SWPU新生赛2021 Reverse部分WriteUp

原创

是Mumuzi 于 2021-10-11 18:14:24 发布 528 收藏 2

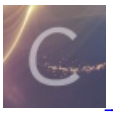
分类专栏： ctf NSSCTF 文章标签： 网络安全

本文链接：https://blog.csdn.net/qq_42880719/article/details/120581571

ctf 同时被 2 个专栏收录

75 篇文章 28 订阅

订阅专栏



NSSCTF

6 篇文章 1 订阅

订阅专栏

我真没想到第一波我做re比misc多。。

第一波放题

简单的逻辑

这个题缺条件，结果太多，初步晃一眼，如果不给个大概范围，能输出的结果应该是几万~几十万。当然因为做了其他的题，了解到“简单”系列基本都是大写，于是才做出来的。

题目如下

```
flag = 'xxxxxxxxxxxxxxxxx'
flag = flag[::-1]
result = 0
for i in range(0,len(flag)-1):
    s1 = ord(flag[i])
    s2 = ord(flag[i+1])
    if i == 0:
        result = (s1<<8)^(s2<<4)^s2
    else:
        result = (result<<4)^((s1<<8)^(s2<<4)^s2)
print(result)
# result = 591620785604527668617886
```

自己推，结论是

```
f[0][0:4] + f[0][4:8] + f[1][0:4] + f[1][4:8] ^ f[2][0:4] + f[2][4:8]^f[3][0:4] +f[3][4:8]^f[4][0:4] +f[4][4:8]^
f[5][0:4]+....+f[13][4:8]^f[14][0:4]+f[14][4:8]^f[15][0:4]+f[15][4:8]^f[16][0:4]^f[17][0:4]+f[16][4:8]^f[17][0:4
]^f[17][4:8]+f[17][4:8]
```

