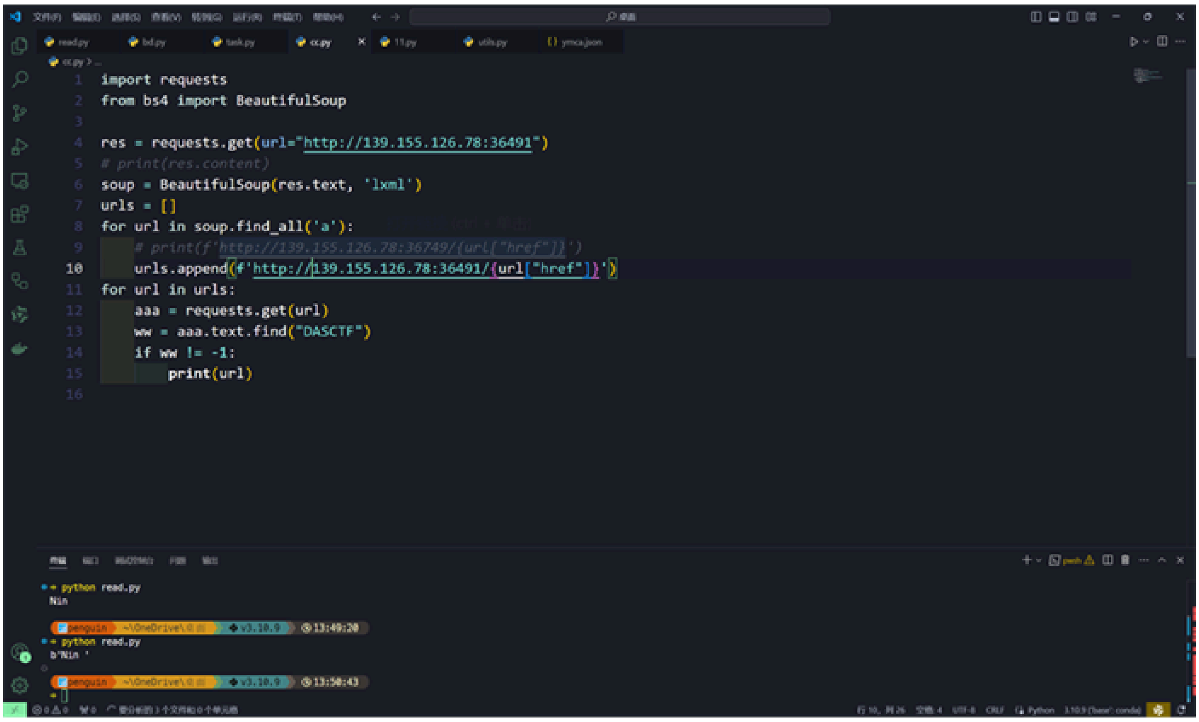
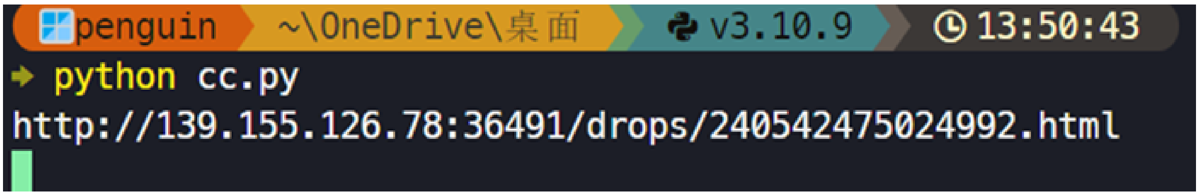
# Crawl

根据提示编写爬虫，得到含有DASCTF的网页



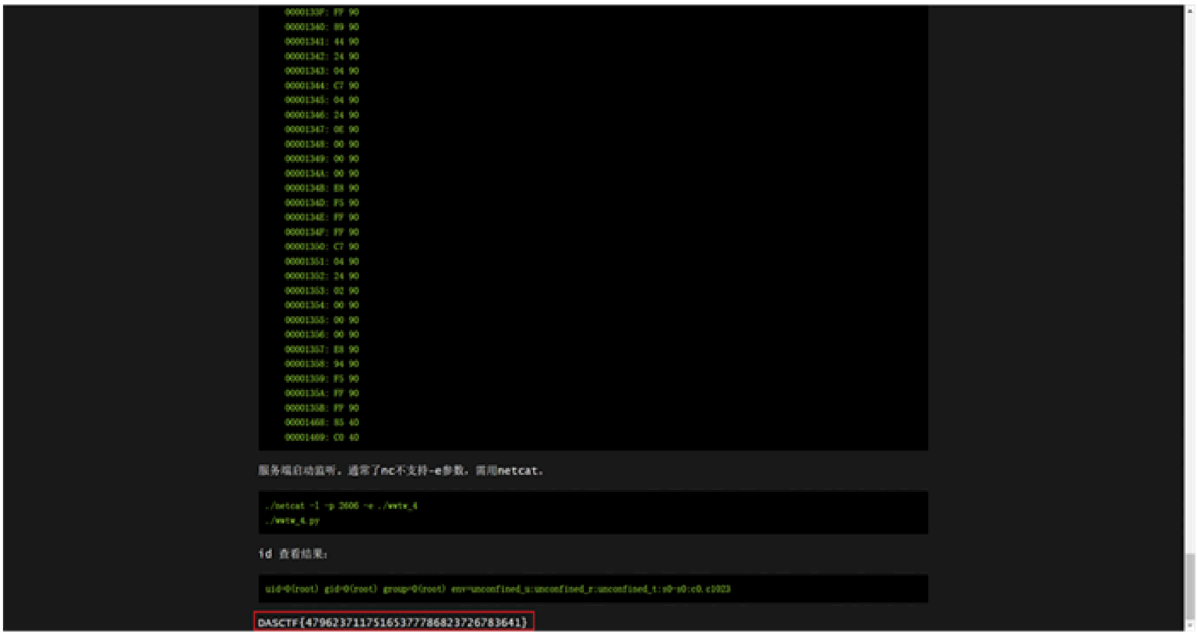
```python
import requests
from bs4 import BeautifulSoup

res = requests.get(url="http://139.155.126.78:36491")
# print(res.content)
soup = BeautifulSoup(res.text, 'lxml')
urls = []
for url in soup.find_all('a'):
    # print(f'http://139.155.126.78:36749/{url["href"]}')
    urls.append(f'http://139.155.126.78:36491/{url["href"]}')
for url in urls:
    aaa = requests.get(url)
    ww = aaa.text.find("DASCTF")
    if ww != -1:
        print(url)
```

```
→ python read.py
Nin

 penguin  ~\OneDrive\桌面  v3.10.9  13:49:20
→ python read.py
b'Nin '

 penguin  ~\OneDrive\桌面  v3.10.9  13:50:43
→
```



```
 penguin  ~\OneDrive\桌面  v3.10.9  13:50:43
→ python cc.py
http://139.155.126.78:36491/drops/240542475024992.html
```

