Mind your Ps and Qs

Link To Solve the Labs:- https://play.picoctf.org/practice/challenge/162?category=2&page=1

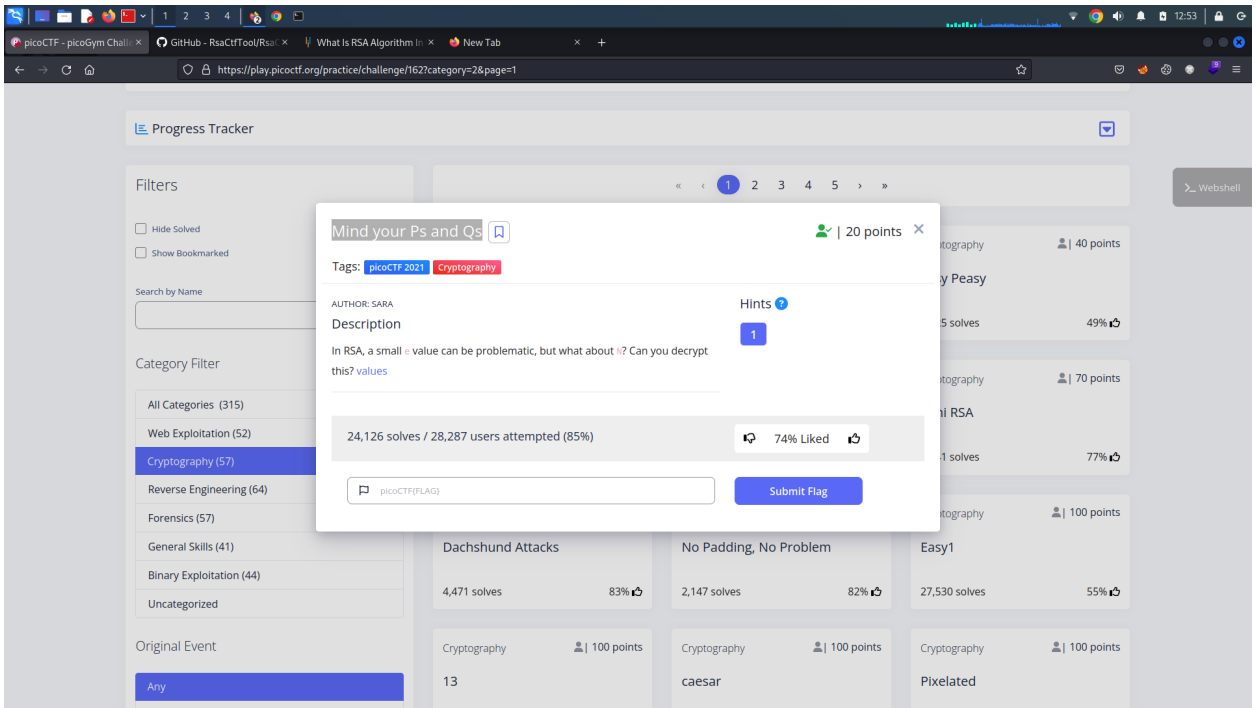
If you don't know about RSA then Below is the link to Short and Sweet tutorial you can refer to understand RSA

https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_understanding_rsa_algorithm.htm

Hello Everyone,

So Today we are Solving the Lab in the 'Cryptography' Category

All the things we are doing are in the Kali Linux.



As you can see the Screenshot of the Description

Description

In RSA, a small e `value` can be problematic, but what about N? Can you decrypt this? values

Thus, the it is clear from the description that we have to decrypt the cypher that is provided in the "Values" file.

Let's examine the "Values" first.

```
┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs]
└─# cat values
Decrypt my super sick RSA:
c: 861270243527190895777142537838333832920579264010533029282104230006461420086153423
n: 1311097532562595991877980619849724600678416443010544132789735880011688905776341342 3
e: 65537
```

As seen in the screenshot up top, we've provided the three values.

C :- Cipher(Encrypted Message)
n :- Product of Two Prime Numbers used to Calculate Modules
e :- Public Exponent

Now that we know of that, we can focus on finding M(Decrypted Message) Means to Decrpt Cypher.

So Let's Start

https://github.com/RsaCtfTool/RsaCtfTool

So, in this case, we'll use the Python-written RsaCtfTool.

Python Must Be Installed on Your Computer or Laptop in Order to Run this tool

Command to Clone The Repository

git clone https://github.com/RsaCtfTool/RsaCtfTool.git

You can see from the List Directory that the "RsaCtfTool" Directory is present.

```
┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs]
└─# git clone https://github.com/RsaCtfTool/RsaCtfTool.git
Cloning into 'RsaCtfTool'...
remote: Enumerating objects: 4581, done.
remote: Counting objects: 100% (112/112), done.
remote: Compressing objects: 100% (66/66), done.
Receiving objects:  39% (1814/4581), 2.43 MiB | 2.41 MiB/s
remote: Total 4581 (delta 70), reused 76 (delta 46), pack-reused 4469
Receiving objects: 100% (4581/4581), 16.52 MiB | 3.64 MiB/s, done.
Resolving deltas: 100% (3150/3150), done.

┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs]
└─#

┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs]
└─# ls
RsaCtfTool  values

┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs]
└─# cd RsaCtfTool

┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs/RsaCtfTool]
└─# ls
attacks       CONTRIBUTING.md  Dockerfile_full  lib         optional-requirements.txt  requirements.txt  sage      test.sh
CHANGELOG.md  Dockerfile       examples         LICENSE.md  README.md                  RsaCtfTool.py     setup.py
```

