

SWPU新生赛2021 Reverse部分WriteUp

原创

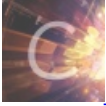
是Mumuzi 于 2021-10-11 18:14:24 发布 528 收藏 2

分类专栏: [ctf NSSCTF](#) 文章标签: [网络安全](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

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

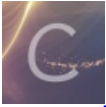
版权



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 = 'xxxxxxxxxxxxxxxxxxxx'
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]
```

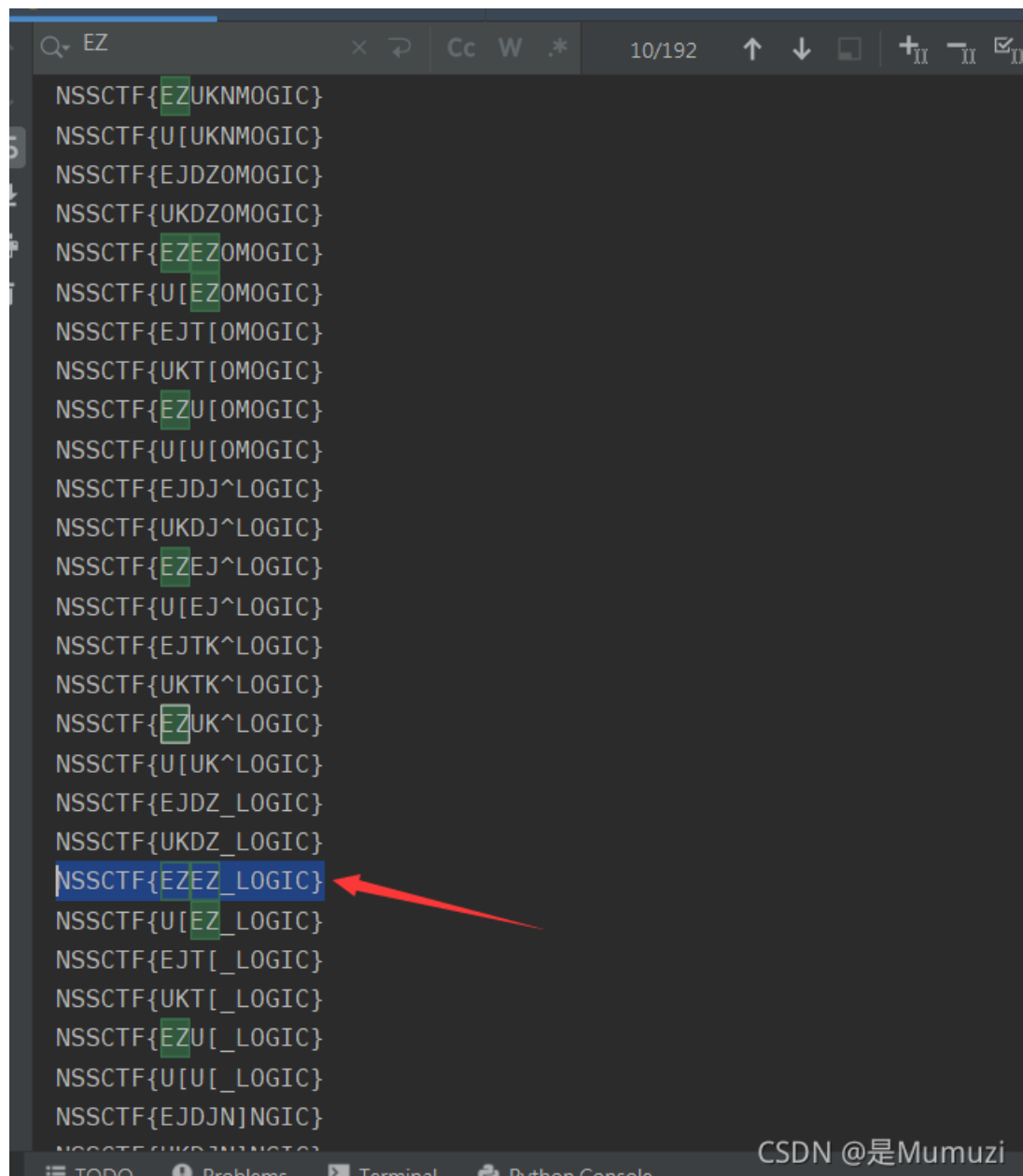
于是想这爆破, 但是脚本写的很烂

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

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



