

### Descripción:

implemente RSA de la mejor manera posible, seguir todas las instrucciones y buenas prácticas ;) <https://es.wikipedia.org/wiki/RSA>

aunque... algo no funciona, no puedo descifrar los mensajes que cifro, algo anda mal podrías ayudarme a descifrar mi bandera?

Esto no puede ser tan fácil o ¿sí?

creo que alguien de una compañía importante ya había encontrado algo parecido.

### Writeup:

Revisando el código fuente del reto debería ser un poco obvio la mala implementación de la generación de llaves. Una condición importante en RSA es que el valor de la función de Euler del modulo debe ser coprimo al exponente público, en otras palabras, que el máximo común divisor entre  **$\phi(n)$**  y  **$e$**  sea igual a 1. Cuando esta condición no se cumple simplemente no se podrá recuperar la llave privada incluso teniendo los valores factorizados de N.

```
e=getPrime(7)
while(1):
    p=getPrime(512)
    q=getPrime(512)
    if(GCD((p-1)*(q-1),e)!=1):
        break
pt=bytes_to_long(FLAG)
ct=pow(pt,e,p*q)
```

A pesar de poseer los dos factores primos **P** y **Q** no es posible recuperar la llave privada ya que esta no cumplirá con la propiedad necesaria para invertir la exponenciación con el exponente público.

Esta vulnerabilidad o “bug” en la generación de llaves RSA fue descubierta por Microsoft y esta afectaba a ciertas versiones de Windows 10. Los detalles de esta vulnerabilidad y los algoritmos para recuperar un texto claro cifrado pueden ser consultados en esta publicación oficial: <https://eprint.iacr.org/2020/1059.pdf>

El autor igual tiene sus propias implementaciones, y pueden ser encontradas en su repositorio oficial: <https://github.com/danshumow-msft/FixBadRsaEncryption>


Este reto pude ser resuelto usando implementaciones propias o reutilizando los scripts del propio autor.

Con los scripts se puede resolver el reto de forma relativamente facil solo se requiere realizar un par de modificaciones para que se encuentre la flag:

```

print("Searching for plaintext...")
i = 1
ell = g_e_torsion
while (i < e):
    pt_hat = ell*z % N
    if (b"hackdef" in long_to_bytes(pt_hat)):
        print("plaintext found.")
        break
    i = i + 1
    ell = ell*g_e_torsion % N
print("Plaintext search finished")
print(long_to_bytes(pt_hat))

```



Ejecutando el script modificado de RSAMath se puede recuperar la flag.

```

kd3n4@DESKTOP-LV9K32B:/mnt/c/Users/night/Desktop/hackdef2022/FixBadRsaEncryption-master/script$ python3.7 RSAMath.py
6560740653527128034613788488518496844419045680402645229733551369763820564928603475288931066034331131594984039360442422116309895082008
5952152022314778310701187163977300713159952449390004576127682138511583490947921515788321955744
8332140629979452603959511380418490992412188014111359441761610239600052117459326413616942453863600537125629729987761876087713566754150
8559233068330768454590507698251717190571313961072530581168215631590971103350386032505116888379488
6.560740653527128e+305
(-17, 8782093788185919416412157819276728059458565084003540858698454589447633827069784179520616387605010176150766036939174895746241591
1324189069809793649703250544896203749717418297001508973465179481658239013361569654446187152767505887, 1)
112
Searching for good generator...
Good generator found in
2
non crt private key operations done
crt private key operations done
g of e torsion group = 81277541089491960174808257687597082149356160511894047412700129440142739224181324473267374071670521984136320315
321221870038861957674177481976155187933515751557844563623529850736124615416094734221275499478522012759332203070285740317
Searching for plaintext...
plaintext found.
Plaintext search finished
b'hackdef{M1cR0Soft_Re$3aRch1nG_vU1Ns_1n_CRyPt0}'

```

La dificultad del reto esta en identificar la mala implementación de RSA en el código fuente, una vez identificada es necesario investigar este bug para llegar a la publicación oficial de Microsoft donde está la teoría necesaria para entender como usar los scripts del autor.