得到flag



```
0000133F: FF 90
00001340: 89 90
00001341: 44 90
00001342: 24 90
00001343: 04 90
00001344: C7 90
00001345: 04 90
00001346: 24 90
00001347: 0E 90
00001348: 00 90
00001349: 00 90
0000134A: 00 90
0000134B: E8 90
0000134D: F5 90
0000134E: FF 90
0000134F: FF 90
00001350: C7 90
00001351: 04 90
00001352: 24 90
00001353: 02 90
00001354: 00 90
00001355: 00 90
00001356: 00 90
00001357: E8 90
00001358: 94 90
00001359: F5 90
0000135A: FF 90
0000135B: FF 90
00001468: 85 40
00001469: C0 40
```

服务端启动监听。通常了nc不支持-e参数，需用netcat。

```
./netcat -l -p 2606 -e ./wwtw_4
./wwtw_4.py
```

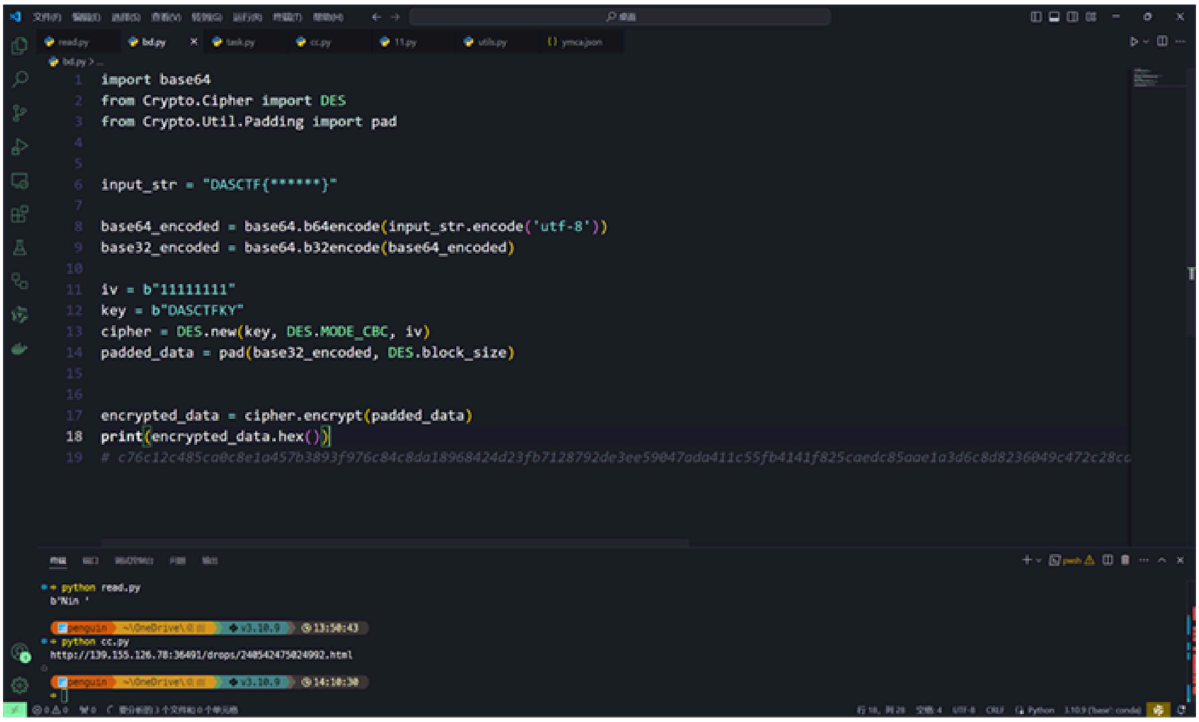
id 查看结果：

```
uid=0(root) gid=0(root) group=0(root) env=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

DASCTF{47962371175165377786823726783641}

# BD

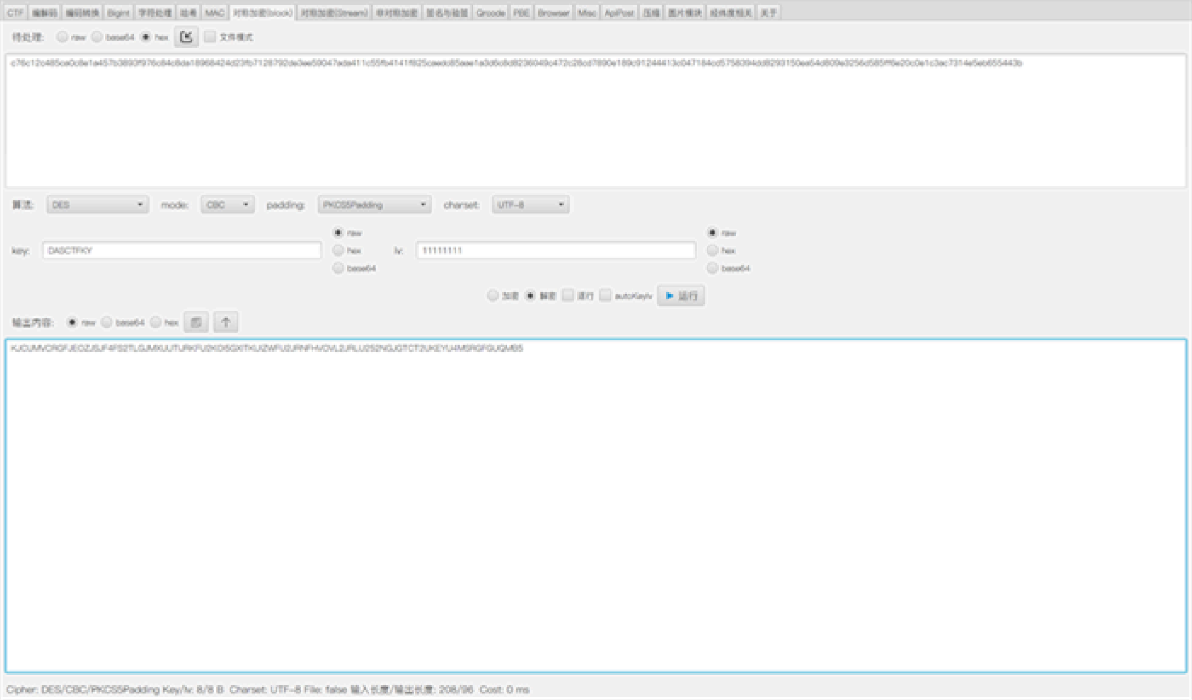
根据代码



