**网鼎杯2024 revers1（four）**

将 four 文件拖进 Exeinfo 分析，64 位无壳的 elf 文件。

打开IDA Pro进行分析进入字符串窗口，可以看到一串很base64码表的东西，但是只有63位先不管 DEFGHIJKLMNOPQRSTUVWXYZABabcdefghijklmnopqrstuvwxyz0123456789+/，下面有 Enter the flag

字样，双击，Ctrl + x 查看交叉引用，来到 main 函数，F5 进行反编译。

可以看到这里要求输入字符串给变量 s，长度必须是 40，暂时 v20 看不出来是什么，他将 32 位 给了dst

仔细观察观察发现有四个 if 函数，如果为真则就退出，里面的条件是 cmp 比较每次比较 8 位， 也就时

将输入的字符串提取出 wdflag{xxxxx},xxxxx 一共 32 位给了dst变量，分为四段，分别进行一 些操作然

后和字符串进行比较，如果相等就返回 0 继续向下，反之就退出。分别在四个if前面加上断点通过 动态调

试找到加密后的字符串进行解密。配置远程调试。监听进程（这里用到centos 7 的调试器）

第一个if就是将dst的前八位乘以2，然后与s2进行比较。因为s2是程序执行后才进行赋值的，所以 要动态 调试，然后利用LazyIDA的convert将s2以python数组的形式输出出来。也可以一个个手打出来。 s2 = [0x6A, 0xC4, 0xCC, 0x62, 0xC2, 0x6E, 0xCA, 0xCA]

第二个if就是将第8~15位输入的字符串与v22数组进行异或，结果存放在v13数组中。最后v13与v11 进行比较，

这里要注意v11和v12是连续的地址，所以两个加起来刚好是8位的数组。 v11 = [0x60, 0x58, 0x16, 0x47, 0x7D, 0x5C, 0x44, 0x5D]

第三个if就是将16~23位输入的字符串进行base64加密，只不过base64加密函数自定义的，码表为 base\_tables CDEFGHIJKLMNOPQRSTUVWXYZABabcdefghijklmnopqrstuvwxyz0123456789+/ 前面有个C， 别看漏了。加密的密文

为 BFO1AjdmPmG 可以利用 cyberchef 进行解密，也可以手搓。

第四个if就是将24~31位输入的字符串用8来补满十六位，然后v9为密钥一共27个字节刚好128位， 使用

AES-128 ECB 加密算法加密，结果与 v4 数组进行比较。

v9 = [0x41, 0x65, 0x73, 0x4D, 0x61, 0x73, 0x74, 0x65, 0x72, 0x41, 0x65, 0x73, 0x4D, 0x61, 0x73, 0x74, 0x42, 0x46, 0x4F, 0x31, 0x41, 0x6A, 0x64, 0x6D, 0x50, 0x6D, 0x47]

v4 = [0x0F, 0xE3, 0x2F, 0xE6, 0x58, 0x20, 0x9B, 0x3A, 0xD6, 0xE4, 0x18, 0x3F, 0xA7, 0x78, 0xA5, 0x82]

完整解密脚本为：

from Crypto.Cipher import AES import binascii

flag = 'wdflag{'

# 第一部分

s2 = [0x6A, 0xC4, 0xCC, 0x62, 0xC2, 0x6E, 0xCA, 0xCA] for i in s2:

flag += chr(int(i/2))

# 第二部分

v11 = [0x60, 0x58, 0x16, 0x47, 0x7D, 0x5C, 0x44, 0x5D] v22 = [0x58, 0x6f, 0x72, 0x72, 0x4c, 0x6f, 0x72, 0x64] for i in range(8):

flag += chr(v11[i] ^ v22[i])

# 第三部分 base\_tables =

'CDEFGHIJKLMNOPQRSTUVWXYZABabcdefghijklmnopqrstuvwxyz0123456789+/' # 加密字符串

m = 'BFO1AjdmPmG'

# 利用cyberchef进行解密 flag += 'd35b7f6a'

# 将密文和密钥从十六进制格式转换为字节

ss = bytes([0x0F, 0xE3, 0x2F, 0xE6, 0x58, 0x20, 0x9B, 0x3A, 0xD6, 0xE4, 0x18, 0x3F, 0xA7, 0x78,0xA5, 0x82])

key\_128 = bytes([0x41, 0x65, 0x73, 0x4D, 0x61, 0x73, 0x74, 0x65, 0x72, 0x41, 0x65, 0x73, 0x4D, 0x61, 0x73, 0x74])

# 使用 AES-128 ECB 模式解密

cipher = AES.new(key\_128, AES.MODE\_ECB) decrypted = cipher.decrypt(ss)

# 输出解密后的结果

flag += str(decrypted[:8]).replace('\'','') flag += '}'

print(flag)