544C4A4C585132544347495957595A535248
print to(s)
```

得到

`LJWXQ2C2GNZTETKXKJTFSMTYGFMWYOJTLJLXQ2TCGIYWYZSRHU6Q====`

base32解码

`ZmxhZ3s2MWRfY2x1Yl93ZWxjb21lfQ==`

base64解码

`flag{61d_club_welcome}`

flip

当初做hctf的一道题，有fp的writeup
链接(http://bobao.360.cn/ctf/learning/179.html)

BASE64?

一看就知道是base32解码

`GUYDIMZVGQ2DMN3CGRQTONJXGM3TINLGG42DGMZXGM3TINLGGY4DGNBXGYZTGNLGGY3DGNBWMU3WI===`

解出

`504354467b4a7573745f743373745f683476335f66346e7d`

貌似是16进制转化

脚本

```
def str2byte(s):
    base='0123456789ABCDEF'
    i = 0
    s = s.upper()
    s1=''
    while i < len(s):
        c1=s[i]
        c2=s[i+1]
        i+=2
        b1=base.find(c1)
        b2=base.find(c2)
        s1+=chr((b1 << 4)+b2)
    return s1

s ='504354467b4a7573745f743373745f683476335f66346e7d'
print str2byte(s)
```

flag:`PCTF{Just_t3st_h4v3_f4n}`

某神的实验室LOGO

直接扔进`stegsolve`分析即得

AES-PIC

这个题是之前对照着bystudent的writeup做的
链接(http://www.bystudent.com/?p=234)

上帝之音

> 这是一段神奇的声音，可是上帝之音似乎和无字天书一样，是我们这些凡人无法理解的，你能以上帝的角度，理解这段WAV的含义么？
> Hint1: 你们做音频题都不喜欢看时域图？
> Hint2: 在数据传输过程中，我们往往会使用一种自带时钟的编码以减少误码率
> godwave.wav.26b6f50dfb87d00b338b58924acdbea1

Audacity 打开就是一段稀奇古怪的音频信号，仔细观察，发现不同段落其幅值有明显差异，应该是调幅了，MATLAB 导入 wav 文件看数据，发现大概是以 64 个点为周期，那么取幅值高的为 1，幅值低的为 0。

```
clc;
clear;
y = audioread('godwave.wav');
he = 0;
data = [];
for i = 1:length(y)
    he = he + abs(y(i,1));
    if mod(i,64) == 0
        if he > 10
            data = [data,1];
        else
            data = [data,0];
        end
        he = 0;
    end
end
fid = fopen('data.txt','w');
for i = 1:length(data)
    fprintf(fid,'%d',data(1,i));
end
fclose(fid);
```

解出的数据是曼彻斯特编码，解码后是一张图片。

```
# coding=utf-8
with open('data.txt', 'r') as f:
    data = f.readline()
    print len(data)
    count = 0
    res = 0
    ans = ''
    key = ""
    while data != '':
        pac = data[:2]
        if pac != '':
            if pac[0] == '0' and pac[1] == '1':
                res = (res<<1)|0
                count += 1
            if pac[0] == '1' and pac[1] == '0':
                res = (res<<1)|1
                count += 1
            if count == 8:
                ans += chr(res)
                count = 0
                res = 0
        else:
            break
        data = data[2:]
with open('out.png', 'wb') as f2:
    f2.write(ans)
```

扫描二维码即可。

LOCALHOST

看来需要`localhost access only!!`直接利用**Modify Headers**直接加上`X-Forwarded-For: 127.0.0.1`即可

warmup-misc

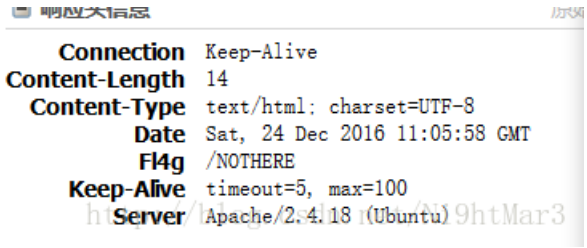
一张加密的脚本图片，逆序写出解密即得flag

```
import base64

def decode(flag,k):
    ag=base64.b64decode(flag)
    r=""
    for i in ag:
        r+=chr(ord(i)^ord(k))
    r+=i
    return r

flag="UAUSAB1QUFBQUFAbfQ=="

print decode(flag,'f')
```

flag： `6ctf{666666}`

warmup-web

打开只有一句无意义的话，源码没有东西，看头
发现



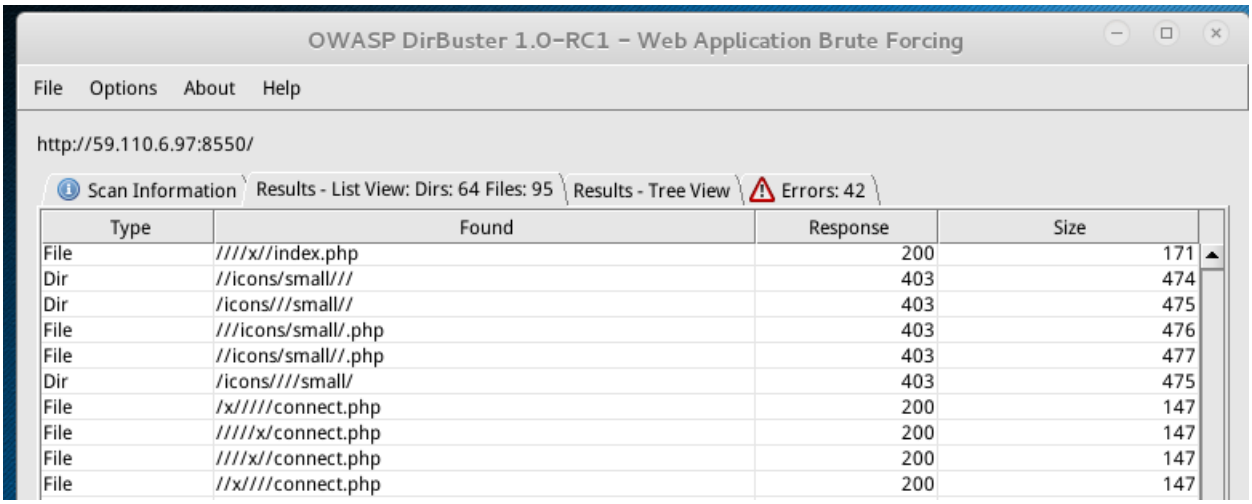
看见有`/`猜测是地址，访问即得**flag**

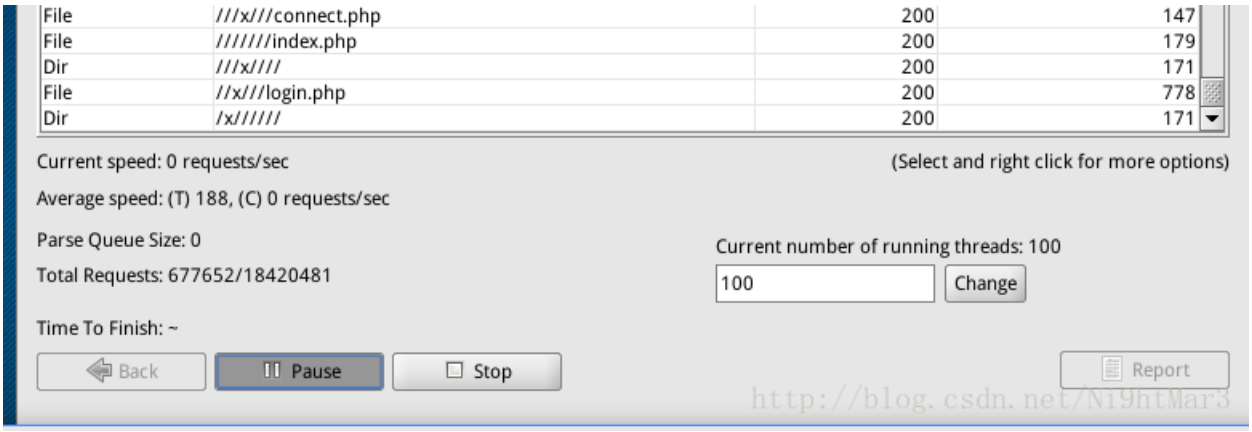
`http://59.110.6.97:8559/NOTHERE`

`flag{OpenYourHeadHole}`

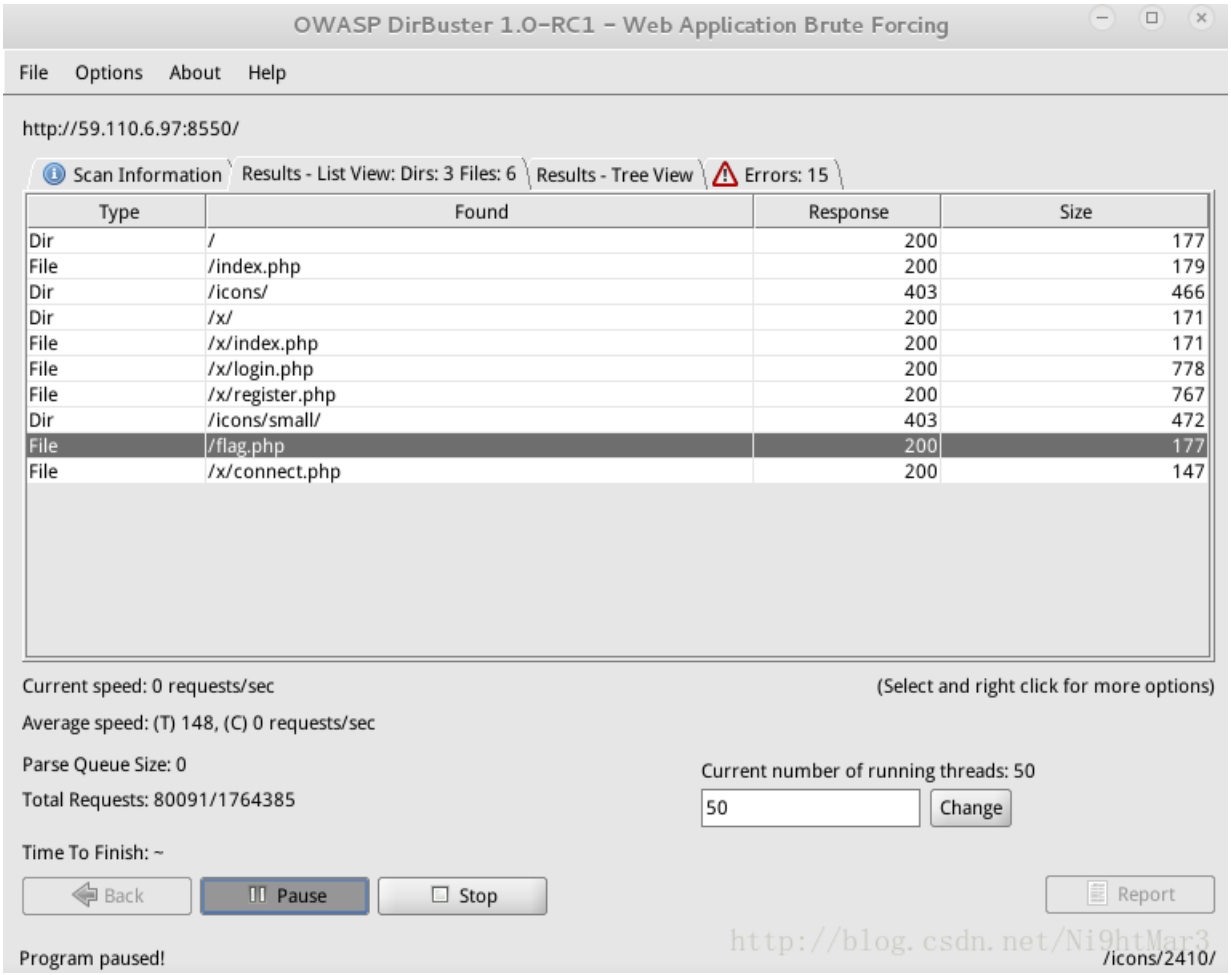
web2

打开网页，提示`Try to find the web directory`看来需要跑目录
其实一开始尝试各种方法，burpsuite抓包，头啥的都分析构造，差一点就猜对啦，真是bi了狗~
我在kali下利用DirBuster跑，第一次用一个字典跑了好久都没跑出来





曾经试了试`/x/register.php`、`/x/login.php`发现没什么用
后来换了一个字典，几乎是瞬出



郁闷，`/flag.php`一开始差一点才对，结果现在用了好长时间，真是醉啦
访问成功得到**flag**：`flag{Quickly!!!!!!!111!!!!}`

warmup-re

拖进IDA进行反编译