```python
import base64
from Crypto.Cipher import DES
from Crypto.Util.Padding import pad


input_str = "DASCTF{******}"

base64_encoded = base64.b64encode(input_str.encode('utf-8'))
base32_encoded = base64.b32encode(base64_encoded)

iv = b"11111111"
key = b"DASCTFKY"
cipher = DES.new(key, DES.MODE_CBC, iv)
padded_data = pad(base32_encoded, DES.block_size)


encrypted_data = cipher.encrypt(padded_data)
print(encrypted_data.hex())
# c76c12c485ca0c8e1a457b3893f976c84c8da18968424d23fb7128792de3ee59047ada411c55fb4141f825caedc85aae1a3d6c8d8236049c472c28cc
```

```
python read.py
b'Nin '

penguin  ~\OneDrive\  v3.10.9  13:50:43
python cc.py
http://139.155.126.78:36491/drops/240542475024992.html

penguin  ~\OneDrive\  v3.10.9  14:10:30
```

使用工具得到原文



待处理: ○ raw ○ base64 ● hex

```
c76c12c485ca0c8e1a457b3893f976c84c8da18968424d23fb7128792de3ee59047ada411c55fb4141f825caedc85aae1a3d6c8d8236049c472c28cd7890e189c91244413c047184cd5758394dd8293150ea54d809e3256d585ff6e20c0e1c3ac7314e5eb655443b
```

算法: DES　mode: CBC　padding: PKCS5Padding　charset: UTF-8

key: DASCTFKY　● raw ○ hex ○ base64

Iv: 11111111　● raw ○ hex ○ base64

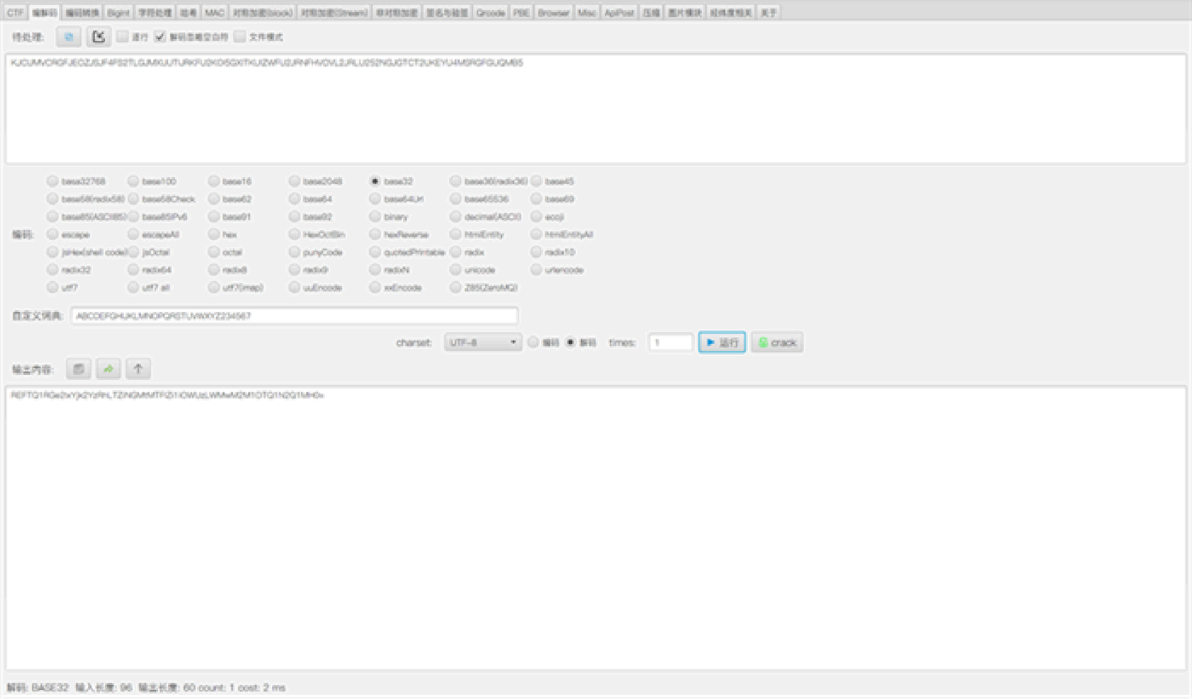
○ 加密 ● 解密

输出内容: ● raw ○ base64 ○ hex

