RSAextreme

RSA_Extreme

Headorteil

Table des matières

Le sujet

On nous donne une commande : `ssh -i rsa_priv_key rsaextreme@157.159.40.163`, une clef publique a télécharger et un indice : « Tu connais les fractions continues ?»

Analyse du problème

On voit donc qu'on doit générer une clé privé à l'aide d'une clé publique. On a donc un N et un e et on cherche a obtenir un d. En effet, les RSA fonctionnent de la manière suivante : on génère deux nombres premiers p et q et on note leur produit N. On génère ensuite e et d tels que $ed \equiv 1[\varphi(N)]$ avec $\varphi(N) = (p-1)(q-1)$. La clef publique est alors le couple (N,e) et la clef privée (N,d).

C'est parti

Tout d'abord on doit extraire notre N et notre e grâce à la commande :

```
openssl rsa -in rsa_pub_key -pubin -text -m
```

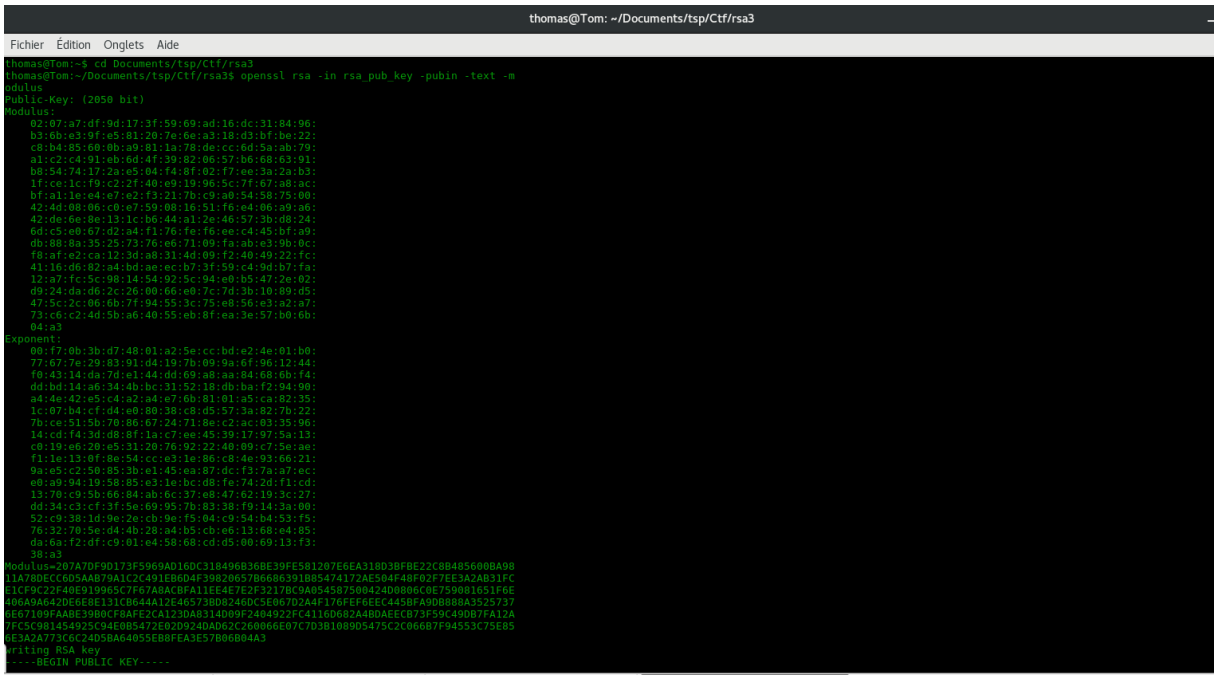

FIG. 1: Voilou le N et le e en hexadécimal

On obtient donc N : le « modulus » qu'on obtient en base 10 grâce à la commande :

```
echo 'ibase=16; <modulus> ' | bc
```

On convertit ensuite le e : « exponent » en base 10 aussi grâce a un site dédié, les « : » et les retours à la ligne étant problématiques pour itérer la méthode précédente.

A l'attaque !!

On se retrouve donc a la recherche d'un d en ayant qu'un N et un e, c'est ici que l'indice intervient, on nous parle de fractions continues, on va donc implémenter une attaque de Wiener. Cette attaque consiste a approximer ϕ(N) par N, en effet on a $N = pq$ et $\varphi(N) = N–(p + q) + 1$, p et q étant très grands, on peut plus ou moins négliger $(p + q) + 1$ devant pq. Or on sait qu'il existe k tel que $ed = k\varphi(N) + 1$ puisque $ed \equiv 1[\varphi(N)]$.

On a donc $\frac{e}{\varphi(N)} - \frac{k}{d} = \frac{1}{d\varphi(N)}$: et donc $\frac{e}{N} - \frac{k}{d} = \frac{1}{dN}$ avec l'approximation qu'on s'est permise. $\frac{e}{N}$ et $\frac{k}{d}$ sont donc semblerait il très proches. On va donc tenter d'obtenir k et d en s'approchant de plus en plus précisément de $\frac{e}{N}$. Pour cela on va utiliser les fractions continues.

On utilise l'algorithme d'Euclide pour récupérer une suite de coefficients successifs afin de construire notre fraction continue.

Enfin on construit notre fraction et on essaye de construire une clé avec son dénominateur comme étant d. Si ça ne fonctionne pas, on doit ajouter un étage a notre fraction et retenter… Par exemple pour un n = 53 et e = 17

e/n : $e = 0 * n + 17$, la fraction est : 0 et on a alors d = 0, on essaye de générer une clef mais ça na fonctionne pas, on itère ;

n/17 : $n = 3 * 17 + 2$, la fraction est : $0 + \frac{1}{3}$ et on a alors d = 3, on essaye de générer une clef mais ça na fonctionne pas, on itère ;

17/2 : $17 = 8 * 2 + 1$, la fraction est : $0 + \frac{1}{3+\frac{1}{8}} = \frac{8}{25}$ et on a alors d = 25, on essaye de générer une clef mais ça na fonctionne pas, on itère ; et cætera et cætera.

L'implémentation

On implémente le tout en python et le résultat est le suivant :

```
from Crypto.PublicKey import RSA
from fractions import Fraction
n = <le N>
e = <le e>
doc = open("/home/thomas/Documents/tsp/Ctf/rsaextreme/rsaextreme", "w")

def f(a, b, c, T):
    if c == 0:
        return(T)
    temp = a%b
    T.append(a/b)
    return f(b, temp, c-1, T)

def frac(res, T):
    if len(T) == 1:
        return Fraction(1, res) + T[0]
    return frac(Fraction(1, res) + T.pop(), T)

def final(a):
    T = f(e, n, a, [])
    for i in range(2, a):
        N = T[:i]
        d = (frac(N.pop(), N)).denominator
        print(d)
        try:
            key = RSA.construct((n,e,d))
            print("Youpi")
            doc.write(key.exportKey())
            return
        except ValueError: None

final(500)

doc.close()
```

C’est déjà fini ;(

On obtient alors un magnifique « Youpi » sur le terminal après quelques minutes et une superbe clé privé dans notre fichier rsaextreme. Il ne reste plus qu’a l’envoyer au serveur avec la commande :

```
ssh -i rsaextreme rsaextreme@157.159.40.163
```

et l’afficher falg.txt avec la commande :

```
cat flag.txt
```

Et voila, on obtient le flag.