```
// local variable allocation has failed, the output may be wrong!
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rax@1
  __int64 v4; // rdx@1
  __int64 v5; // rax@1
```

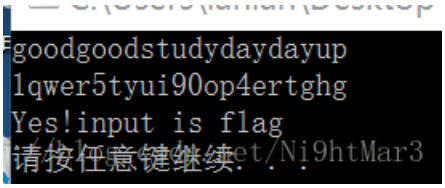
```
__int64 v6; // rbx@1
__int64 v7; // rdx@1
__int64 v8; // rax@1
__int64 v9; // rdx@1
__int64 v10; // rax@2
__int64 v11; // rdx@2
int result; // eax@2
__int64 v13; // rax@5
__int64 v14; // rdx@5
unsigned __int64 v15; // rbx@7
unsigned __int64 v16; // rax@7
__int64 v17; // rax@8
__int64 v18; // rdx@8
char v19[48]; // [sp+20h] [bp-60h]@1
char v20[48]; // [sp+50h] [bp-30h]@1
char v21[48]; // [sp+80h] [bp+0h]@4
int v22; // [sp+80h] [bp+30h]@1
int v23; // [sp+84h] [bp+34h]@1
int v24; // [sp+88h] [bp+38h]@1
int v25; // [sp+8Ch] [bp+3Ch]@1
int v26; // [sp+C0h] [bp+40h]@1
int v27; // [sp+C4h] [bp+44h]@1
int v28; // [sp+C8h] [bp+48h]@1
int v29; // [sp+CCh] [bp+4Ch]@1
int v30; // [sp+D0h] [bp+50h]@1
int v31; // [sp+D4h] [bp+54h]@1
int v32; // [sp+D8h] [bp+58h]@1
int v33; // [sp+DCh] [bp+5Ch]@1
int v34; // [sp+E0h] [bp+60h]@1
int v35; // [sp+E4h] [bp+64h]@1
int v36; // [sp+E8h] [bp+68h]@1
int v37; // [sp+ECh] [bp+6Ch]@1
int v38; // [sp+F0h] [bp+70h]@1
int v39; // [sp+F4h] [bp+74h]@1
int v40; // [sp+F8h] [bp+78h]@1
int v41; // [sp+FCh] [bp+7Ch]@1
int v42; // [sp+100h] [bp+80h]@1
char v43; // [sp+10Ah] [bp+8Ah]@4
char v44; // [sp+108h] [bp+88h]@4
int i; // [sp+10Ch] [bp+8Ch]@3

_main();
v22 = 86;
v23 = 30;
v24 = 24;
v25 = 1;
v26 = 21;
v27 = 90;
v28 = 27;
v29 = 29;
v30 = 6;
v31 = 29;
v32 = 76;
v33 = 84;
v34 = 22;
v35 = 20;
v36 = 85;
v37 = 28;
v38 = 22;
v39 = 21;
```