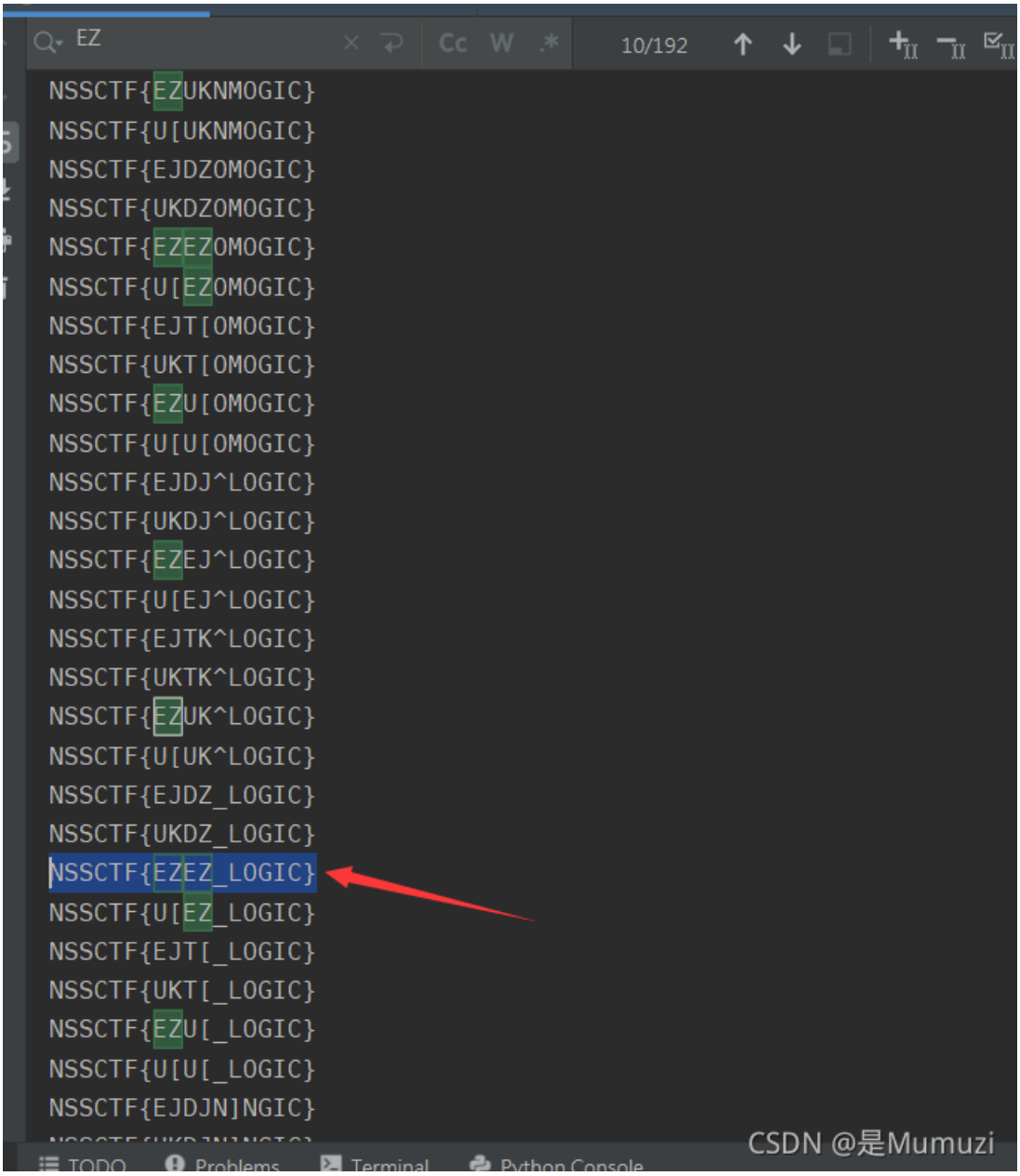
于是想这爆破，但是脚本写的很烂

然后本来是爆挺多的，结果输出的结果也太多了。。。

于是最后就爆破ascii在64~95

```python
s = str(bin(591620785604527668617886))[2:]
c = []
s = s[::-1]
for i in range(len(s)//4+1):
    c.append(s[i*4:i*4+4][::-1])
print(c[::-1])

from tqdm import tqdm
#table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ_'
hide = ['0111', '1101', '0011', '1011', '1001', '1010', '1110', '0000', '1110', '0010']
table = ['0100','0101','0110','0111','0011','0010']
for i in range(len(table)):
    table[i] = int(table[i], 2)
for counts in range(len(hide)):
    hide[counts] = int(hide[counts], 2)

trys = 2

for tmp1 in range(trys):
    for tmp2 in tqdm(range(trys)):
        for tmp3 in range(trys):
            for tmp4 in range(trys):
                for tmp5 in range(trys):
                    for tmp6 in range(trys):
                        for tmp7 in range(trys):
                            for tmp8 in range(trys):
                                for tmp9 in range(trys):
                                        tmp = ''
                                        tmp += chr((4 << 4) ^ (int(hide[0] ^ table[tmp1])))
                                        tmp += chr((table[tmp1] << 4) ^ (int(hide[1] ^ table[tmp2])))
                                        tmp += chr((table[tmp2] << 4) ^ (int(hide[2] ^ table[tmp3])))
                                        tmp += chr((table[tmp3] << 4) ^ (int(hide[3] ^ table[tmp4])))
                                        tmp += chr((table[tmp4] << 4) ^ (int(hide[4] ^ table[tmp5])))
                                        tmp += chr((table[tmp5] << 4) ^ (int(hide[5] ^ table[tmp6])))
                                        tmp += chr((table[tmp6] << 4) ^ (int(hide[6] ^ table[tmp7])))
                                        tmp += chr((table[tmp7] << 4) ^ (int(hide[7] ^ table[tmp8])))
                                        tmp += chr((table[tmp8] << 4) ^ (int(hide[8] ^ table[tmp9])))
                                        tmp += chr((table[tmp9] << 4) ^ (int(hide[9] ^ table[3])))
                                        flag = '}' + tmp + '{FTCSSN'
                                        result = 0
                                        for i in range(0,len(flag)-1):
                                            s1 = ord(flag[i])
                                            s2 = ord(flag[i+1])
                                            if i == 0:
                                                result = (s1<<8)^(s2<<4)^s2
                                            else:
                                                result = (result<<4)^((s1<<8)^(s2<<4)^s2)
                                        if(result == 591620785604527668617886):
                                            print(flag[::-1])
```

