Easy RSA

Problem: We encrypted the flag with RSA, can you crack it?

Hint: Short public key

Given: easy-rsa.tar.gz (custom1.enc, custom2.enc, custom3.enc, and public\_key.pem)

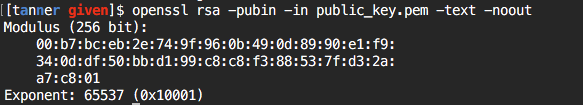
Steps:

1) Understanding the public key that is in the given folder. The public key file is the public\_key.pem file.

If you try to open this file regularly it will not open correctly. Lucky there is a tool called openssl that can make this file very readable. The following command will give you your public\_key file in a readable text

$openssl rsa –pubin –in public\_key.pem –text –noout

The following will be given:



We now have our public key(modulus), and our exponent that will be needed for decryption.

The current modulus is not usable as is. It needs to be converted to the decimal version. This can be done easily by using the python interpreter. We get to this by simply typing python in our terminal. After we are inside the python interpreter we need to create a few variables that will be used. Something like the following:

>>> line1=’00:b7:bc:eb:2e:74:9f:96:0b:49:0d:89:90:e1:f9:’

>>> line2=’32:0d:df:50:bb:d1:99:c8:c8:f3:88:53:7f:d3:2a:’

>>> line3=’a7:c8:01’

>>> total=””

>>> line1=line1.split(‘:’) #going to split up the string at each colon and put it in a list

>>> line2=line2.split(‘:’)

>>> line3=line3.split(‘:’)

>>> for i in line1:

… total+=i #this will append each of the list elements together 00b7bceb2e749f…

>>>for i in line2:

… total+=i

>>>for i in line3:

total+=i

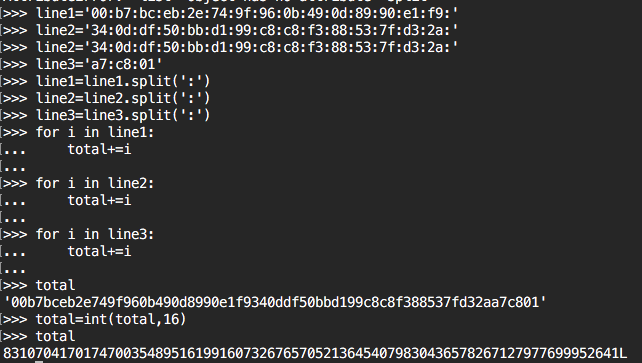
Now total will have all the individual list elements in a single string and should look like the following:

00b7bceb2e749f960b490d8990e1f9320ddf50bbd199c8c8f3885375d32aa7c801

Now we want to convert this to a decimal value.

>>>total=int(total,16) #this convers hex to decimal

The following is a screenshot of the above commands (note total=”” was already initialized before screenshot)



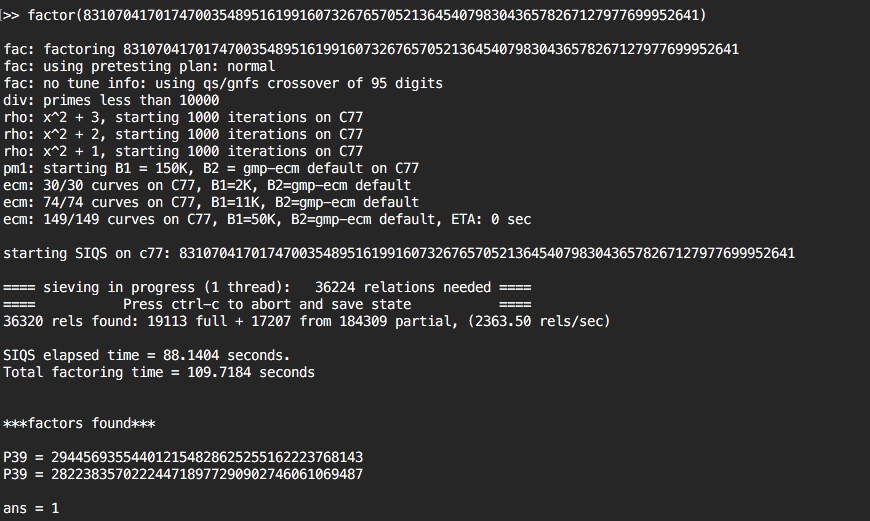
We now have to long integer which is our new modulus in decimal:

831070417017470035489516199160732676570521364540798304366578267127977699952641L

2) Now that we have the decimal version the modulus we need to get the two largest prime factors of this modulus. There is a great tool called ‘yafu’ that can be used to do this for us.

Yafu can be read about at this [link](https://github.com/DarkenCode/yafu). Yafu can be installed [here](https://sourceforge.net/projects/yafu/).

Now we will factorize this integer to get our P1 and P2 which are needed for the decryption of rsa. When running this tool it looks like the following and will take a little time.



We now have out P1 and P2. We can now start the decryption process.

P1 = 294456935544012154828625255162223768143

P2 = 282238357022244718977290902746061069487

3) Now we can begin understanding our script to decrypt the rsa files we were given.

I will go over some variables that you will need to know to go over decryption.

z = our file

c = cipher text as number

n = our long decimal value of the modulus from the public key

e = the exponent given in the public key

P1=prime factor 1 of the modulus key

P2=prime factor 2 of the modulus key

r=(prime factor 1 – 1) multiplied by (prime fact2 -1) provides our totient number

d=the decryption key

p=the plaintext as number

pt=plaintext in a string representation

There are also a couple libraries in python that we will need to have for our decryption script.

-gmpy

-Crypto.Util.number

-base64

4) Now we can begin to build our script that will decrypt the given rsa files.

First lets open the first file custom1.enc and read it as encoded hex values

z = open(‘custom1.enc’,’r’).read().encode(‘hex’)

Now turn the cipher text to a number

c = int(z,16)

Set n to be our long modulus number we got in step 2

n=83107041701747003548951619916073267657052136454079830436578267127977699952641

Set e to the exponent we were given in our public key

e = 65537

Set P1 and P2 that we obtained from the yafu tool

P1 = 282238357022244718977290902746061069487

P2 = 294456935544012154828625255162223768143

Set r to the equation to generate totient number

r = (P1-1)\*(P2-1)

For the d variable we user gmpy to get our invert of the e over r. We need to use the gmpy library that we imported earlier

d = int(gmpy.invert(e, r).digits())

Now set p to generate our plaintext using the rsa equation c^d%n

p = pow(c,d,n)

Let pt be your new plaintext that is decoded from hex into a readable string then print it to receive the plaintext. To use the decode, pt must be an even number.

pt = hex(p).strip('0x').strip('L').decode('hex')

print pt

We now have our script set up to decrypt rsa with a short public key. Go through and replace each file name with the custom1.enc, custom2.enc, and custom3.enc and concatenate the new string into your flag

FLAG{R5A\_i5\_n00b}

Sample Script:

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

import gmpy

from Crypto.Util.number import \*

import base64

z = open('custom3.enc','r').read().encode('hex')

c = int(z,16)

n = 83107041701747003548951619916073267657052136454079830436578267127977699952641

e = 65537

P1 = 282238357022244718977290902746061069487

P2 = 294456935544012154828625255162223768143

r = (P1-1)\*(P2-1)

d = int(gmpy.invert(e, r).digits())

p = pow(c,d,n)

#print hex(p)

#print p

pt = hex(p).strip('0x').strip('L').decode('hex')

print pt