```
  v40 = 30;
  v41 = 29;
  v42 = 23;
  LODWORD(v3) = std::operator>><char,std::char_traits<char>>(*(_QWORD *)&argc, argv, v19, refptr__ZSt3c
  std::operator>><char,std::char_traits<char>>(*(_QWORD *)&argc, argv, v20, v3);
  LODWORD(v5) = strlen(*(_QWORD *)&argc, argv, v4, v19);
  v6 = v5;
  LODWORD(v8) = strlen(*(_QWORD *)&argc, argv, v7, v20);
  if ( v6 == v8 )
  {
    for ( i = 0; ; ++i )
    {
      v15 = i;
      LODWORD(v16) = strlen(*(_QWORD *)&argc, argv, v9, v19);
      if ( v15 >= v16 )
        break;
      v44 = v19[i];
      v43 = v20[i];
      v21[i] = v43 ^ v44;
      v9 = (unsigned int)v21[i];
      if ( (_DWORD)v9 != *(&v22 + i) )
      {
        LODWORD(v13) = std::operator<<<std::char_traits<char>>(*(_QWORD *)&argc, argv, "key error", ref
        std::ostream::operator<<(
          *(_QWORD *)&argc,
          argv,
          refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_,
          v13);
        system(*(_QWORD *)&argc, argv, v14, "pause");
        return 0;
      }
    }
    LODWORD(v17) = std::operator<<<std::char_traits<char>>(
                     *(_QWORD *)&argc,
                     argv,
                     "Yes!input is flag",
                     refptr__ZSt4cout);
    std::ostream::operator<<(
      *(_QWORD *)&argc,
      argv,
      refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_,
      v17);
    system(*(_QWORD *)&argc, argv, v18, "pause");
    result = 0;
  }
  else
  {
    LODWORD(v10) = std::operator<<<std::char_traits<char>>(*(_QWORD *)&argc, argv, "lenth error", refpt
    std::ostream::operator<<(
      *(_QWORD *)&argc,
      argv,
      refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_,
      v10);
    system(*(_QWORD *)&argc, argv, v11, "pause");
    result = 0;
  }
  return result;
}
```

主要意思就是输两串长度相同的字符串，并逐位异或，最终得到的值与v22~v42每个进行比较，相等就输出，由于第一个字符串已经给了`goodgoodstudydaydayup`这样直接脚本一跑就行啦

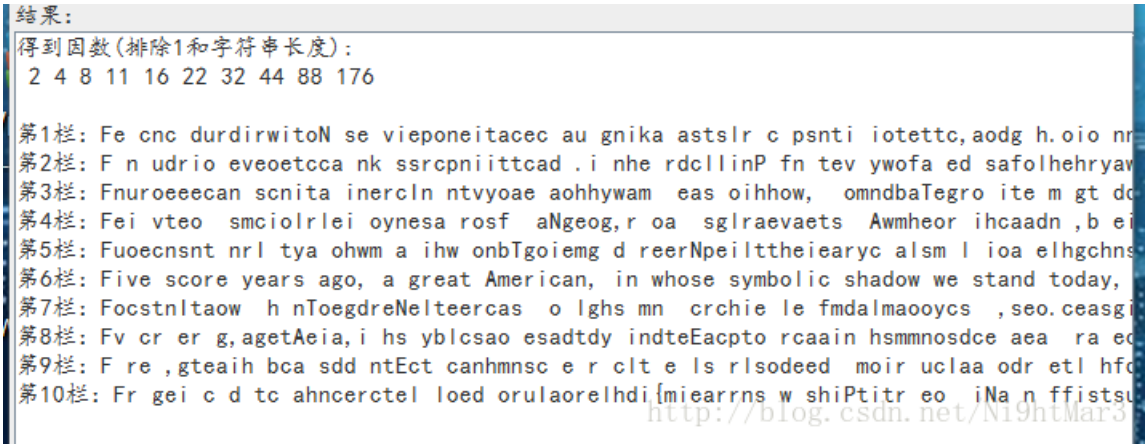
```
s='goodgoodstudydaydayup'
r=""
v=[86,30,24,1,21,90,27,29,6,29,76,84,22,20,85,28,22,21,30,29,23]
j=0
for i in s:
    r+=chr(ord(i)^(v[j]))
    j = j+1

print r
```

得到第二个字符串`1qwer5tyui90op4ertghg`
然后输进去就是**flag**



zlmm

打开好长的字符，不过看见了`{}`猜测是栅栏密码，题目名字也是，懒得分析，直接拖进工具得到flag



发现22栏的像是文章，看完发现flag
`flag{zha+lanmima666}`

warmip-crypto

一串不知道是啥的字符`BH=CWG=EO=IEI=;DEDEDEY`
这个一直想都没想出来，最后好像跟偏移有关，顿时灵光一现，开始查相关的flag格式，就找`{`、`}`，这两个相差2，找与`Y`的偏移量是2的是`W`，这样的话直接计算`Y`与`}`的偏移量，然后全员偏移就行

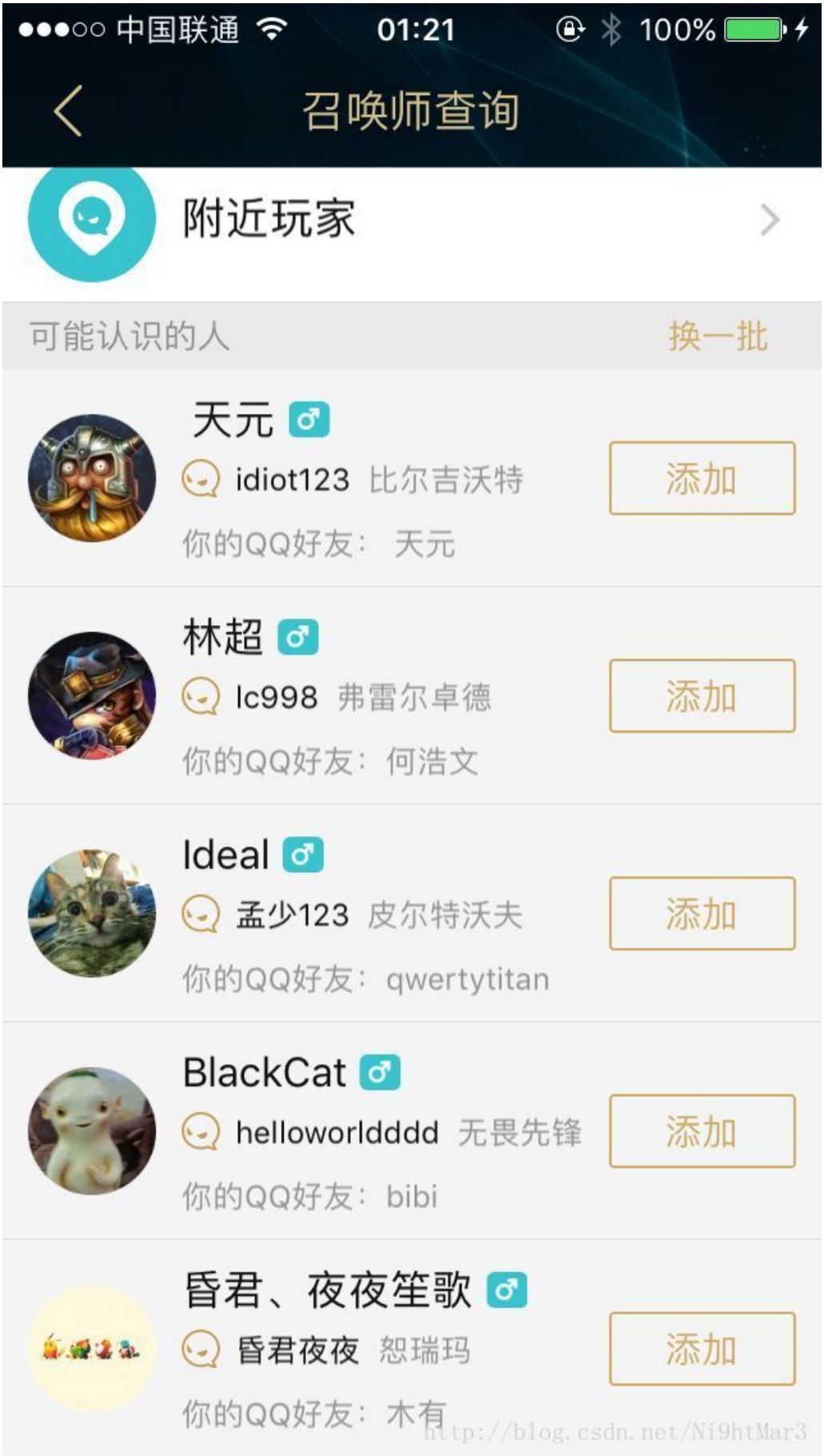
```
r="BH=CWG=EO=IEI=;DEDEDEY"
t=""
for i in r:
    t+=chr(ord(i)+36)
print t
```

得到**flag:** `flag{kaisamima_hihihi}`

warmup-game

> hint:据说bibi师傅在无畏先锋的id就是此题flag

直接看qq好友的游戏人生，发现字符被挡住，这时候我尝试两种方法，一种猜测，差不多试了一会出来啦
另一种方法是手机下载了一个掌盟助手，通过可能认识的人就得到id



warmup-rsa

打开得到

```
n1=0x18f60afa6b9938df69338805ae7fbd5652da3ac8fa5b7b65e4755149ba3f80d071fe8845fa20ea3e57e21fb2f630e47e48
n2=0x18f60afa6b9938df69338805ae7fbd5652da3ac8fa5b7b65e4755149ba3f80d071fe8845fa20ea3e57e21fb2f630e47e48
e1=0x17e1
e2=0x43a5
c1=0xb6e66aa0d4d5ad1460482f45aab87e80a99c1ff3af605fd9cea82d76d464272f3dd2e1797e3fede64cffcd54b2a7a5e21f
c2=0x9bcbfea3c3130364bbcf352b7810df031293949ed147919dec3ecfdd48f77e9486ae811d95f8c79eb477f4424d475dc611
```

这个n1，n2明显一样，猜测是共模攻击
访问bystudent大神的博客，改变一下他的代码

```python
#coding=utf-8

import sys

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

def main():
    #n = int(raw_input("input n:"))
    #c1 = int(raw_input("input c1:"))
    #c2 = int(raw_input("input c2:"))
    #e1 = int(raw_input("input e1:"))
    #e2 = int(raw_input("input e2:"))
    n=int("18f60afa6b9938df69338805ae7fbd5652da3ac8fa5b7b65e4755149ba3f80d071fe8845fa20ea3e57e21fb2f630
    e1=int("17e1",16)
    e2=int("43a5",16)
    c1=int("b6e66aa0d4d5ad1460482f45aab87e80a99c1ff3af605fd9cea82d76d464272f3dd2e1797e3fede64cffcd54b2a
    c2=int("9bcbfea3c3130364bbcf352b7810df031293949ed147919dec3ecfdd48f77e9486ae811d95f8c79eb477f4424d4

    s = egcd(e1, e2)
    s1 = s[1]
    s2 = s[2]
    # 求模反元素
    if s1<0:
        s1 = - s1
        c1 = modinv(c1, n)
    elif s2<0:
        s2 = - s2
        c2 = modinv(c2, n)

