HTB WRITEUP

Eurus

[CRIPTO] TwoForOne

Analysis
Theory
Decrypt

ANALYSIS

Alice sent two times the same message to Bob. Probably are two message encrypted with RSA.

Little recap about RSA. PublicKey (N,e), PrivateKey (N,d). Where N is p*q with p and q are large prime and Phi(N) = P(p*q) = (p-1)(q-1).

Whit a little of investigation we can see that the two public key share the modulus and if the two key share the same modulus we can attack the encryption.

THEORY

$C_1 = m^{e1} \, mod \, n \; and \; C_2 = m^{e2} \, mod \, n$

Using extended Euclidean algorithm:

$$e1 * u + e2 * v = gcd(e1, e2)$$

if one of the two value from u or v is negative we need to calculate:

$i = c_1^{-1} \, mod \, n \; or \; i = c_2^{-1} \, mod \, n$

and finally we can calculate the message m with:

$m = C_1^u * i^{-v} \, mod \, n \; or \; m = i^{-u} * C_2^u$

And so with the following python code we can decrypt the message and retrive the flag.

DECRYPT

```
from Crypto.PublicKey import RSA
import sys,base64

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
    return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

def pad_even(x):
        return ('', '0')[len(x)%2] + x


def CipherB2n(c):
    c2 = base64.b64decode(c)
    return int.from_bytes(c2, 'big')

def CipherN2b(m):
    hex_m=hex(m)[2:]
    if hex_m[-1] == 'L' :
        hex_m=hex_m[:-1]
    return bytes.fromhex(hex_m).decode('ascii')
```

```
def main():
  rsa_pub1 = open("./key1.pem", "r")
  key1 = RSA.importKey(rsa_pub1.read())
  n1 = key1.n

  rsa_pub2 = open("./key2.pem", "r")
  key2 = RSA.importKey(rsa_pub2.read())
  n2 = key2.n


  msg1 = open("./message1", "r")
  msg2 = open("./message2", "r")

  msg1_s = msg1.read()
  msg2_s = msg2.read()

  msg1.close()
  msg2.close()

  c1 = CipherB2n(msg1_s)
  c2 = CipherB2n(msg2_s)

  e1 = key1.e
  e2 = key2.e

  sys.setrecursionlimit(1000000)
  n_common = n1
  #[1, u, v]
  s = egcd(e1, e2)
  u = s[1]
  v = s[2]

  if(not (e1*u+e2*v)==1):
    return
  print("OK 1")

  if u<0:
    u = - u
    c1 = modinv(c1, n_common)
  elif v<0:
    v = - v
    c2 = modinv(c2, n_common)

  m=(pow(c1,u,n_common)*pow(c2,v,n_common)) % n_common

  print(m)
  print(CipherN2b(m))


if __name__ == "__main__":
  main()
```