i春秋 第二届春秋欢乐赛 rsa256

原创

sps98 于 2019-11-28 01:40:15 发布 222 收藏

分类专栏： ctf crypto rsa

本文链接：https://blog.csdn.net/SPS98/article/details/103019519


ctf 同时被 3 个专栏收录

11 篇文章 0 订阅

订阅专栏


crypto

1 篇文章 0 订阅

订阅专栏


rsa

1 篇文章 0 订阅

订阅专栏

这次拿到题目后, 最起码还能推测出解题思路了, 不过因为不够成熟, 还是习惯性的去看看其他师傅写的wp.

题目给了个公钥文件`public.key`， 使用openssl解码得出n(`Modules`)与e(`Exponent`)

```
C:\Users\i\Documents\Scripts\openssl\bin>openssl rsa -pubin -text -modulus -in warmup -in t\public.key
RSA Public-Key: (256 bit)
Modulus:
    00:d9:9e:95:22:96:a6:d9:60:df:c2:50:4a:ba:54:
    5b:94:42:d6:0a:7b:9e:93:0a:ff:45:1c:78:ec:55:
    d5:55:eb
Exponent: 65537 (0x10001)
Modulus=D99E952296A6D960DFC2504ABA545B9442D60A7B9E930AFF451C78EC55D555EB
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhANmelSKWptlg38JQSrpUW5RC1gp7npMK
/OUceOxV1VXrAgMBAAE=
-----END PUBLIC KEY-----
https://blog.csdn.net/SPS98
```

> 命令:
> openssl rsa -pubin -text -modulus -in warmup -in 文件路径

n为16进制数字,先转为10进制后,

```
C:\Users\i\Documents\Scripts\openssl\bin>py
Python 3.7.4 (default, Aug  9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print(0xD99E952296A6D960DFC2504ABA545B9442D60A7B9E930AFF451C78EC55D555EB)
98432079271513130981267919056149161631892822707167177858831841699521774310891
>>>
```


不是很长,使用yafu或factordb.com分解出p和q

```
C:\Users\i\Documents\Scripts\CTF Tools\2.密码学\yafu-1.34>yafu-x64.exe
factor(98432079271513130981267919056149161631892822707167177858831841699521774310891)


fac: factoring 98432079271513130981267919056149161631892822707167177858831841699521774310891
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits

starting SIQS on c77: 98432079271513130981267919056149161631892822707167177858831841699521774310891

==== sieving in progress (1 thread):    36224 relations needed ====
====           Press ctrl-c to abort and save state           ====


SIQS elapsed time = 4.2759 seconds.
Total factoring time = 4.3379 seconds


***factors found***

P39 = 302825536744096741518546212761194311477
P39 = 325045504186436346209877301320131277983

ans = 1
```


于是,该有的参数都有了,就可以开始解密了
(这里用了师傅的脚本)

```python
import gmpy2
import rsa

p = 302825536744096741518546212761194311477
q = 325045504186436346209877301320131277983
n = 98432079271513130981267919056149161631892822707167177858831841699521774310891
e = 65537

d = int(gmpy2.invert(e, (p - 1) * (q - 1)))
private_key = rsa.PrivateKey(n, e, d, p, q)
with open('encrypted.message1', 'rb') as f:
    print(rsa.decrypt(f.read(), private_key).decode())
with open('encrypted.message2', 'rb') as f:
    print(rsa.decrypt(f.read(), private_key).decode())
with open('encrypted.message3', 'rb') as f:
    print(rsa.decrypt(f.read(), private_key).decode())
```

然后输出三行被拆分的flag(这里就不放了)

gmpy2还有rsa库整起来好麻烦=-=, 还以为conda默认带了

拖了好多天的wp终于填完了, 离flag渐行渐远=.=!!

今天才发现自己把n的生成方法搞成`p * q`了, 我好菜(哭哭)

参考链接:

- https://www.ichunqiu.com/writeup/detail/693