RSA With The Dogs Writeup

Christopher Wrogg, cwrogg, christheyankee, OPz qt

March 27 2021

Background

So some background on why this challenge even came to be. We realized pretty quickly that a lot of teams solved ANYTHING by drew and then stopped doing crypto challenges and so we wanted to lure people into doing EAV-Secure Diffie–Hellman and other harder challenges. I apologize for how late this challenge was posted and not thoroughly researching all of the solution avenues that existed. Spoiler, dcode has an out of the box solution to this problem that I had no idea existed in the 15 minutes I spent writing this.

The Theory

Our lord and savior (I'm only half joking here) cryptologist Michael J. Wiener proved that against RSA systems with sufficiently small d ($d < \frac{1}{3}N^{\frac{1}{4}}$) one can recover d in polynomial time. We can identify from the given variables that Wiener's attack might be useful since e is the inverse of d in RSA and our e here is unnaturally (and wrongly) large. There are a lot of ways to abuse this but in particular we want to take advantage of the fact that:

$$edg = k(p-1)(q-1) + g$$

which is almost always true in RSA where values of g are small by convention. The derivation of this fact is trivial and on the wikipedia page about this exploit.

The Solve

There is not much to talk about here, only that I should have perhaps spent a little more time designing something that would have been a bit more resistant to online tools like dcode. I really wanted this challenge to be about guiding beginners (there were a lot) to Ganapati's github repo RsaCtfTool. I have also written my own solution that draws on their solution in sage if you're interested in checking that out. It resides on the next page.

```
from math import sqrt
from Crypto.Util.number import long_to_bytes

N = Large number, see git repo
E = Large number, see git repo
c = Large number, see git repo

def contfrac_to_rational (frac):
    if len(frac) == 0:
        return (0,1)
    num = frac[-1]
    denom = 1
    for _ in range(-2,-len(frac)-1,-1):
        num, denom = frac[_]*num+denom, num
    return (num,denom)

n = N
e = E

a = e//n
frac = [a]
while a * n != e:
    e,n = n,e-a*n
    a = e//n
    frac.append(a)
convergents = []
for i in range(len(frac)):
    convergents.append(contfrac_to_rational(frac[0:i]))

for (k,d) in convergents:
    if k!=0 and (E*d-1)%k == 0:
        phi = (E*d-1)//k
        s = N - phi + 1
        discr = s*s - 4*N
        if(discr>=0):
            t = -1
            if discr.is_square():
                t = sqrt(discr)
            if t!=-1 and (s+t)%2==0:
                break
print(long_to_bytes(int(pow(c,d,N))))
```