    #m = (c1**s1)*(c2**s2)%n
    m = (pow(c1,s1,n)*pow(c2,s2,n))%n  #效率较高
    print m

if __name__ == '__main__':
    sys.setrecursionlimit(1000000) #例如这里设置为一百万
    main()
```

在这里其实出现了一个问题，就是由于循环次数过多，导致**python**默认的深度不够，会报错

> RuntimeError: maximum recursion depth exceeded in cmp

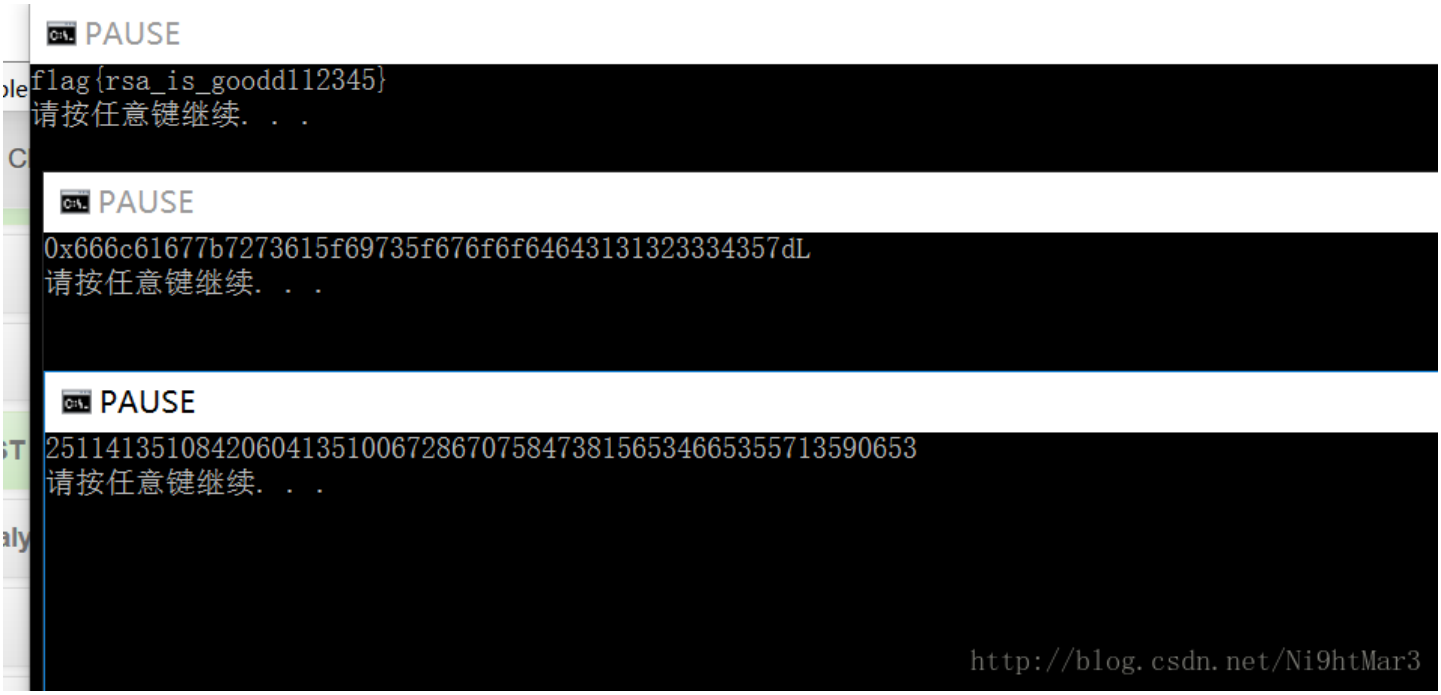
这时候百度得知改动技巧

```
import sys
sys.setrecursionlimit(1000000) #例如这里设置为一百万
```

这样就可以完美的运行，得到m

`25114135108420604135100672867075847381565346653557135906 53`

转化成字符即得**flag：** `flag{rea_is_goodd112345}`



wn

文本中给了c,e,n

```
c=0x63ef5dc677da219989328c3e02cf6fff8aa7a47929d5fff01808005d7abbf40d62fea27fe6ec2abe5d98915e1ecd65b4a9f
e=0x1dd2f1e955f2b2a230e23a691dbfa129672588f4f14f20527eb6731b1c6bec25ee955e706a2c1a3e8740900e32ffa977ac8
n=0x7d7adb3106f3ed066a0a94da1822290f596f1f6bf23f0ee64ce5804d59b82458f65ad251575777edeb75abef82fb7dc8ac8
```

看来都好大啊，度娘相关攻击手段

链接(http://www.tuicool.com/articles/iYBZBfy)

看来需要采用低解密指数攻击，github得到了相关的代码，修改一下

```
import random, sys

def rational_to_contfrac (x, y):
    '''
    Converts a rational x/y fraction into
    a list of partial quotients [a0, ..., an]
    '''
    a = x//y
    if a * y == x:
        return [a]
    else:
        pquotients = rational_to_contfrac(y, x - a * y)
        pquotients.insert(0, a)
        return pquotients
```

```
    def convergents_from_contfrac(frac):
        '''
        computes the list of convergents
        using the list of partial quotients
        '''
        convs = [];
        for i in range(len(frac)):
            convs.append(contfrac_to_rational(frac[0:i]))
        return convs

    def contfrac_to_rational (frac):
        '''Converts a finite continued fraction [a0, ..., an]
         to an x/y rational.
         '''
        if len(frac) == 0:
            return (0,1)
        elif len(frac) == 1:
            return (frac[0], 1)
        else:
            remainder = frac[1:len(frac)]
            (num, denom) = contfrac_to_rational(remainder)
            # fraction is now frac[0] + 1/(num/denom), which is
            # frac[0] + denom/num.
            return (frac[0] * num + denom, num)
    import random, sys

    def bitlength(x):
        '''
        Calculates the bitlength of x
        '''
        assert x >= 0
        n = 0
        while x > 0:
            n = n+1
            x = x>>1
        return n

    def isqrt(n):
        '''
        Calculates the integer square root
        for arbitrary large nonnegative integers
        '''
        if n < 0:
            raise ValueError('square root not defined for negative numbers')

        if n == 0:
            return 0
        a, b = divmod(bitlength(n), 2)
        x = 2**(a+b)
        while True:
            y = (x + n//x)//2
            if y >= x:
                return x
            x = y

    def is_perfect_square(n):
        '''
        If n is a perfect square it returns sqrt(n),

        otherwise returns -1
```

```
        Otherwise returns -1
        '''
        h = n & 0xF; #last hexadecimal "digit"

        if h > 9:
            return -1 # return immediately in 6 cases out of 16.

        # Take advantage of Boolean short-circuit evaluation
        if ( h != 2 and h != 3 and h != 5 and h != 6 and h != 7 and h != 8 ):
            # take square root if you must
            t = isqrt(n)
            if t*t == n:
                return t
            else:
                return -1

        return -1

    def hack_RSA(e,n):
        '''
        Finds d knowing (e,n)
        applying the Wiener continued fraction attack
        '''
        frac = rational_to_contfrac(e, n)
        convergents = convergents_from_contfrac(frac)

        for (k,d) in convergents:

            #check if d is actually the key
            if k!=0 and (e*d-1)%k == 0:
                phi = (e*d-1)//k
                s = n - phi + 1
                # check if the equation x^2 - s*x + n = 0
                # has integer roots
                discr = s*s - 4*n
                if(discr>=0):
                    t = is_perfect_square(discr)
                    if t!=-1 and (s+t)%2==0:
                        print("Hacked!")
                        return d

    if __name__ == "__main__":
        n=990023173701890142960417396557829467604818405521894936859951202214773485865832785292323872121243 8
        e=235308230249427956073994778236213308201165228968967264823632398966978083858618015310724273760982 0
        d=hack_RSA(e,n)
        print d
```

得到`d=30011`
这样直接脚本解密就行