```
KJCUMVCRGFJEOZJSJF4FS2TLGJMXUUTURKFU2KDISGXITKUZWFU2JRNFHVOVL2JRLU2S2NGJGTCT2UKEYU4MSRGFGUQMB5
```

Cipher: DES/CBC/PKCS5Padding Key/Iv: 8/8 B Charset: UTF-8 File: false 输入长度/输出长度: 208/96 Cost: 0 ms

Base32解密



待处理:

KJCUMVCRGFJEOZJSJF4FS2TLGJMXUUTIJRKFU2KOI5GXITKUIZWFU2JRNFHVOVL2JRLU252NGJGTCT2UKEYU4MSRGFGVEDA5

编码: ● base32

自定义词典: ABCDEFGHIJKLMNOPQRSTUVWXYZ234567

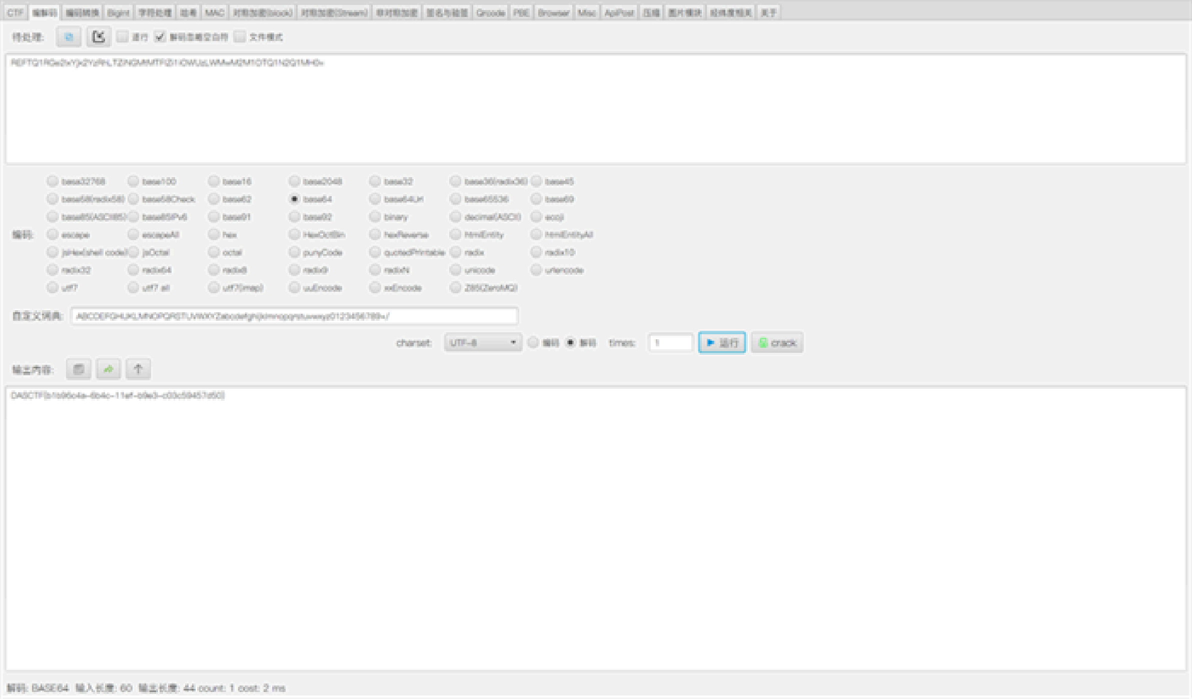
charset: UTF-8 ○ 编码 ● 解码 times: 1

输出内容:

REFTQ1RGe2IxYjk2YzRhLTZiNGMtMTFlZi1iOWUzLWMwM2M1OTQ1N2Q1MH0=

解码: BASE32 输入长度: 96 输出长度: 60 count: 1 cost: 2 ms

Base64解密得到flag



待处理:

REFTQ1RGe2IxYjk2YzRhLTZiNGMtMTFlZi1iOWUzLWMwM2M1OTQ1N2Q1MH0=

编码: ● base64

自定义词典: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/

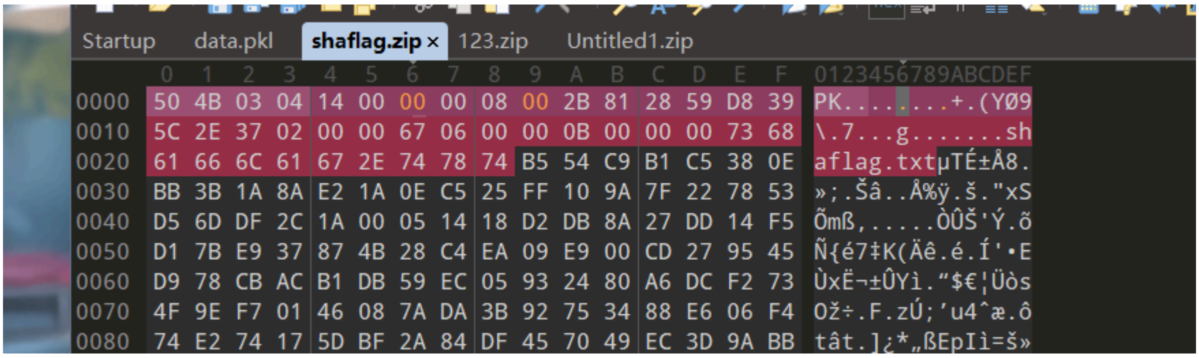
charset: UTF-8 ○ 编码 ● 解码 times: 1

输出内容:

DASCTF{b1b96c4a-6b4c-11ef-b9e3-c03c59457d50}

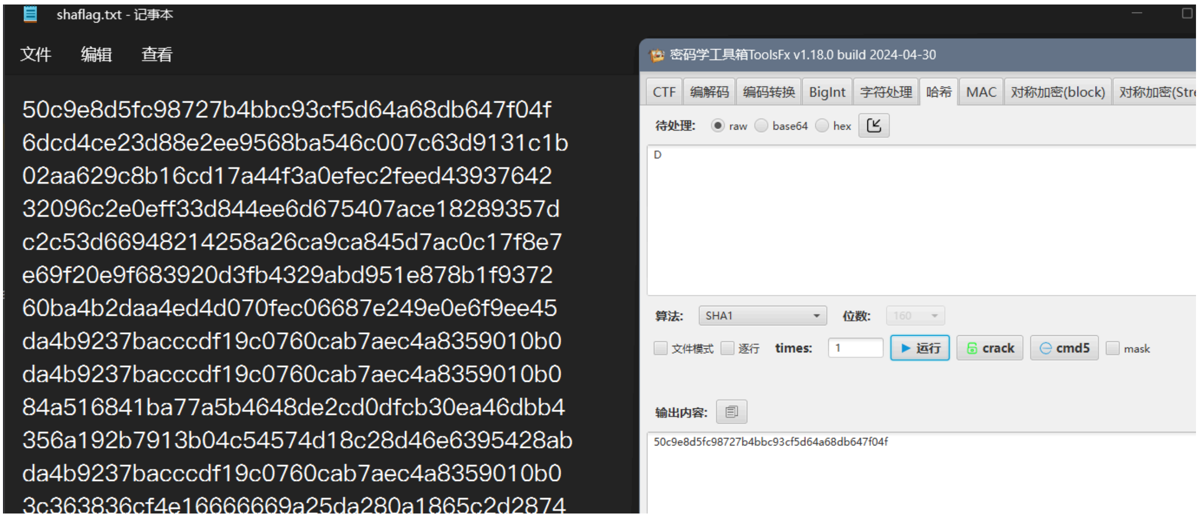
解码: BASE64 输入长度: 60 输出长度: 44 count: 1 cost: 2 ms

# what_flag



