README.md

<<< Return to ABCTF 2016 tasks and writeups

Old RSA (Cryptography, 70 points)

> I recovered an RSA encrypted message from the 1980's. can you decrypt it?
>
> c = 298469475192145751624974137250604125461192332168511842462673577700824630302 25
>
> n = 707360252392652399763150886901745940216466548816264214610090894808706334009 73
>
> e = 3

This is a basic "RSA challenge". We are given an RSA public key (n, e) and an encrypted message (c), which is likely to be the flag. The security of the RSA algorithm is based on the mathematical difficulty of finding two prime factors of a very large number. However, this challenge has a small modulus (only 77 digits), so it is very easy to recover the two factors p and q (thanks to factorDB for example), and then easy to decrypt the flag.

Running this Python script gave us the flag and 70 points.

```
import gmpy
from Crypto.PublicKey import RSA

#problem statement
n = 707360252392652399763150886901745940216466548816264214610090894808706334009 73
c = 298469475192145751624974137250604125461192332168511842462673577700824630302 25
e = long(3)

#thanks to factorDB
p = 238324208831434331628131715304428889871
q = 296805874594538235115008173244022912163

#computing the private key exponent d
d = long(gmpy.invert(e,(p-1)*(q-1)))

#recovering the private key
key = RSA.construct((n,e,d))

print key.decrypt(c)
#687255797750574777816118221724271222836487386007058011149452654604 5

print hex(key.decrypt(c))
#0x41424354467b746831735f7761735f683472645f696e5f313938307dL

print hex(key.decrypt(c))[2:-1].decode('hex')
#ABCTF{th1s_was_h4rd_in_1980}
```