```
Q EZ 10/192
NSSCTF{EZUKNMOGIC}
NSSCTF{U[UKNMOGIC}
NSSCTF{EJDZOMOGIC}
NSSCTF{UKDZOMOGIC}
NSSCTF{EZEZOMOGIC}
NSSCTF{U[EZOMOGIC}
NSSCTF{EJT[OMOGIC}
NSSCTF{UKT[OMOGIC}
NSSCTF{EZU[OMOGIC}
NSSCTF{U[U[OMOGIC}
NSSCTF{EJDJ^LOGIC}
NSSCTF{UKDJ^LOGIC}
NSSCTF{EZEJ^LOGIC}
NSSCTF{U[EJ^LOGIC}
NSSCTF{EJTK^LOGIC}
NSSCTF{UKTK^LOGIC}
NSSCTF{EZUK^LOGIC}
NSSCTF{U[UK^LOGIC}
NSSCTF{EJDZ_LOGIC}
NSSCTF{UKDZ_LOGIC}
NSSCTF{EZEZ_LOGIC}
NSSCTF{U[EZ_LOGIC}
NSSCTF{EJT[_LOGIC}
NSSCTF{UKT[_LOGIC}
NSSCTF{EZU[_LOGIC}
NSSCTF{U[U[_LOGIC}
NSSCTF{EJDJN]NGIC}
NSSCTF{UKDZ]MNGIC}
CSDN @是Mumuzi
```

```
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

唔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uncomply6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]] = 2