```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000 50 4B 03 04 14 00 00 00 08 00 2B 81 28 59 D8 39  PK........+.(YØ9
0010 5C 2E 37 02 00 00 67 06 00 00 0B 00 00 00 73 68  \.7...g.......sh
0020 61 66 6C 61 67 2E 74 78 74 B5 54 C9 B1 C5 38 0E  aflag.txtµTÉ±Å8.
0030 BB 3B 1A 8A E2 1A 0E C5 25 FF 10 9A 7F 22 78 53  »;.Šâ..Å%ÿ.š."xS
0040 D5 6D DF 2C 1A 00 05 14 18 D2 DB 8A 27 DD 14 F5  Õmß,.....ÒÛŠ'Ý.õ
0050 D1 7B E9 37 87 4B 28 C4 EA 09 E9 00 CD 27 95 45  Ñ{é7‡K(Äê.é.Í'•E
0060 D9 78 CB AC B1 DB 59 EC 05 93 24 80 A6 DC F2 73  ÙxË¬±ÛYì.“$€¦Üòs
0070 4F 9E F7 01 46 08 7A DA 3B 92 75 34 88 E6 06 F4  Ož÷.F.zÚ;’u4ˆæ.ô
0080 74 E2 74 17 5D BF 2A 84 DF 45 70 49 EC 3D 9A BB  tât.]¿*„ßEpIì=š»
```

猜测是伪加密 修改文件头 可以打开shaflag 根据文件名猜测为sha加密



