

一道比较基础的re题目，但是很考验细心程度

## 收获




做题一定要细心，不能粗心大意

但是先大概知道程序的运行逻辑和结构是非常有必要的

## 题解

### 解压

拿到的是tar文件，首先我们要做的应该是解压它

|  |                 |        |       |
|--|-----------------|--------|-------|
|  attachment.tar | 2021/5/20 19:22 | TAR 文件 | 32 KB |
|  exp.py         | 2021/5/20 19:48 | PY 文件  | 1 KB  |
|  rome.exe       | 2020/3/5 18:02  | 应用程序   | 28 KB |

解压后得到rome.exe文件

### ida直接拖入

```
unsigned __int8 v2; // [esp+24h] [ebp-34h] BYREF
unsigned __int8 v3; // [esp+25h] [ebp-33h]
unsigned __int8 v4; // [esp+26h] [ebp-32h]
unsigned __int8 v5; // [esp+27h] [ebp-31h]
unsigned __int8 v6; // [esp+28h] [ebp-30h]
int v7; // [esp+29h] [ebp-2Fh]
int v8; // [esp+2Dh] [ebp-2Bh]
int v9; // [esp+31h] [ebp-27h]
int v10; // [esp+35h] [ebp-23h]
unsigned __int8 v11; // [esp+39h] [ebp-1Fh]
char v12[29]; // [esp+3Bh] [ebp-1Dh] BYREF

strcpy(v12, "Qsw3sj_lz4_Ujw@1");
printf("Please input:");
scanf("%s", &v2);
result = v2;
```

没有对代码做混淆，逻辑比较简单，这里利用了scanf函数溢出的功能。

```

37     {
38         v1[0] = v7;
39         v1[1] = v8;
40         v1[2] = v9;
41         v1[3] = v10;
42         *(_DWORD *)&v12[17] = 0;
43         while ( *(int *)&v12[17] <= 15 )
44         {
45             if ( *((char *)v1 + *(_DWORD *)&v12[17]) > 64 && *((char *)v1 + *(_DWORD *)&v12[17]) <= 90 )
46                 *((_BYTE *)v1 + *(_DWORD *)&v12[17]) = (*((char *)v1 + *(_DWORD *)&v12[17]) - 51) % 26 + 65;
47             if ( *((char *)v1 + *(_DWORD *)&v12[17]) > 96 && *((char *)v1 + *(_DWORD *)&v12[17]) <= 122 )
48                 *((_BYTE *)v1 + *(_DWORD *)&v12[17]) = (*((char *)v1 + *(_DWORD *)&v12[17]) - 79) % 26 + 97;
49             ++*(_DWORD *)&v12[17];
50         }
51         *(_DWORD *)&v12[17] = 0;
52         while ( *(int *)&v12[17] <= 15 )
53         {
54             result = (unsigned __int8)v12[*(_DWORD *)&v12[17]];
55             if ( *((_BYTE *)v1 + *(_DWORD *)&v12[17]) != (_BYTE)result )
56                 return result;
57             ++*(_DWORD *)&v12[17];
58         }
59         result = printf("You are correct!");
60     }
61 }
62 }
63 }
64 }

```

也就是scanf读入，然后溢出到除v2以外的其他变量。

前5个字符就是判断输入是否符合格式，以及最后一个输入是否为}

## 分析代码逻辑

主要逻辑的指针的引用比较多，容易绕晕。

其实是对输入做了换值操作。

即如果是'A'到'Z'则原值-51后求模26，最后加上65作为新值

如果是'a'到'z'则原值-79后求模26，最后加上97作为新值

本来这种逻辑是有漏洞的，即可能有两个不同的字符，经过操作后映射为同一个新值，但是上面的输入里不会出现这种情况。

直接写脚本爆破

```
exp.py - C:\Users\wsxk\Desktop\ctf\BUUCTF\ACTF新生赛2020\rome\exp.py (3.6.8)
File Edit Format Run Options Window Help
key = 'Qsw3sj_lz4_Ujw@l'.encode()
key = list(key)
print(len(key))
flag = []
for i in range(len(key)):
    if key[i]>64 and key[i]<=90:
        for j in range(65, 91):
            if (j-51)%26+65 == key[i]:
                flag.append(j)
                break
    else:
        if key[i]>=97 and key[i]<123:
            for j in range(97, 123):
                if (j-79)%26+97 == key[i]:
                    flag.append(j)
                    break
        else:
            flag.append(key[i])
print(bytes(flag))
```

## 错误示范

因为当时漏看了指针或者少引用了啥操作。导致脚本一直跑不对。。。所以才要细心看代码。