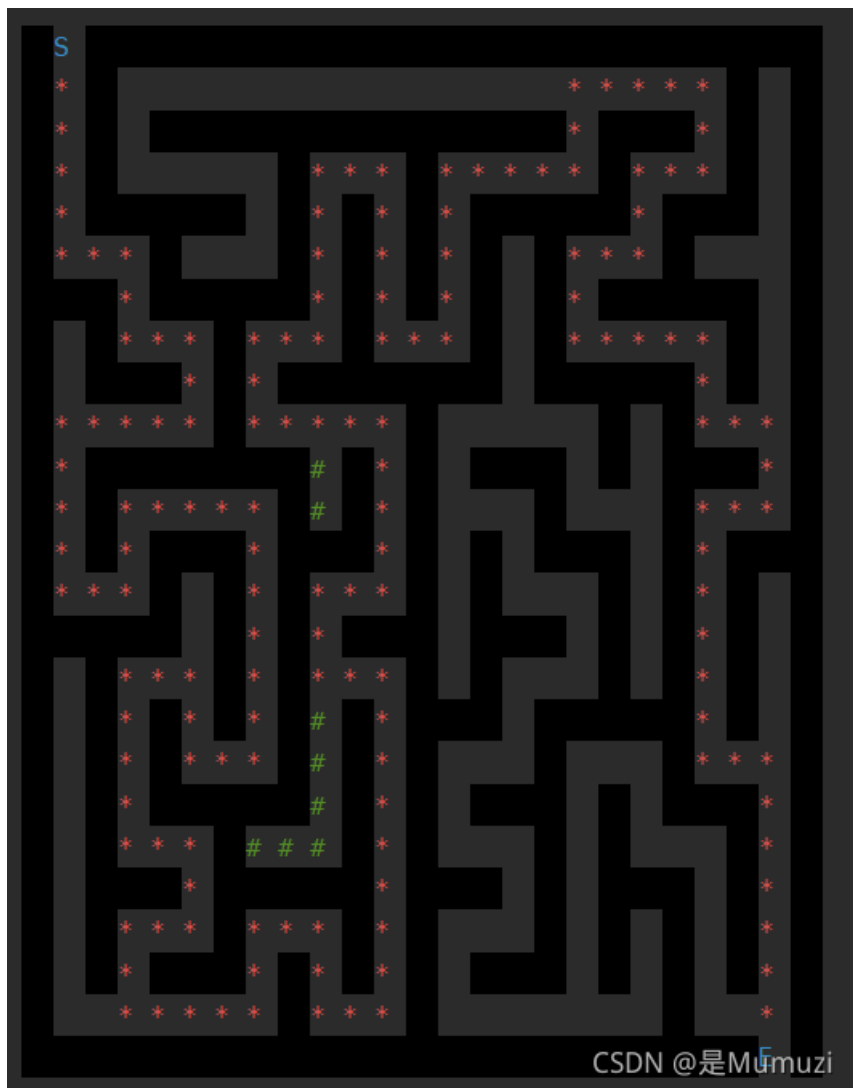
```



```

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

```



得到

```

sssssddssddssaaaassssddwwdddsssssaawwaassssddssaassdddwwdssdwwwwwwwwaawwddwwwwaaaawddwwwwdd
ssssddwwwwdddwwdddssaassaassdddssddssaassssssddssssss
md5一下就行

```

```

NSSCTF{69193150b15c87d39252d974bc323217}

```

fakerandom

我又双叒叕是爆破

```
import random
flag = 'xxxxxxxxxxxxxxxxxxxx' #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 = 'qwertyuiopasd fghjkzxcvb123456#$'
import libnum
s = "u#k4ggia61egegzjuqz12jhfsfkay"
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_MpjbXynggrM~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{astlsReallyFunny}

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.MainActivity里面找到密文

祝桐桐棲榭椒棊棊祝棚榮榮棵桐採

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 = '祝桐桐棲榭椒棊棊祝棚榮榮棵桐採'
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() == 'w4s1PUYsJ80YwpRXVjvDkVPCgzIEJ27Dt2I='):
    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解码之后就知道了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"ŵV;ÑSf2'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

```
7 input_str = NSSCTF{88888888888888888888}
8 if(Encrypt(s_input_str).decode() == 'w4s1PUYsJ80YwpRXVjvDkVPCgzIEJ27Dt2I='):
9     print('good!')
10 else:
11     print('nonono!')
12
13
14 fff = "É5=F, 'ØCCHwV;ÑSNBH2EOT'n÷b"
15 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]
16 for i in range(len(fff)):
17     print(chr(ord(fff[i])^table[i]),end=' ')
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

for i in range(len(fff))

Run: 字节码 ×

```
√231, 80, 53, 27, 232, 168, 56, 6, 198, 238, 221, 10, 84, 191, 47, 193, 105, 104, 41, 234, 181,
√212, 42, 85, 143, 29, 46, 241, 55, 188, 121, 178, 103, 37, 179, 126, 194, 58, 133, 67, 7, 0, 20
√230, 5, 244, 16, 215, 165, 118, 149, 183, 140, 75, 44, 68, 2, 90, 1, 138, 78, 26, 159, 211, 240
√197, 192, 236, 120, 252, 182, 245, 144, 202, 23, 157, 235, 70, 239, 206, 229, 142, 170, 186, 17
√217, 132, 91, 195, 213, 225, 196, 222, 180, 89, 156, 40, 227, 14, 63, 162, 116, 171, 218, 117,
√208, 223, 167, 190, 48, 99, 248, 108, 95, 73, 128, 207, 111, 173, 57, 66, 189, 145, 253, 147, 2
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,b'w4
nonono!
NSSCTF{more_qwq_lol}
Process finished with exit code 0
```

CSDN @是Mumuzi

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 )           101-->e
        Str1[i] = 51;           51-->3
}
for ( i = 0; i <= 665; ++i )
{
    if ( Str1[i] == 97 )           同理输入的a会变成4
        Str1[i] = 52;
}
if ( strcmp(Str1, Str2) )         所以flag为
    printf("you are wrong,see again!");
else
    printf("you are right!");     {easy_reverse}
system("pause");
return 0;
}
```

CSDN @是Mumuzi

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;           就输入的在这个范围就-2
}
if ( strcmp(Str, Str2) )         不在这个范围就+24
    printf(&byte_404024);       这题不要直接逆
else
    printf(aBingo);             爆他
system("pause");
return 0;
}
```

CSDN @是Mumuzi

```
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}
```