```
50c9e8d5fc98727b4bbc93cf5d64a68db647f04f
6dcd4ce23d88e2ee9568ba546c007c63d9131c1b
02aa629c8b16cd17a44f3a0efec2feed43937642
32096c2e0eff33d844ee6d675407ace18289357d
c2c53d66948214258a26ca9ca845d7ac0c17f8e7
e69f20e9f683920d3fb4329abd951e878b1f9372
60ba4b2daa4ed4d070fec06687e249e0e6f9ee45
da4b9237bacccdf19c0760cab7aec4a8359010b0
da4b9237bacccdf19c0760cab7aec4a8359010b0
84a516841ba77a5b4648de2cd0dfcb30ea46dbb4
356a192b7913b04c54574d18c28d46e6395428ab
da4b9237bacccdf19c0760cab7aec4a8359010b0
3c363836cf4e16666669a25da280a1865c2d2874
```

待处理: ● raw ○ base64 ○ hex

D

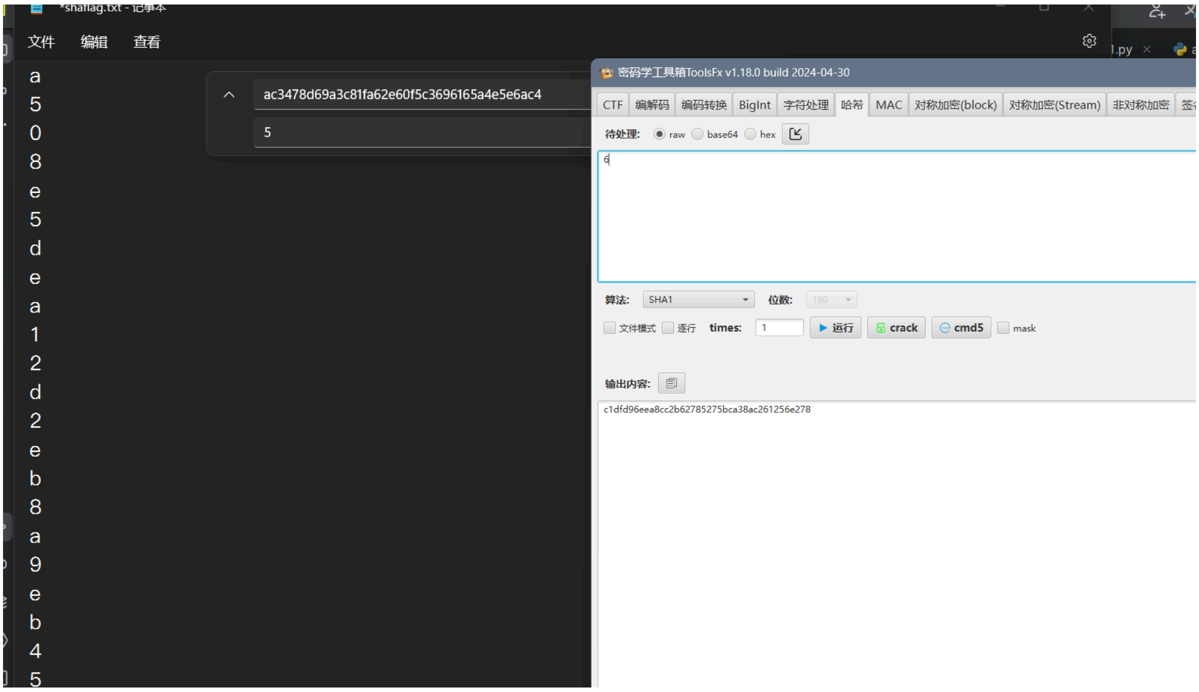
算法: SHA1 位数: 160

□ 文件模式 □ 逐行 times: 1 ▶ 运行 □ mask

输出内容:

50c9e8d5fc98727b4bbc93cf5d64a68db647f04f

逐个尝试发现前六行为DASCTF 后面内容即为flag 手工进行单字符测试 爆破