```
import math
n=990023173701890142960417396557829467604818405521894936859951202214773485865832785292323872121243887690
e=235308230249427956073994778236213308201165228968967264823632398966978083858618015310724273760982061600
c=78847675738622154353770360889018654644288664450280369751802826792060046022001348324250602498778067327

d=30011
m=pow(c,d,n)
print m
mh=hex(m)
print mh

def str2byte(s):
    base='0123456789ABCDEF'
    i = 0
    s = s.upper()
    s1=''
    while i < len(s):
        c1=s[i]
        c2=s[i+1]
        i+=2
        b1=base.find(c1)
        b2=base.find(c2)
        if b1 == -1 or b2 == -1:
            return None
        s1+=chr((b1 << 4)+b2)
    return s1

print str2byte(mh[2:-1])
```

得到**flag**：`flag{wiener_attack_attack_you}`

```
706900059475126915342392145157176574191401939387554759734748236412319101
0x666c61677b7769656e65725f61747461636b5f61747461636b5f796f757dL
flag{wiener_attack_attack_you}
请按任意键继续. . .                    http://blog.csdn.net/Ni9htMar3
```

android

本以为安卓题很难，但看到这么多人出题，尝试一下，用**Jeb**打开，就两个类

MainActivity里存放着`gUID_P@Rt}T10n_$C#3m3Gu]d_par7|t)On_SCH3M3`

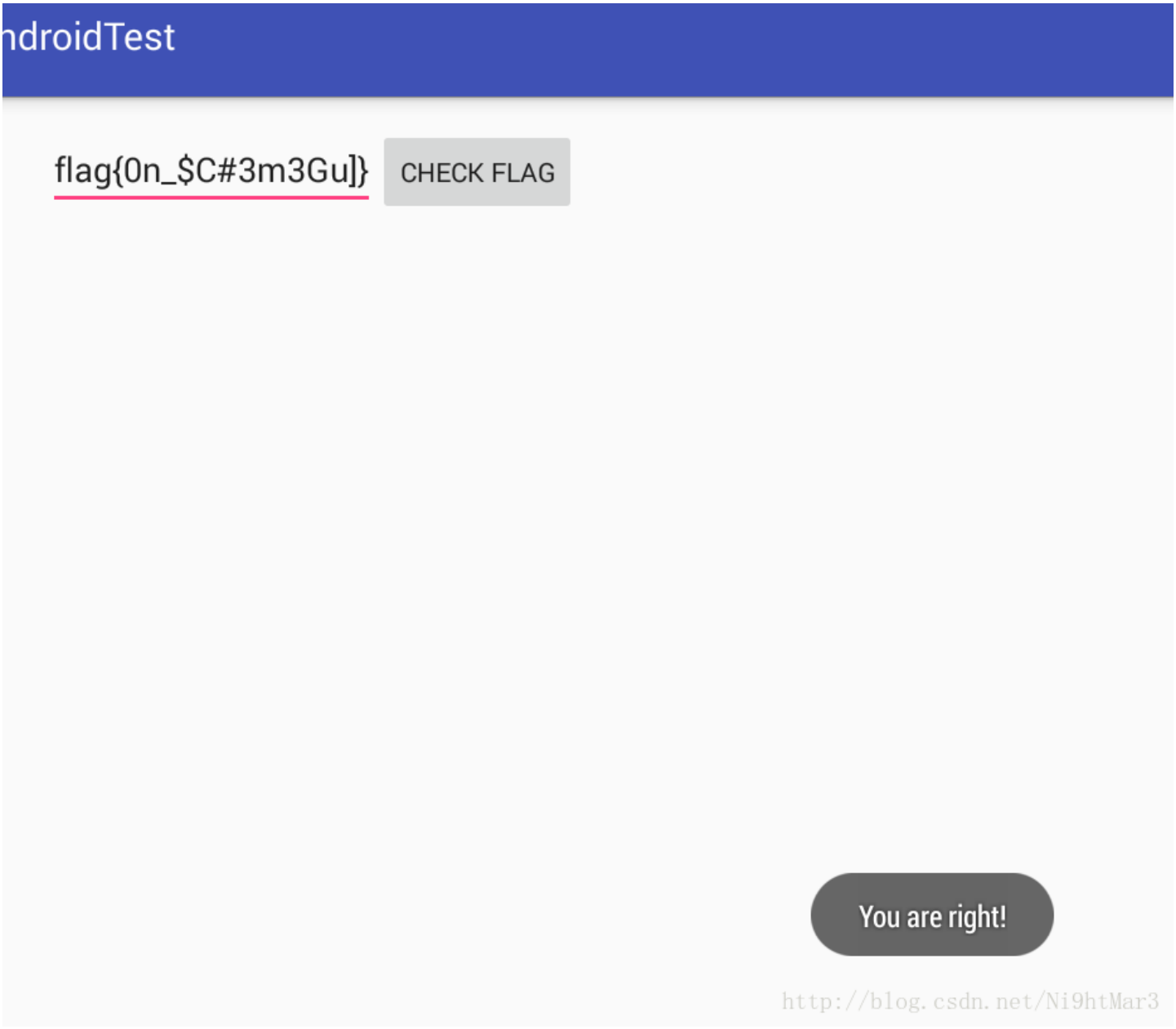
a是一个判断程序，直接反编译

```
package com.a.sample.androidtest;

import android.content.Context;
import android.view.View;
import android.view.View$OnClickListener;
import android.widget.EditText;
import android.widget.Toast;

class a implements View$OnClickListener {
    a(MainActivity arg1, EditText arg2, Context arg3) {
        this.c = arg1;
        this.a = arg2;
        this.b = arg3;
        super();
    }

    public void onClick(View arg9) {
        if(this.a.getText().toString().equals(String.format("flag{%s}", new Object[]{MainActivity.m.
                substring(12, 24)}))) {
            Toast.makeText(this.b, "You are right!", 1).show();
        }
        else {
            Toast.makeText(this.b, "You are wrong!", 1).show();
        }
    }
}
```

本以为很复杂，结果就一个字符串匹配。。。比web题简单好多，通过分析，就是要12-24区间的值

`flag{0n_$C#3m3Gu]}`

本以为这样里面乱的需要处理，结果尝试虚拟机跑一下结果对啦。。。。



web1

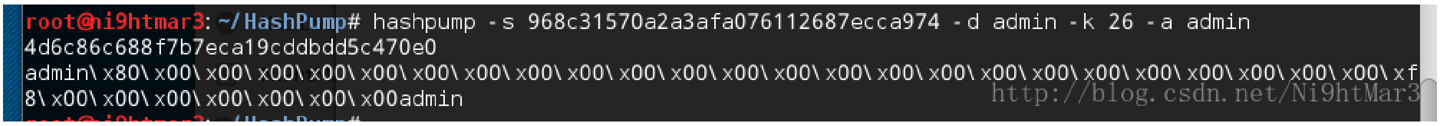
打开什么也没有，查看源码得到提示