输出挺多的
但是结合后面的几个"简单"系列都有EZ在里面,正好发现这里开头有EZ，于是手动一个个测试



> NSSCTF{EZEZ_LOGIC}

简简单单的逻辑

爆破就行了，不去动脑逆，主要是懒

```
list = [47, 138, 127, 57, 117, 188, 51, 143, 17, 84, 42, 135, 76, 105, 28, 169, 25]
results='bcfba4d0038d48bd4b00f82796d393dfec'
flag = ''
for i in range(len(list)):
    for j in range(32,128):
        key = (list[i]>>4)+((list[i] & 0xf)<<4)
        result = str(hex(j^key))[2:].zfill(2)
        if(result == results[i*2:i*2+2]):
            flag += chr(j)
            break
print(flag)
```

> NSSCTF{EZEZ_RERE}

简简单单的解密

我真不会re，真不知道是RC4，我还是爆破的，爆破在这里是真神

```
import base64,urllib.parse
key = "HereIsFlagggg"
flag = ''
enc = "%C2%A6n%C2%87Y%1Ag%3F%C2%A01.%C2%9C%C3%B7%C3%8A%02%C3%80%C2%92W%C3%8C%C3%BA"
enc = urllib.parse.unquote(enc)
s_box = list(range(256))
j = 0
for i in range(256):
    j = (j + s_box[i] + ord(key[i % len(key)])) % 256
    s_box[i], s_box[j] = s_box[j], s_box[i]
res = []
i = j = 0
for s in range(len(enc)):
    i = (i + 1) % 256
    j = (j + s_box[i]) % 256
    s_box[i], s_box[j] = s_box[j], s_box[i]
    t = (s_box[i] + s_box[j]) % 256
    k = s_box[t]
    for ff in range(32,128):
        if(chr(ff ^ k) == enc[s]):
            print(chr(ff),end='')
```

> NSSCTF{REAL_EZ_RC4}

非常简单的逻辑题

还是爆破，反正每个位不会影响到其他的位，爆，都可以爆

```
s = 'wesyvbniazxchjko1973652048@$+-&*<>'
results = 'v0b9n1nkajz@j0c4jjo3oi1h1i937b395i5y5e0e$i'
flag = ''
for i in range(21):
    for j in range(32,128):
        s1 = j//17
        s2 = j%17
        result = s[(s1+i)%34]+s[-(s2+i+1)%34]
        if(results[i*2:i*2+2] == result):
            flag += chr(j)
            break
print(flag)
```