```
a
5
0
8
e
5
d
e
a
1
2
d
2
e
b
8
a
9
e
b
4
5
```

ac3478d69a3c81fa62e60f5c3696165a4e5e6ac4

5

待处理: ● raw ○ base64 ○ hex

6

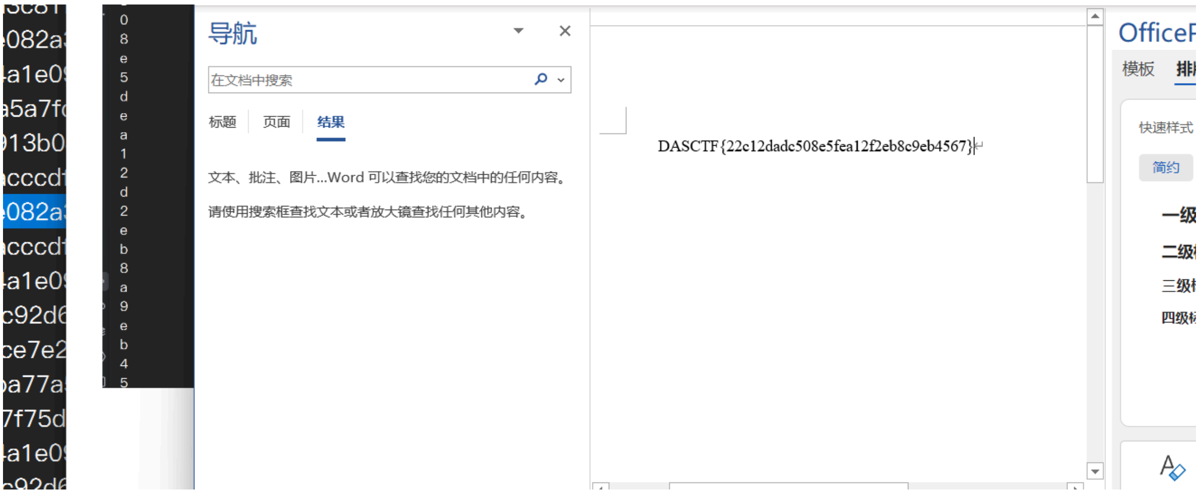
算法: SHA1 位数: 160

□ 文件模式 □ 逐行 times: 1 ▶ 运行 □ mask

输出内容:

c1dfd96eea8cc2b62785275bca38ac261256e278

经过不断试错 得出



在文档中搜索

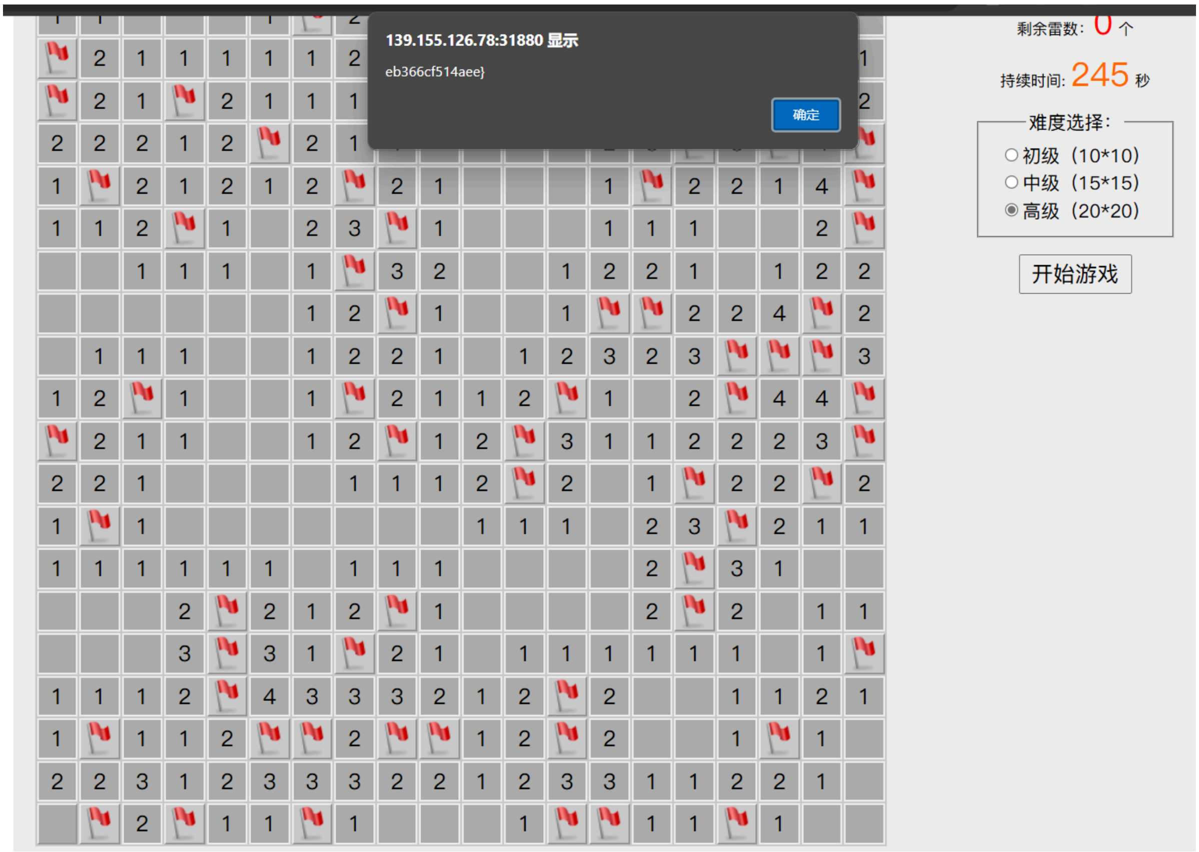
文本、批注、图片...Word 可以查找您的文档中的任何内容。

请使用搜索框查找文本或者放大镜查找任何其他内容。

DASCTF{22c12dadc508e5fea12f2eb8c9eb4567}

# Minesweepe

破解组和手工组同时进行 手工组做出来了 我都能手搓出来高级难度 前两个忘记截图也可以忽略吧)



139.155.126.78:31880 显示

eb366cf514aee}

剩余雷数: 0 个

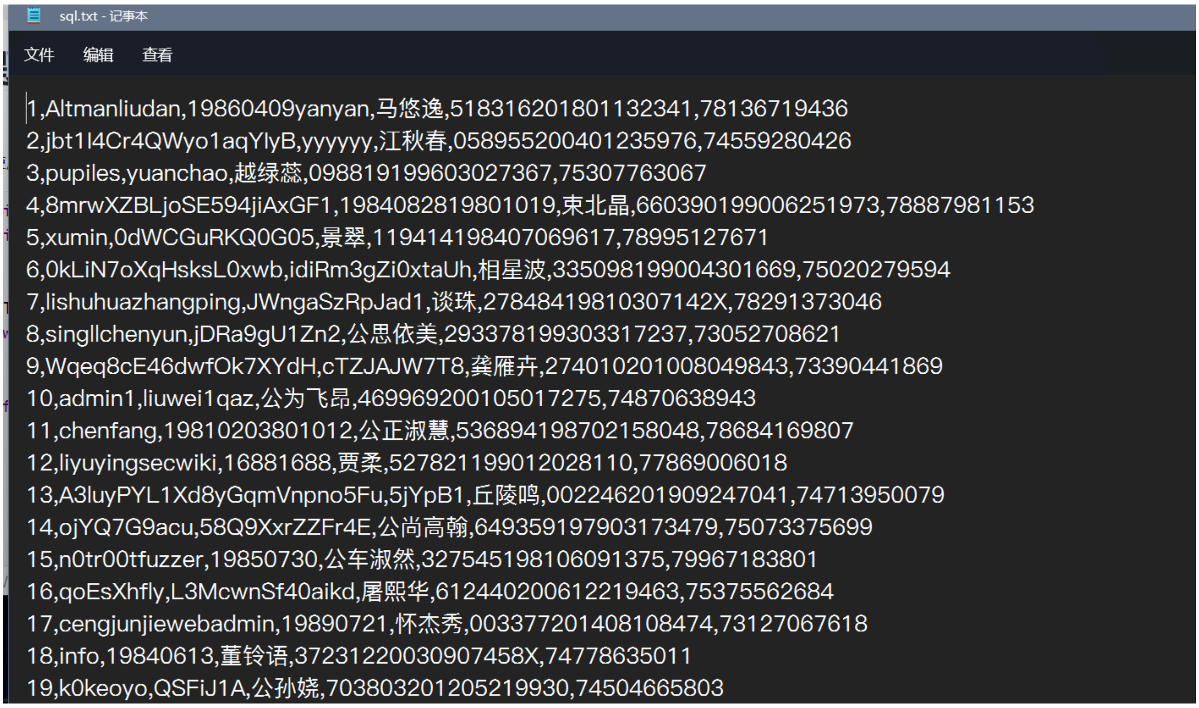
持续时间: 245 秒

难度选择:

○ 初级（10*10）
○ 中级（15*15）
● 高级（20*20）

# data-tuo

通过替换功能 将数据处理为如下图格式