```
<!-- index.txt -->
```

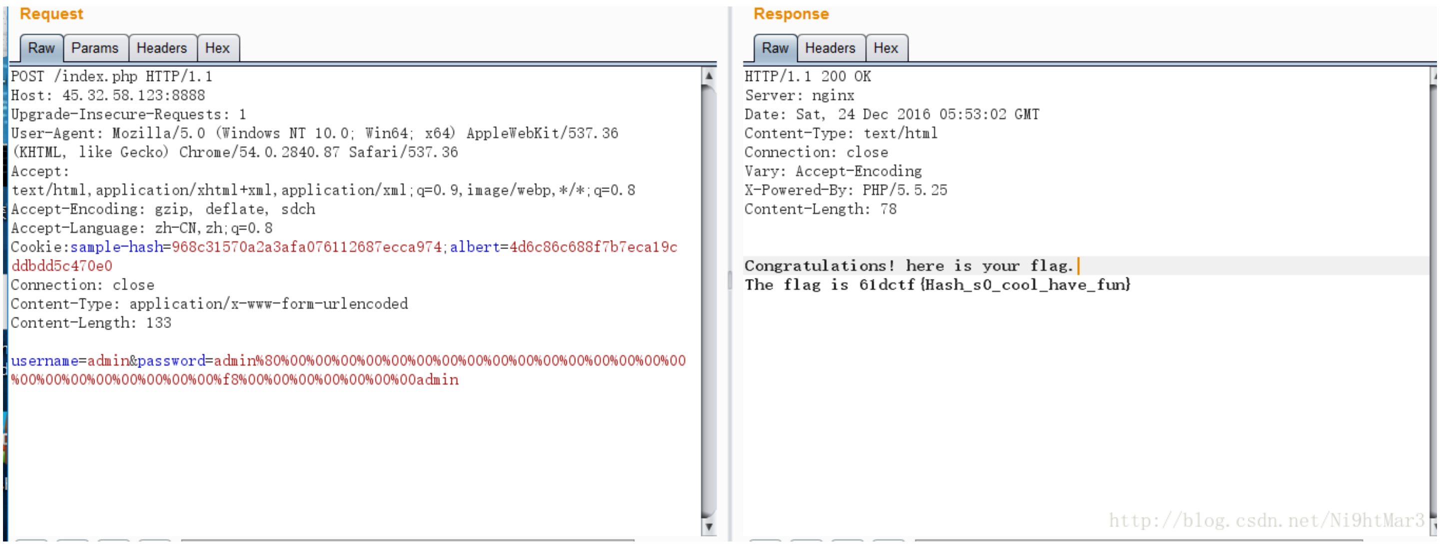
查看得到源码

```php
<?php
$flag="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
$secret = "xxxxxxxxxxxxxxxxxxxxxxxxxxx"; // guess it length :)
$username = $_POST["username"];
$password = $_POST["password"];
$cookie = $_COOKIE['albert'];
if (!empty($_COOKIE['albert'])) {
if (urldecode($username) === "admin" && urldecode($password) !=="admin") {
    if ($_COOKIE['albert'] === md5($secret . urldecode($username . $password))) {
        echo "Congratulations! here is your flag.\n";
        die ("The flag is ". $flag);
    }
    else {
        die ("Cookie Not Correct");
    }
}
else {
    die ("Go AWAY");
}
}
setcookie("sample-hash", md5($secret . urldecode("admin" . "admin")), time() + (60 * 60 * 24 * 7));
?>
```

一看明显就是哈希扩展攻击，利用**hashpump**构造即得



由于长度不知道，在这里小小的额爆破了一下20-30之间的长度，发现在26的时候成功



这期间构造POST发现点错误，发现我无论怎么改变cookie都会显示`Go AWAY`明显数据包没构造正确，后来检查发现忘记写`Content-Type: application/x-www-form-urlencode`
真是的浪费了好多时间

了解哈希长度扩展攻击

哈希长度扩展攻击适用于加密情况为：`hash($SECRET, $message)`的情况，其中 hash 最常见的就是 md5、hash1。我们可以在不知道`$SECRET`的情况下推算出另外一个匹配的值。如上例所给的 PHP 代码：

- 我们知道`md5($secret . urldecode($username . $password))`的值
- 我们知道`$hsh`的值
- 我们可以算出另外一个 md5 值和另外一个 $hsh 的值，使得 `$hsh == md5($SECRET . $_COOKIE["albert"]))`
  这样即可通过验证。

想要知道更多的可以访问下面这个github上`hash_extender`作者的文章：
[(]Everything you need to know about hash length extension attacks](https://blog.skullsecurity.org/2012/everything-you-need-to-know-about-hash-length-extension-attacks)

D0ge

这题下载下来就是一个图片，一开始使用了各种姿势就是没有分析下来，后来出了**Hint:** `bftools`
知道需要用`bftools`来进行解码，度娘查一下，发现**第六届西电信息安全大赛XDCTF2015**有道类似的题
然后两行代码直接运行解密即得一串字符串

```
C:\Users\lanlan>cd C:\Users\lanlan\Desktop\bftools

C:\Users\lanlan\Desktop\bftools>bftools.exe decode braincopter C:\Users\lanlan\Desktop\D0ge.jpg -o -out.bf

C:\Users\lanlan\Desktop\bftools>bftools.exe decode braincopter C:\Users\lanlan\Desktop\D0ge.jpg -o -out.bf

C:\Users\lanlan\Desktop\bftools>bftools.exe run -out.bf
GYYWIY3UMZ5UQML6IIYHSLCXMVWEGMDNMUWVI3ZNJAZVEZL5
C:\Users\lanlan\Desktop\bftools>                                    http://blog.csdn.net/Ni9htMar3
```

`GYYWIY3UMZ5UQML6IIYHSLCXMVWEGMDNMUWVI3ZNJAZVEZL5`
然后发现是base32解码
脚本一跑即得
`61dctf{H1~B0y,WelC0me-To-H3Re}`

CrackMe

先打开看看，出现一句话
`Give me your favourite prime number`
拖到IDA里分析一下，有四个`check`一个一个分析

先分析check1:

反编译得到代码

```
puts("Give me your favourite prime number");
  scanf("%lld", &n);
  a = 0x100000LL;
  b = 0LL;
  c = 0LL;
  while ( !(a & 1) )
  {
    a /= 2LL;
    ++b;
  }
  while ( !(b & 1) )
  {
    b /= 2LL;
    ++c;
  }
  if ( c == n )
  {
    flag += n;
    v1 = 1;
  }
  else
  {
    puts("^^^^Your math is bad^^^^");
    v1 = 0;
  }
  return v1;
}
```

很好理解，就是让输入的`n`等于`c`的值，而`c`的值是先由a的值得到b，在由b的值得到c
懒得分析，直接便代码自己跑

```
#include <iostream>

using namespace std;

main()
{
    int64_t a,b,c;
    a = 1048576;
    b = 0;
    c = 0;
    while ( !(a & 1) )
    {
        a /= 2;
        ++b;
    }
    while ( !(b & 1) )
    {
        b /= 2;
        ++c;
    }
    cout<<c;
}
```

得到`c=2`，其实这里还有`flag=c=2`先记住，后面要用

然后看check2

```
puts("Do you know how to calculate the number");
  puts("please enter a number");
  scanf("%lld", &n);
  v2 = n;
  if ( tmpans != n * tmp % mod || n > 999999 )
    goto LABEL_15;
  while ( n )
  {
    x[++pos] = n % 10;
    n /= 10LL;
  }
  if ( dword_448044 >= dword_448048
    || dword_448048 >= dword_44804C
    || dword_44804C >= dword_448050
    || dword_448050 >= dword_448054
    || dword_448054 >= dword_448058 )
  {
LABEL_15:
    puts("You do not know how to calculate the mod~~~~");
    v1 = 0;
  }
  else
  {
    flag += v2;
    v1 = 1;
  }
  return v1;
```

```
tmpans=779852816
tmp=123456
mod=1000000007
```

意思很好理解，输入n不能使得`779852816!=n*123456%1000000007`或`n>999999`
然后后面的代码意思是将n的每一位给x数组，然后要求每一位都大于后面的一位。

```
import math

tmpans=779852816
tmp=123456
mod=1000000007

for i in range(0,1000000):
    if tmpans==(tmp*i)%mod:
        print i
        x=[]
        while i:
            x.append(i%10)
            i/=10

        print x
```

这样分析分析就可以知道结果`n=654321`
`x=[1,2,3,4,5,6]`
然后`flag+=n 为654323`

然后看check3

```
int check3(void)
{
  _BYTE *v0; // eax@4
  _BYTE *v1; // eax@4
  __int64 v3; // [sp+0h] [bp-C8h]@1
  __int64 v4; // [sp+8h] [bp-C0h]@9
  __int64 v5; // [sp+18h] [bp-B0h]@9
  int v6; // [sp+34h] [bp-94h]@4
  int v7; // [sp+40h] [bp-88h]@5
  char TlsValue; // [sp+44h] [bp-84h]@1
  int v9; // [sp+48h] [bp-80h]@1
  int (__cdecl *v10)(int, int, int, int, int, unsigned __int8 *); // [sp+5Ch] [bp-6Ch]@1
  int *v11; // [sp+60h] [bp-68h]@1
  char *v12; // [sp+64h] [bp-64h]@1
  void *v13; // [sp+68h] [bp-60h]@1
  __int64 *v14; // [sp+6Ch] [bp-5Ch]@1
  int j; // [sp+78h] [bp-50h]@4
  int i; // [sp+7Ch] [bp-4Ch]@2
  int v17; // [sp+80h] [bp-48h]@1
  int v18; // [sp+90h] [bp-38h]@1
  char v19; // [sp+A0h] [bp-28h]@1
  char v20; // [sp+B0h] [bp-18h]@1

  v10 = __gxx_personality_sj0;
  v11 = dword_442E70;
  v12 = &v20;
  v13 = &loc_4019EE;
  v14 = &v3;
  _Unwind_SjLj_Register(&TlsValue);
  v9 = -1;
  std::string::string((std::string *)&v19);
  std::allocator<char>::allocator(&v17);
  v9 = 2;
  std::string::string((int)&v18, "MerryChristmas", (int)&v17);
  std::allocator<char>::~allocator(&v17);
  v9 = 1;
  puts("Do you wanna enjoy the festival tomorrow?");
  std::operator>><char,std::char_traits<char>,std::allocator<char>>((std::istream *)&std::cin, (std::st
  if ( std::string::length((std::string *)&v19) == 14 )
  {
    for ( i = 0; i <= 13; ++i )
    {
      j = i % 6 + 1;
      v9 = 1;
      v0 = (_BYTE *)std::string::operator[]((std::string *)&v19, i);
      v6 = *v0 + x[j];
      v1 = (_BYTE *)std::string::operator[]((std::string *)&v18, i);
      if ( v6 != *v1 )
      {
        v9 = 3;
        std::string::~string((std::string *)&v18);
        v9 = -1;
        std::string::~string((std::string *)&v19);
        v7 = 0;
        goto LABEL_12;
      }
    }
    for ( j = 0; j <= 13; ++j )
```