> NSSCTF{Fake_RERE_QAQ}

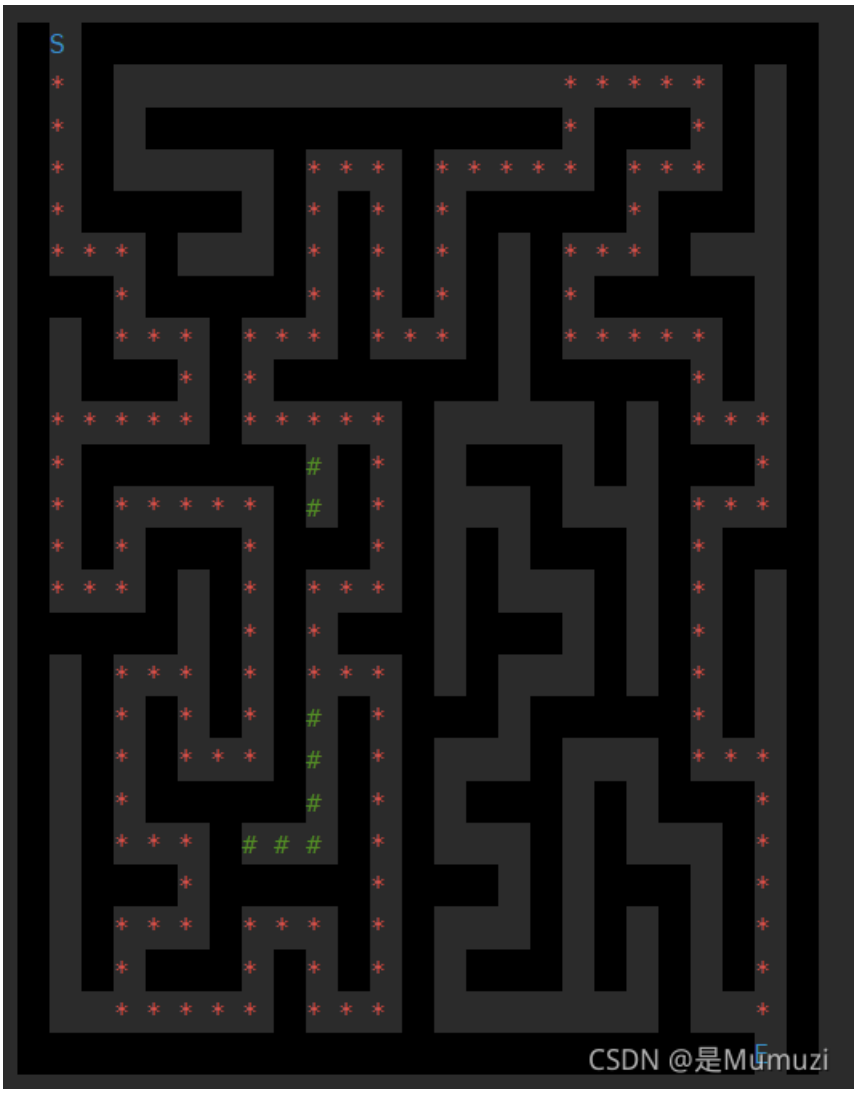
老鼠走迷宫

嗐我还以为是啥一直没做，结果是python逆向，直接用python-exe-unpacker-master逆出pyc然后uncompyle6逆一下，得到地图，我先用PIL画了一下发现挺长，然后又想起了2021DASCTF实战精英夏令营暨DASCTF July X CBCTF里用了一个走迷宫的，效果不错，直接套上去了

```
dirs = [(0, 1), (1, 0), (0, -1), (-1, 0)]  # 当前位置四个方向的偏移量
path = []  # 存找到的路径


def mark(maze, pos):  # 给迷宫maze的位置pos标"2"表示“倒过了”
    maze[pos[0]][pos[1]] = 2


def passable(maze, pos):  # 检查迷宫maze的位置pos是否可通行
    return maze[pos[0]][pos[1]] == 0


def find_path(maze, pos, end):
    mark(maze, pos)
    if pos == end:
        print(pos, end=" ")  # 已到达出口，输出这个位置。成功结束
        path.append(pos)
        return True
    for i in range(4):  # 否则按四个方向顺序检查
        nextp = pos[0] + dirs[i][0], pos[1] + dirs[i][1]
        # 考虑下一个可能方向
        if passable(maze, nextp):  # 不可行的相邻位置不管
            if find_path(maze, nextp, end):  # 如果从nextp可达出口，输出这个位置，成功结束
                print(pos, end=" ")
                path.append(pos)
                return True
    return False


def see_path(maze, path):  # 使寻找到的路径可视化
    for i, p in enumerate(path):
        if i == 0:
            maze[p[0]][p[1]] = "E"
        elif i == len(path) - 1:
            maze[p[0]][p[1]] = "S"
        else:
            maze[p[0]][p[1]] = 3
```

```
            maze[p[0]][p[1]] = 3
    print("\n")
    for r in maze:
        for c in r:
            if c == 3:
                print('\033[0;31m' + "*" + " " + '\033[0m', end="")
            elif c == "S" or c == "E":
                print('\033[0;34m' + c + " " + '\033[0m', end="")
            elif c == 2:
                print('\033[0;32m' + "#" + " " + '\033[0m', end="")
            elif c == 1:
                print('\033[0;;40m' + " " * 2 + '\033[0m', end="")
            else:
                print(" " * 2, end="")
        print()


if __name__ == '__main__':
    maze = [
        [
            1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [
            1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1],
        [
            1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1],
        [
            1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
        [
            1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1],
        [
            1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1],
        [
            1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1],
        [
            1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1],
        [
            1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1],
        [
            1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1],
        [
            1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1],
        [
            1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1],
        [
            1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1],
        [
            1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1],
        [
            1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
        [
            1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1],
        [
            1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1],
        [
            1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1],
        [
            1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1],
        [
            1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1],
        [
```

```
        1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1],
    [
        1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    [
        1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    [
        1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1],
    [
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1]]
start = (0, 1)
end = (24, 23)
find_path(maze, start, end)
see_path(maze, path)
```



