## 原文链接：

<https://www.anquanke.com/post/id/217151?display=mobile>

## [AFCTF2018]可怜的RSA

* 题目链接（buu平台）

[https://buuoj.cn/challenges#[AFCTF2018]可怜的RSA](https://buuoj.cn/challenges" \l "[AFCTF2018]%E5%8F%AF%E6%80%9C%E7%9A%84RSA)

* 拿到附件：
  + flag.enc
  + public.key

就明显是拿到两个附件，一个是RSA加密的密钥文件，还有一个是flag加密后的密文的文件。

这边的话一种就是常规的用openssl，个人不太喜欢用。python能解决的问题我一般就python解决了。这边用到python的Crypto库，是做CTF密码学非常常用的一个库。

pip install pycrypto

安装好这个Crypto库  
然后导入Crypto.PublicKey.RSA

用RSA模块的import\_key函数将我们的publickey读入，拿到RSA加密的公钥n和e的具体参数

**from** Crypto.PublicKey **import** RSA

f=open("public.key","r")

key=RSA.import\_key(f.read())

f.close()

e=key.e

n=key.n

然后发现n可以分解，直接在线网站<http://factordb.com/>把n给分解了，得到公钥n的两个因子。  
然后就是常规的RSA求解私钥

这边因为flag.enc是RSA的PKCS1\_OAEP加密得来的。所以我们这边也是给生成一个私钥文件。

在做到这边的时候，如何导出一个私钥文件。找了一下百度上的方法。都是先generate后给参数分别赋值的。但是我发现我并不行，试了一下python3和python2下的Crypto库都得到一个报错

Exception has occurred: AttributeError

can't set attribute

也就是现在无法通过这么直接赋值了。  
这种情况的话，可以去看下python调用的Crypto库里面的RSA模块的一个底层的实现。

发现有一个construct函数，传入一个rsa\_components参数，是一个元组型的数据，也就是tuple类型的，分别是(n,e,d,p,q)

phi=(p-1)\*(q-1)

d=gmpy2.invert(e,phi)

rsa\_components=(n,e,int(d),p,q)

arsa=RSA.construct(rsa\_components)

arsa.exportKey()

然后导出的私钥，对加密后的密文，使用PKCS1\_OAEP模块进行解密即可。

* 题解exp如下

**from** Crypto.PublicKey **import** RSA**from** Crypto.Cipher **import** PKCS1\_OAEP

f=open("public.key","r")

key=RSA.import\_key(f.read())

f.close()

e=key.e

n=key.n**import** base64**from** Crypto.Util.number **import** \***import** gmpy2

p= 3133337

q=25478326064937419292200172136399497719081842914528228316455906211693118321971399936004729134841162974144246271486439695786036588117424611881955950996219646807378822278285638261582099108339438949573034101215141156156408742843820048066830863814362379885720395082318462850002901605689761876319151147352730090957556940842144299887394678743607766937828094478336401159449035878306853716216548374273462386508307367713112073004011383418967894930554067582453248981022011922883374442736848045920676341361871231787163441467533076890081721882179369168787287724769642665399992556052144845878600126283968890273067575342061776244939

print(p\*q==n)

f=open("flag.enc","r")

c\_base64=f.read().strip("\n")

c\_bytes=base64.b64decode(c\_base64)

c=bytes\_to\_long(c\_bytes)

phi=(p-1)\*(q-1)

d=gmpy2.invert(e,phi)

rsa\_components=(n,e,int(d),p,q)

arsa=RSA.construct(rsa\_components)

rsakey = RSA.importKey(arsa.exportKey())

rsakey = PKCS1\_OAEP.new(rsakey)

decrypted = rsakey.decrypt(c\_bytes)

print(decrypted)