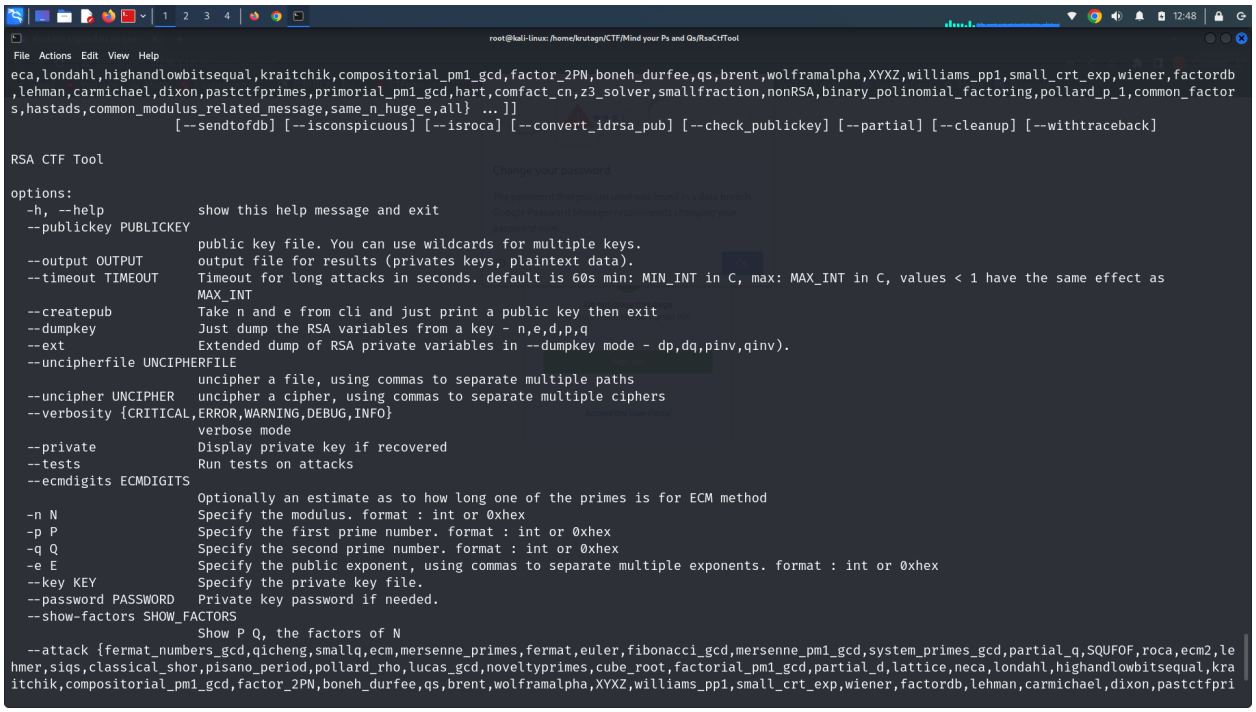
Let's use the "CD RsaCtfTool" command to change directories to that directory.

We're going to use a Python file called "RsaCtfTool.py" to decipher the cypher, as you can see in the screenshot up top.

Let's learn how to use this tool, then. So let's check out the Tool's Help Menu.

