

XCTF攻防世界 Normal_RSA

原创

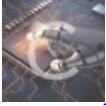
[CleanMe](#) 于 2020-12-06 22:07:15 发布 2084 收藏 9

分类专栏: [ctf crypto](#) 文章标签: [加密解密](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/m0_49490199/article/details/110730250

版权



[ctf](#) 同时被 2 个专栏收录

1 篇文章 0 订阅

订阅专栏



[crypto](#)

2 篇文章 0 订阅

订阅专栏

XCTF攻防世界 Normal_RSA

实验环境: windows 10

实验所需工具:

python工具:

[yafu](#) (可以在<https://github.com/DarkenCode/yafu>上下载)

[gmpy2](#) (直接pip install gmpy2即可)

rsa加密的相关知识可以看这两篇文章

http://www.ruanyifeng.com/blog/2013/06/rsa_algorithm_part_one.html

http://www.ruanyifeng.com/blog/2013/07/rsa_algorithm_part_two.html

使用yafu对n进行质因数分解，到yafu安装的文件目录输入yafu-x64使用

输入 `factor(87924348264132406875276140514499937145050893665602592992418171647042491658461)`

即可分解出质因数

```
E:\CTF\CTF\crypto\yafu-1.34>yafu-x64
factor(87924348264132406875276140514499937145050893665602592992418171647042491658461)

fac: factoring 87924348264132406875276140514499937145050893665602592992418171647042491658461
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
rho: x^2 + 3, starting 1000 iterations on C77
rho: x^2 + 2, starting 1000 iterations on C77
rho: x^2 + 1, starting 1000 iterations on C77
pm1: starting B1 = 150K, B2 = gmp-ecm default on C77
ecm: 30/30 curves on C77, B1=2K, B2=gmp-ecm default
ecm: 74/74 curves on C77, B1=11K, B2=gmp-ecm default
ecm: 149/149 curves on C77, B1=50K, B2=gmp-ecm default, ETA: 0 sec
starting SIQS on c77: 87924348264132406875276140514499937145050893665602592992418171647042491658461
==== sieving in progress (1 thread): 36224 relations needed ====
==== Press ctrl-c to abort and save state ====
36209 rels found: 18054 full + 18155 from 193233 partial, (2000.14 rels/sec)
SIQS elapsed time = 107.0532 seconds.
Total factoring time = 120.3009 seconds

***factors found***
P39 = 319576316814478949870590164193048041239
P39 = 275127860351348928173285174381581152299
ans = 1
```

其中的两个p39为分解出来的两个质因数

`p1 = 319576316814478949870590164193048041239`

`p2 = 275127860351348928173285174381581152299`

然后在计算n的欧拉函数 $\phi(n) = (p1-1)*(p2-1)$

```
>>> p1 = 319576316814478949870590164193048041239
>>> p2 = 275127860351348928173285174381581152299
>>> y=(p1-1)*(p2-1)
>>> y
87924348264132406875276140514499937144456189488436765114374296308467862464924L
>>>
```

`$\phi(n) = 87924348264132406875276140514499937144456189488436765114374296308467862464924$`

再利用 `gmpy2` 计算出 e 模 $\phi(n)$ 的逆元 d

`gmpy2.invert()`方法中第一个参数为 e ，第二个参数为 $\phi(n)$ ，通过此方法计算可以得到 d

```
>>> p1 = 319576316814478949870590164193048041239
>>> p2 = 275127860351348928173285174381581152299
>>> y = (p1-1)*(p2-1)
>>> y
87924348264132406875276140514499937144456189488436765114374296308467862464924L
>>>
>>> import gmpy2
>>> e=65537
>>> d=gmpy2.invert(e,y)
>>> d
mpz(10866948760844599168252082612378495977388271279679231539839049698621994994673)
>>>
```

得到 $d = 10866948760844599168252082612378495977388271279679231539839049698621994994673$

到这里，我们已经知道了

```
n = 87924348264132406875276140514499937145050893665602592992418171647042491658461
e = 65537
d = 10866948760844599168252082612378495977388271279679231539839049698621994994673
```

看到好多文章中都用了 `rsatool` 这个工具，但是我安装总是会报错，也不知道是什么原因，现在都没安装好。。。

但是看了一下 `rsatool` 的作用是已知 n, e, d 和密文文件可以进行解密得到从而得到明文

既然用不了 `rsatool`，那就根据RSA加解密的原理来计算

设 m 为明文， c 为密文，则RSA的加解密算法为

$$c = m^e \pmod n$$

密文 c 在 `flag.enc` 文件中，打开文件看到是这样的

```
m>·ß#îáÓ†<0x10>¼°x †ž<0x0e>æe½=<0x08>ImÚd'A™<0x11><0x0c>y
```

但根据公式， c 需要为数字，那么就用16进制编码重新打开文件，得到了密文的16进制形式

```
6d3e b7df 23ee e1d3 8710 beba 78a0 878e
0e9c 65bd 3d08 496d da64 9241 9911 0c79
```

$c = 0x6d3eb7df23eee1d38710beba78a0878e0e9c65bd3d08496dda64924199110c79$

根据

$$m = c^d \pmod n$$

现在已经得到了明文 **m** 的**16进制**形式

那么就可以把16进制转换为字符串，即可得到明文

但是又出现了一个问题，我放到在线网站上转换为字符串的时候，转换不出字符串

16进制转换文本 / 文本转16进制



https://blog.csdn.net/m0_49490199

后来发现这串16进制的字符个数为奇数个，正常的话16进制的字符应该是由两个**0~f**的数组成，也就是这串16进制的字符数的个数一定是偶数

```
>>> hex(m)
'0x2c0fe04e3260e5b8700504354467b323536625f69355f6d336469756d7d0aL'
>>> len(hex(m))
64
>>> len(str(hex(m)))
64
>>>
```

可以看到这一串字符串的长度为 $64 - 2 - 1 = 61$ 是奇数(减2是因为最前面的0x，减1是因为最后面的L(表示长整型))
于是就把第一个字符删去，再转换为字符串



https://blog.csdn.net/m0_49490199

其中可见字符串即为**flag**