得到

sssssddssddssaaaassssddwwddddsssssaawwaassssddssaassddddwwddssddwwwwwwwwwaawwddwwwwaaaawwddwwwwddssssddwwwwddddwwddddssaassaassddddssddssaasssssssddssssss

md5一下就行

> NSSCTF{69193150b15c87d39252d974bc323217}

fakerandom

我又双叒叕是爆破

```
import random
flag = 'xxxxxxxxxxxxxxxxxxx'  #len=20
random.seed(1)
l = []
for i in range(4):
    l.append(random.getrandbits(8))
    print(l)
result = [201, 8, 198, 68, 131, 152, 186, 136, 13, 130, 190, 112, 251, 93, 212, 1, 31, 214, 116, 244]
flag = ''
for i in range(len(l)):
    random.seed(l[i])
    for n in range(5):
        tmp = random.getrandbits(8)
        for j in range(32,128):
            tmps = j^tmp
            if(tmps == result[i*5+n]):
                flag += chr(j)
                break
print(flag)
```

> NSSCTF{FakeE_random}

fakebase

这个得爆破吧
推一下就知道每次都要上一次得到的数*31 + 这次得到的数。
然后数的话从密文里得，找索引。
但是最后取余得到的数不知道是多少，没有输出，于是爆破

```
s_box = 'qwertyuiopasdfghjkzxcvb123456#$'
import libnum
s = "u#k4ggia61egegzjuqz12jhfspfkay"
count = []
s = s[::-1]
for i in s:
    ind = s_box.find(i)
    count.append(ind)
print(count)
flag = 1
for j in range(31):
    for i in range(len(count)):
        if(i==0):
            flag = j*31 + count[i]
        else:
            flag = flag*31 + count[i]
    print(libnum.n2s(flag))
```

> NSSCTF{WHAt_BASe31}

astJS

最后发现一段密文，是EXXH_MpjxBxYnjggrM~eerv
然后去和NSSCTF做异或或者相减来找规律，发现异或得到的数都相同，于是找到规律，找到flag

```
s = 'EXXH_Mpjx BxYnjggrM~eerv'
f = 'NSSCTF'
for i in range(len(s)):
    print(chr(11^ord(s[i])),end='')
```

> NSSCTF{astIsReallyFunny}

easyapp

随便找个逆的工具，我用jadx
注意这里这个app先要加上.zip后缀解压，里面才是app…
在com MainActivety里面找到密文
梘椤椤棲梂橄棊棲梳棚榮榮棵椤棌
然后发现chr之后，高2位都相同，低2位都不同，于是用低2位与NSSCTF做异或和加减操作，又发现异或得到的值相同，于是又找到规律

```
s = '梘椤椤棲梂橄棊棲梳棚榮榮棵椤棌'
f = 'NSSCTF'
for i in range(len(s)):
    print(chr(int(hex(ord(s[i]))[4:],16)^177),end='')
```

> NSSCTF{apkYYDS}

PYRE

本来还是python-exe-unpacker-master逆出pyc然后uncompyle6逆一下，结果uncompyle6我逆不出来，只能去手撸字节码，详细方法是用以下指令对pyc文件做处理

```
import dis, marshal, sys

header_sizes = [
    # (size, first version this applies to)
    # pyc files were introduced in 0.9.2 way, way back in June 1991.
    (8,  (0, 9, 2)),  # 2 bytes magic number, \r\n, 4 bytes UNIX timestamp
    (12, (3, 6)),     # added 4 bytes file size
    # bytes 4-8 are flags, meaning of 9-16 depends on what flags are set
    # bit 0 not set: 9-12 timestamp, 13-16 file size
    # bit 0 set: 9-16 file hash (SipHash-2-4, k0 = 4 bytes of the file, k1 = 0)
    (16, (3, 7)),     # inserted 4 bytes bit flag field at 4-8
    # future version may add more bytes still, at which point we can extend
    # this table. It is correct for Python versions up to 3.9
]
header_size = next(s for s, v in reversed(header_sizes) if sys.version_info >= v)

with open('code.pyc', "rb") as f:
    metadata = f.read(header_size)  # first header_size bytes are metadata
    code = marshal.load(f)          # rest is a marshalled code object

dis.dis(code)
```