```
    {
      v9 = 1;
      LODWORD(v5) = *(_BYTE *)std::string::operator[]((std::string *)&v19, j);
      v5 = (signed int)v5;
      v4 = mod;
      v3 = flag + (signed int)v5 * tmp;
      flag = (flag + (signed int)v5 * tmp) % mod;
    }
    v9 = 3;
    std::string::~string((std::string *)&v18);
    v9 = -1;
    std::string::~string((std::string *)&v19);
    v7 = 1;
  }
  else
  {
    v9 = 3;
    std::string::~string((std::string *)&v18);
    v9 = -1;
    std::string::~string((std::string *)&v19);
    v7 = 0;
  }
```

分析这代码的意思，基本上是得到一个长度为**14**的字符串使得按位进行模6循环加与`v18=MerryChristmas`比较相等输出，懒省事写个脚本

```
s='MerryChristmas'
r=''
j=0
for i in s:
    k=j%6+1
    r+=chr(ord(i)-k)
    j=j+1
print r
```

得到结果

> v19= Lcont=gpfoog`q

然后后面的也有用

```
for ( j = 0; j <= 13; ++j )
{
v9 = 1;
LODWORD(v5) = *(_BYTE *)std::string::operator[]((std::string *)&v19, j);
v5 = (signed int)v5;
v4 = mod;
v3 = flag + (signed int)v5 * tmp;
flag = (flag + (signed int)v5 * tmp) % mod;
}
```

是一个flag的循环得到，将刚进去的v19每一位转成数值型进行计算得到结果，依旧代码

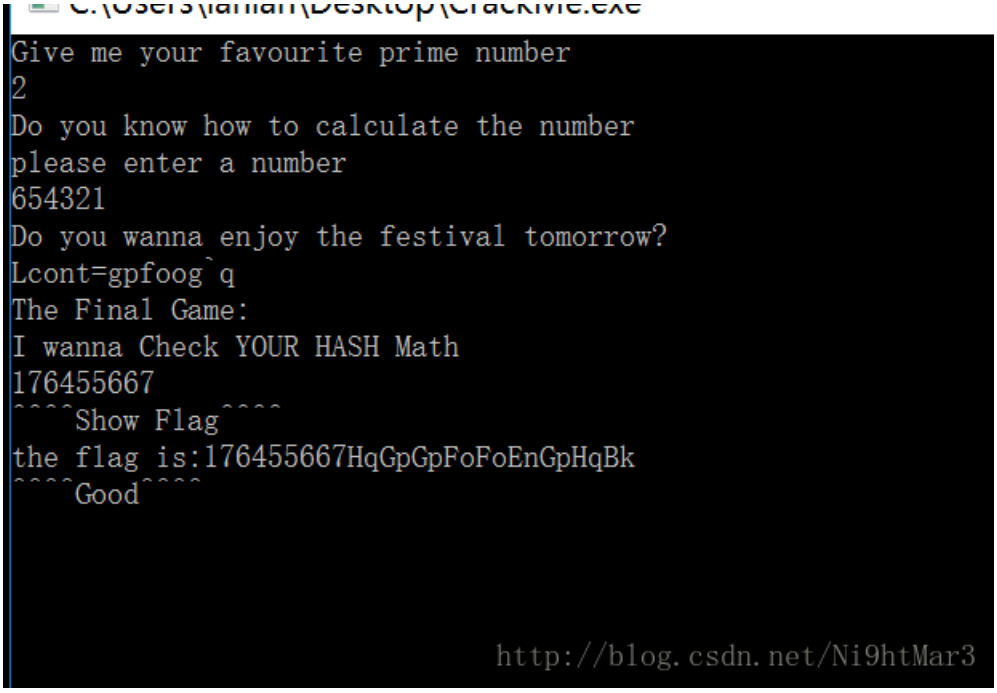
```
j=654323
st='Lcont=gpfoog`q'
mod=1000000007
tmp=123456

for i in st:
    j=(j + ord(i) * tmp)%mod

print j
```

得到结果`flag=176455667`

check4

```
signed int check4(void)
{
  signed int v1; // [sp+14h] [bp-4h]@2

  puts("The Final Game:");
  puts("I wanna Check YOUR HASH Math");
  scanf("%lld", &n);
  if ( flag == n )
  {
    puts("^^^^Show Flag^^^^");
    v1 = 1;
  }
  else
  {
    v1 = 0;
  }
  return v1;
}
```

就是输n=flag就行了
这里有一个坑。我就一直提交，结果发现一直不停，一直以为自己算错啦，结果经过提示：没停不代表不对，开始下断点，终于让他得到flag

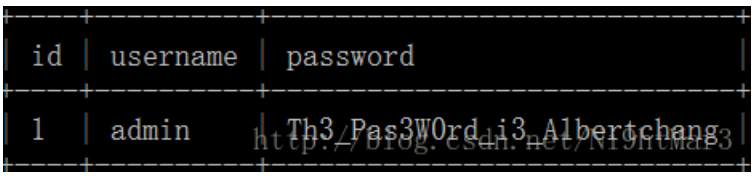


web4

打开网页居然出现张图片，查看源码

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#000000">
<div style=" margin-top:70px;color:#FFF; font-size:23px; text-align:center">Hi   <font c
<font size="3" color="#FFFF00">
id?????????????????????????????????????????</font> </div></br></br></br><center>
<img src="1.jpg" style="position:absolute; top:0px;left:0px; width:100%" /></center>
</body>
</html>
```

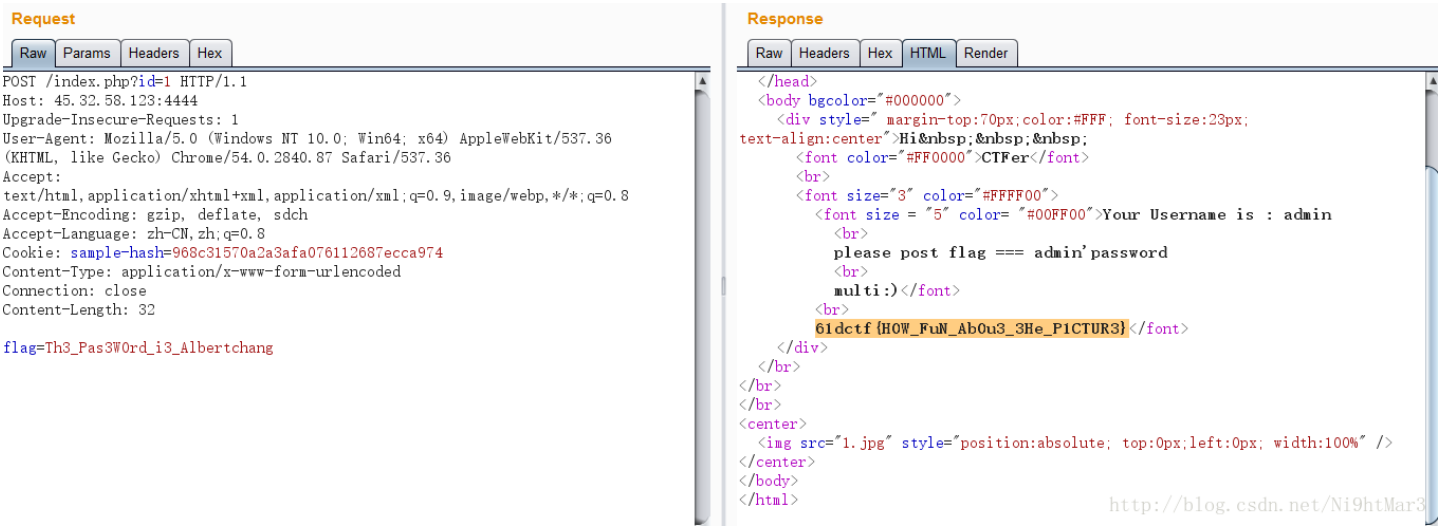
发现有一个`id? ? ? ?`莫非这是注入点？？
先尝试看看sqlmap跑跑吧，反正我懒
`sqlmap.py -u http://45.32.58.123:4444/index.php?id=1 --tamper=space2comment --dump --batch`
结果真的跑出来啦。。。