```
1,Altmanliudan,19860409yanyan,马悠逸,518316201801132341,78136719436
2,jbt1l4Cr4QWyo1aqYlyB,yyyyyy,江秋春,058955200401235976,74559280426
3,pupiles,yuanchao,越绿蕊,098819199603027367,75307763067
4,8mrwXZBLjoSE594jiAxGF1,1984082819801019,束北晶,660390199006251973,78887981153
5,xumin,0dWCGuRKQ0G05,景翠,119414198407069617,78995127671
6,0kLiN7oXqHsksL0xwb,idiRm3gZi0xtaUh,相星波,335098199004301669,75020279594
7,lishuhuazhangping,JWngaSzRpJad1,谈珠,27848419810307142X,78291373046
8,singllchenyun,jDRa9gU1Zn2,公思依美,293378199303317237,73052708621
9,Wqeq8cE46dwfOk7XYdH,cTZJAJW7T8,龚雁卉,274010201008049843,73390441869
10,admin1,liuwei1qaz,公为飞昂,469969200105017275,74870638943
11,chenfang,19810203801012,公正淑慧,536894198702158048,78684169807
12,liyuyingsecwiki,16881688,贾柔,527821199012028110,77869006018
13,A3luyPYL1Xd8yGqmVnpno5Fu,5jYpB1,丘陵鸣,002246201909247041,74713950079
14,ojYQ7G9acu,58Q9XxrZZFr4E,公尚高翰,649359197903173479,75073375699
15,n0tr00tfuzzer,19850730,公车淑然,327545198106091375,79967183801
16,qoEsXhfly,L3McwnSf40aikd,屠熙华,612440200612219463,75375562684
17,cengjunjiewebadmin,19890721,怀杰秀,003377201408108474,73127067618
18,info,19840613,董铃语,37231220030907458X,74778635011
19,k0keoyo,QSFiJ1A,公孙娆,703803201205219930,74504665803
```

使用如下脚本对数据进行处理

```
import hashlib
import csv


list1 = []
with open('sql_2.txt', 'r', encoding='utf-8') as f:
    for line in f.readlines():
        list1.append(line.split(','))
for rec in list1:
    # num = rec[0] 无需脱敏

    # user 脱敏
    user = rec[1]
    # print(len(user))
    if len(user) == 2:
        user_cp = user[0] + '*'
        rec[1] = user_cp
    else:
        user_cp = user[0] + (len(user)-2) * '*' + user[-1]
        rec[1] = user_cp

    # passwd 脱敏
    passwd = rec[2]
    passwd = passwd.encode('utf-8')
    rec[2] = hashlib.md5(passwd).hexdigest()

    # name 脱敏
    name = rec[3]
    # print(len(user))
    if len(name) == 2:
        name_cp = name[0] + '*'
        rec[3] = name_cp
    else:
        name_cp = name[0] + (len(name) - 2) * '*' + name[-1]
        rec[3] = name_cp

    # id 脱敏
    id = rec[4]
    new_id = '*' * 6 + id[6:10] + '*' * 8
    rec[4] = new_id

    # phone 脱敏
    phone = rec[5]
    new_phone = phone[0:3] + '*' * 4 + phone[7:-1]
    rec[5] = new_phone

with open('output.csv','w') as f2:
    wt = csv.writer(f2, lineterminator='\n')
    wt.writerows(list1)
```

提交后发现没有进行utf-8编码 在excel软件中另存为 csv utf-8格式 成功得出flag