这个是网上有的，生成的东西因为O和0太多，我进行了处理
看这里https://pastebin.ubuntu.com/p/X5xXWF6cQ4/

然后开始手搓，总之搞了半个多小时，弄完了
其中我把tmp4改成了flag
如下

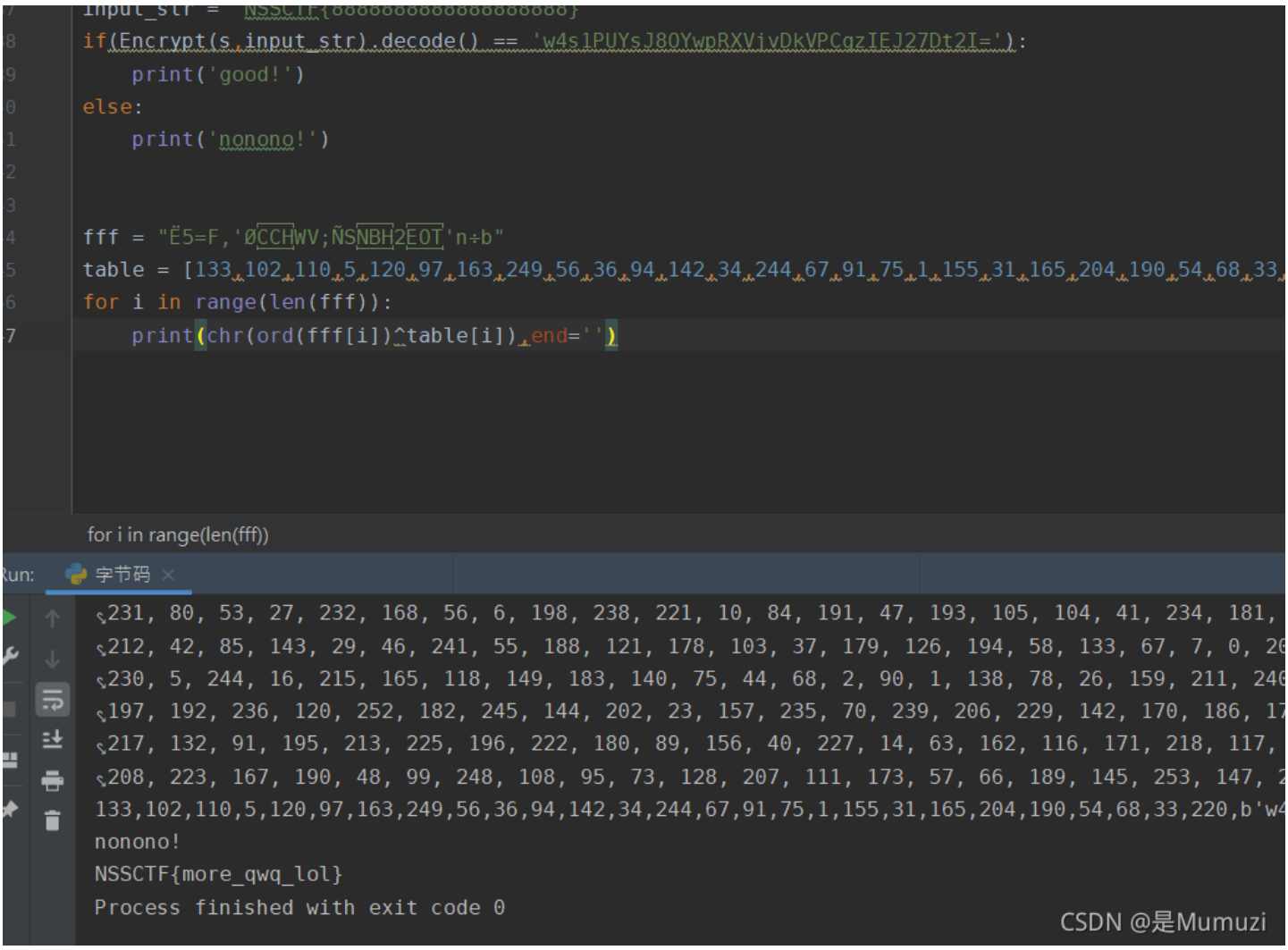
```
import hashlib
import base64
def init(s2,enc):
    a1 = 0
    enc = hashlib.md5(enc.encode()).hexdigest()
    a2 = []
    for enc2 in range(256):
        s2.append(enc2)
        a2.append(enc[enc2%len(enc)])
    for enc2 in range(256):
        a1 = ((a1+s2[enc2])+ord(a2[enc2]))%256
        s2[a1],s2[enc2] = s2[enc2],s2[a1]


def Encrypt(tmp6,flag):
    tmp = 0
    tmp2 = 0
    tmp3 = ''
    for tmp5 in flag:
        tmp = (tmp+1)%256
        tmp2 = (tmp2 + tmp6[tmp]) % 256
        tmp6[tmp2],tmp6[tmp] = tmp6[tmp],tmp6[tmp2]
        tmp7 = (tmp6[tmp] + tmp6[tmp2])%256
        tmp8 = chr(ord(tmp5)^(tmp6[(tmp6[tmp]+tmp6[tmp2])%256]))
        tmp3 += tmp8
    tmp3 = base64.b64encode(tmp3.encode())
    print(tmp3)
    return tmp3

input_str = input('input flag pls:')
s = []
init(s,'bJLVFYw3WI5ncGez')
print(s)
if(Encrypt(s,input_str).decode() == 'w4s1PUYsJ8OYwpRXVjvDkVPCgzIEJ27Dt2I='):
    print('good!')
else:
    print('nonono!')
```