Command to Display Help menu

Python RsaCtfTool.py –help



```
root@kali-linux: /home/krutagn/CTF/Mind your Ps and Qs/RsaCtfTool
File  Actions  Edit  View  Help
eca,londahl,highandlowbitsequal,kraitchik,compositorial_pm1_gcd,factor_2PN,boneh_durfee,qs,brent,wolframalpha,XYXZ,williams_pp1,small_crt_exp,wiener,factordb
,lehman,carmichael,dixon,pastctfprimes,primorial_pm1_gcd,hart,comfact_cn,z3_solver,smallfraction,nonRSA,binary_polinomial_factoring,pollard_p_1,common_factor
s,hastads,common_modulus_related_message,same_n_huge_e,all} ...]]
                    [--sendtofdb] [--isconspicuous] [--isroca] [--convert_idrsa_pub] [--check_publickey] [--partial] [--cleanup] [--withtraceback]

RSA CTF Tool

options:
  -h, --help            show this help message and exit
  --publickey PUBLICKEY
                        public key file. You can use wildcards for multiple keys.
  --output OUTPUT       output file for results (privates keys, plaintext data).
  --timeout TIMEOUT     Timeout for long attacks in seconds. default is 60s min: MIN_INT in C, max: MAX_INT in C, values < 1 have the same effect as
                        MAX_INT
  --createpub           Take n and e from cli and just print a public key then exit
  --dumpkey             Just dump the RSA variables from a key - n,e,d,p,q
  --ext                 Extended dump of RSA private variables in --dumpkey mode - dp,dq,pinv,qinv).
  --uncipherfile UNCIPHERFILE
                        uncipher a file, using commas to separate multiple paths
  --uncipher UNCIPHER   uncipher a cipher, using commas to separate multiple ciphers
  --verbosity {CRITICAL,ERROR,WARNING,DEBUG,INFO}
                        verbose mode
  --private             Display private key if recovered
  --tests               Run tests on attacks
  --ecmdigits ECMDIGITS
                        Optionally an estimate as to how long one of the primes is for ECM method
  -n N                  Specify the modulus. format : int or 0xhex
  -p P                  Specify the first prime number. format : int or 0xhex
  -q Q                  Specify the second prime number. format : int or 0xhex
  -e E                  Specify the public exponent, using commas to separate multiple exponents. format : int or 0xhex
  --key KEY             Specify the private key file.
  --password PASSWORD   Private key password if needed.
  --show-factors SHOW_FACTORS
                        Show P Q, the factors of N
  --attack {fermat_numbers_gcd,qicheng,smallq,ecm,mersenne_primes,fermat,euler,fibonacci_gcd,mersenne_pm1_gcd,system_primes_gcd,partial_q,SQUFOF,roca,ecm2,le
hmer,siqs,classical_shor,pisano_period,pollard_rho,lucas_gcd,noveltyprimes,cube_root,factorial_pm1_gcd,partial_d,lattice,neca,londahl,highandlowbitsequal,kra
itchik,compositorial_pm1_gcd,factor_2PN,boneh_durfee,qs,brent,wolframalpha,XYXZ,williams_pp1,small_crt_exp,wiener,factordb,lehman,carmichael,dixon,pastctfpri
```

The Help Menu is shown in the screenshot above. So now Look into it

--uncipher :- uncipher a cipher
-n :- Specify the modulus.
-e :- Specify the public exponent

The three flags mentioned above can help us solve this CTF.

Let's issue a command to finish this lab.

python RsaCtfTool.py --uncipher 861270243527190895777142537838333832920579264010533029282104230006461420086153423 -n

1311097532562595991877980619849724606784164430105441327897358800116889057763413423 -e 65537

```
root@kali-linux: /home/krutagn/CTF/Mind your Ps and Qs/RsaCtfTool
File  Actions  Edit  View  Help
  --withtraceback        show tracebacks

┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs/RsaCtfTool]
└─# python RsaCtfTool.py --uncipher 861270243527190895777142537838333832920579264010533029282104230006461420086153423 -n 1311097532562595991877980619849724606784164430105441327897358800116889057763413423 -e 65537
private argument is not set, the private key will not be displayed, even if recovered.

[*] Testing key /tmp/tmpgi9b56uj.
attack initialized...
attack initialized...
[*] Performing smallq attack on /tmp/tmpgi9b56uj.
[+] Time elapsed: 0.3734 sec.
[*] Performing mersenne_primes attack on /tmp/tmpgi9b56uj.
 24%|                                                                                                  | 12/51 [00:00<00:00, 154391.56it/s]
[+] Time elapsed: 0.0056 sec.
[*] Performing fibonacci_gcd attack on /tmp/tmpgi9b56uj.
100%|                                                                                                  | 9999/9999 [00:00<00:00, 201276.83it/s]
[+] Time elapsed: 0.0501 sec.
[*] Performing system_primes_gcd attack on /tmp/tmpgi9b56uj.
100%|                                                                                                  | 7007/7007 [00:00<00:00, 919283.33it/s]
[+] Time elapsed: 0.0360 sec.
[*] Performing lucas_gcd attack on /tmp/tmpgi9b56uj.
100%|                                                                                                  | 9999/9999 [00:00<00:00, 202079.86it/s]
[+] Time elapsed: 0.0502 sec.
[*] Performing factordb attack on /tmp/tmpgi9b56uj.
[*] Attack success with factordb method !
[+] Total time elapsed min,max,avg: 0.0056/0.3734/0.1031 sec.

Results for /tmp/tmpgi9b56uj:

Uncipered data :
HEX : 0x007069636f4354467b736d6131315f4e5f6e305f67306f645f31333638363637397d
INT (big endian) : 1301638252944910606592729142534253543799622213535290525663957395900284941573977 3
INT (little endian) : 37119719776712686220408522365100361254955019426847706732211053811485132026256711 68
utf-8 : picoCTF{sma11_N_n0_g0od_13686679}
utf-16 : 灣捩飛福獻慦ㄱ也湟弸で擘ㄔ聖堊孇紹
STR : b'\x00picoCTF{sma11_N_n0_g0od_13686679}'

┌──(root㉿kali-linux)-[/home/krutagn/CTF/Mind your Ps and Qs/RsaCtfTool]
└─#
```

We finally executed the above command and located the flag.

picoCTF{sma11_N_n0_g0od_13686679}