访问`http://45.32.58.123:4444/index.php?id=1`
出来提示：

> please post flag === admin'password

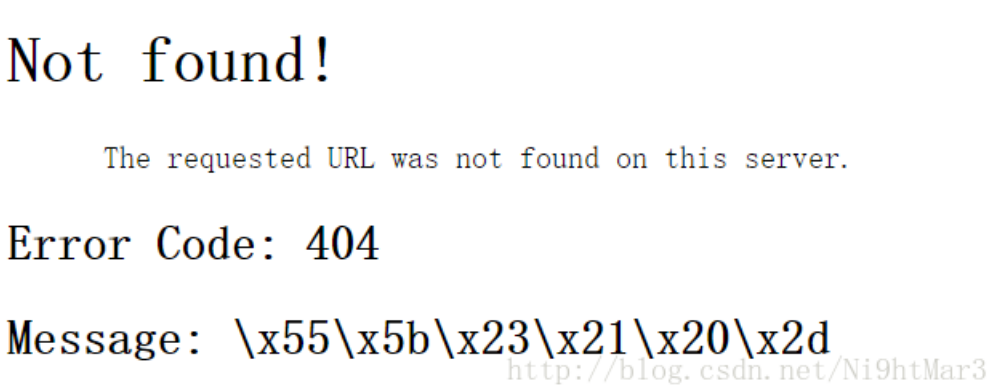
这样的话就是构造`POST`包，期间各种理解错题意尝试了好几次方式，均得不到**flag**，结果后来一直循环尝试之前的方法有一种就对啦。。。好吧



web6

打开是一个这个界面

一开始查看源代码啥的什么都没有，那么就随便输输



发现每次输Message都在改变，查看源代码，发现

```
</p>
<h2>Error Code: 404</h2>
<h2>Message: \x55\x5b\x23\x21\x20\x2d</h2>
<!-- Flag Here:H:rg5U>9B!N`I<e0S`Jsfacf[4<jjoxj~qS&{3/O
&hPPa=L1');Ba<(vk=jn;[t3;+hYuf\L*`
MNO})ie-/,`]C>Yy^]dsrXI5s@
R(C%1XNxfh>I]V&7
pd44g3/.5 '  XNxe+\e      :t}\$;y!]ScR
OZ\"[C2V+o2*QTB-4-Fk%>1(o^] p/_.sJ_7P6G<OooCNOSfyOLO=qsfhs~111F$gB7%`"ZcghxI1NY>R'v6Ub)PYsn|s Fub1<``^1$ff0_o7x7pnB
-7BI1Bwc'9iv2^ 'yvrJ`EJefMWmtg~OiYLU@~{bxdRQEU~br[MZSfdrGAo
^L_|dfebci{LN{NGBx|U}kaDJtZ5KQ%--></body>
```

下面一串类似乱码的字符提示是flag，看来flag就是这个，但这个每次提交也会改变，但截出来发现这个虽然都在改变，但长度都保证在512字节中，这是一个非常重要的一点
然后上面有一个重要的提示

> Do You Know liumima?

百度一下，这是一个关于流密码的加密题
链接(http://wenku.baidu.com/link?url=dEDeXst74HI-yaldjZ7vKQH8CrEpj_jScvanVqn2hyOu_tB5dQLMBj2O4c8Js3cVQjlbzwpQVbgR3x1AIdb1X0l0LZjBBXhZS8AJu6iNxJu)
里面有简要介绍解密原理
大意就是将当你输入的值与flag与message长度一直，且异或就能得到真正的Flag，这样的话就能构造脚本，由于这个不用交互，构造512字节的`1`,所以得到一组Flag，Message就可以
由于长度过大，所以用了点小的转换来使得三者都为10进制然后异或

Message:

```
\x77\x0e\x03\x73\x4a\x45\x1b\x45\x62\x60\x49\x18\x01\x57\x76\x51\x43\x14\x7e\x53\x17\x75\x06\x5a\x43\x4
```

去掉\x

```
770e03734a451b4562604918015776514314 7e531775065a434856411d706e45557c56441b6d5b19107d6374600d6a1075791d0
```

转化成10进制Message:

```
485700964415255903211113372964504659639279456393066557980938092352095890535932872895141800499542172674 0
```

Flag:

```
8^GE(^ c},Sy^       7lv.r   %*OML;4 FA\E0} p%OhhaPj7:S}aE6XGys(>~<?eoC$'w%>@ XJ `/<A7pS&S8!!!h[!HLdD}[m
OF\~
```

10进制的Flag：

```
217980398411617158842241975551949019674179278908268446065163864232720395592880711787731069229386783564 9
```

得到最终flag：

```
470455327039912774504782716414819151563525179727516887293855531622782540801134307473237930221874616439744548606 0
```

将flag转化成16进制

```
>>> hex(470455327039912774504782716414819151563525179727516887293855531622782540801134307473237930221874616439744548606 0
901302286604707431269004612732197308457656759455375420603218702318939545354683271587415569287437713500121732297092152771
207043948274334804679420080015777456387042834595098133798214794143307934480546239009501061499722694957754817254327248058
878710749251011291175482553163866513499358730564202362820061861736116845020171633399855846703945162444749819099261613607
997268964484566815207171575140460456913511597414200263947615614984336049177752718175406461733739801840570154848794719506
339984079500864529113920967133889738558632265706584561940478762377949677839631076012677765269818841410965743811233066942
257516174724043004345712010693092003426245749886752409664048763489731588657721808092196255234426155905568198280232001777
649027533427444705607748566102491984438426214534857660916383783919493271929164136258277002491549620961797444313641270636
531147782682227701519735763607902703612747790753656098325535775681534718707935782211816226609024003768005866364124450347
966339174223322413498454261014032815047773356691063956088245829023142308883219700097185845883821106785038206520290241090
450459991160024730090556766466844696377 57)
’0x7351574c487a6a444a65667d7d3238604f246e6d587c59633e763f636130567d292c607c7036757973663d70296f374e52446f7374232a3d32235
325755569507e3e294842255d7072796b35516b7a70447a3c2f6874365f5f5d7326434f3777446c316d697c327c7d4a235b6052463c79545c377e675
c5359464b232467686e4e59312452632e48383f457c7042767242727c4b5c69667026746157654a4c4d54496e2f3e7e6e5b4646407d6a7b4d71722c7
35331454b434d2c72353f54237b2454317e763c507b364532502d53554a5b254d574c5c48522f693234313559614 4f537a2f3262725b5259754e793
475217c212972774f522c635d7b5d4c75742c452c634b2d2a516d50446b6c7e443d5d6d447e264f58784a765f53282d3e3f4468614a2f2328546b575
e2c5b5471505e6b5c583557645e6b6c68392f7e75633847704a24655e666e387a37486879607c766e32605f7748594a415b473f2568477726714f473
e624b34634f6c71674d4824785629654f3c51705730755f77515a3837316956474e4d396c6e497a4f373d4446304636316463746747667b3736313438393
433323738393439303834353738353638373935363738393639303538363839373436383937353039383438393730333234383733313234373839313
233343738393132333734383738393034343733383935 5f626f795f3f24255e327dL’
```

然后转化成字符即得

```
PAUSE
sQWLHzjDJef}}28`O$nmX|Yc>v?ca0V}),`|p6uysf=p)o7NRDost#*=2#S%uUiP`>)HB%]pryk5QkzpDz</ht6__]s&CO7wD1lmi|2|}J#[`RF<yT\7`g\S
YFK#$ghnNY1$Rc.H8?E|pBvrBr|K\ifp&taWeJLMTIn/>`n[FF@}j{Mqr,sS1EKCM,r5?T#{$T1`v<P{6E2P-SUJ[%MWL\HR/i2415YaDOSz/2br[RYuNy4u
!||!)rwOR,c]{]Lut,E,cK-*QmPDk1`D=]mD`&OXxJv_S(->?DhaJ/#(TkW^,[TqP^k\X5Wd^k1h9/`uc8GpJ$e^fn8z7Hhy`|vn2`_wHYJA[G?%hGw&qOG>b
K4cOlqgMH$xV)eO<QpWOu_wQZ871iVGNM91nIzO7=DFOF61dctf{7614894327894908457856879567896905868974689750984897032487312478912 3
4789123748789044738 95_boy_?$%^2}
请按任意键继续. . .
```

脚本

```
    def str2byte(s):
        base='0123456789ABCDEF'
        i = 0
        s = s.upper()
        s1=''
        while i < len(s):
            c1=s[i]
            c2=s[i+1]
            i+=2
            b1=base.find(c1)
            b2=base.find(c2)
            if b1 == -1 or b2 == -1:
                return None
            s1+=chr((b1 << 4)+b2)
        return s1

    s ='7351574c487a6a444a65667d7d3238604f246e6d587c59633e763f636130567d292c607c7036757973663d70296f374e524
    print str2byte(s)
```

flag GET！！！~

`61dctf{7614894327894908457856879567896905868974689750984897032487312478912347891237487890447389 5_boy_?$%^2}`

异或脚本如下:

```
#Flag 为所截取到的512字节字符
#Message 为处理后所得到的10进制
Message=
Flag=
s=''
for i in Flag:
    k = str(bin(ord(i)))[2:]
    if len(k) < 8:
        k = '0'*(8-len(k))+k
    s += k
Flag = int(s,2)

#print Flag

in_ = '111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111

s2 = ''
for i in in_:
    m = str(bin(ord(i)))[2:]
    if len(m) < 8:
        m = '0'*(8-len(m))+m
    s2 += m
in_ = int(s2,2)
#print in_
#in_ = 200686490938213618932925030686723900966346071385575706818931139925151927414142413276833387391430

flag = Flag^Message^in_
print flag
```