输入一个NSSCTF进去，和需要对比的前几位相等，那就没事了。
然后逆向吧
第一步肯定是解码base，第二步，把tmp6[(tmp6[tmp]+tmp6[tmp2])%256]的值找出来，base解码之后就知道flag总长度
中间插入一个print(tmp6[(tmp6[tmp]+tmp6[tmp2])%256],end=',')
得到133,102,110,5,120,97,163,249,56,36,94,142,34,244,67,91,75,1,155,31,165,204,190,54,68,33,220

解密脚本如下：

```
fff = "Ë5=F,'Ø”WV;ÑSƒ2'n÷b" #base64解码
table = [133,102,110,5,120,97,163,249,56,36,94,142,34,244,67,91,75,1,155,31,165,204,190,54,68,33,220]
for i in range(len(fff)):
    print(chr(ord(fff[i])^table[i]),end='')
```

注：复制到CSDN的好像有问题，自己解码操作一遍就行



> NSSCTF{more_qwq_lol}

第二波放题

re1

如果没记错好像是原题，或者改个flag的题

IDA打开，F5

```
_main();
strcpy(Str2, "{34sy_r3v3rs3}");
printf("please put your flag:");
scanf("%s", Str1);
for ( i = 0; i <= 665; ++i )
{
  if ( Str1[i] == 101 )
    Str1[i] = 51;
}
for ( i = 0; i <= 665; ++i )
{
  if ( Str1[i] == 97 )
    Str1[i] = 52;
}
if ( strcmp(Str1, Str2) )
  printf("you are wrong,see again!");
else
  printf("you are right!");
system("pause");
return 0;
}
```

101-->e
51-->3
输入的e会变成3,
同理输入的a会变成4
所以flag为
{easy_reverse}

NSSCTF{easy_reverse}

re1和re2多解很正常，但是理解到出题人意思就行

re2

IDA F5

```
_main();
strcpy(Str2, "ylqq]aycqyp{");
printf(&Format);
gets(Str);
v7 = strlen(Str);
for ( i = 0; i < v7; ++i )
{
  if ( (Str[i] <= 96 || Str[i] > 98) && (Str[i] <= 64 || Str[i] > 66) )
    Str[i] -= 2;
  else
    Str[i] += 24;
}
if ( strcmp(Str, Str2) )
  printf(&byte_404024);
else
  printf(aBingo);
system("pause");
return 0;
}
```

就输入的在这个范围就-2
不在这个范围就+24
这题不要直接逆
爆他

```
s = 'ylqq]aycqyp{'
for i in s:
    for tmp in range(32,128):
        s = ord(i)
        if((tmp<=96 or tmp >98) and (tmp <= 64 or tmp >66)):
            if(tmp-2==s):
                print(chr(tmp),end='')
                break
        else:
            if(tmp+24==s):
                print(chr(tmp),end='')
                break
```

输出

> anss_caesar}

第一个a肯定要换成{才河里，所以flag为

> NSSCTF{nss_caesar}