Picoctf--EVEN RSA CAN BE BROKEN??

Tuesday, April 15, 2025 3:18 PM

EVEN RSA CAN BE BROKEN???

Easy | Cryptography | picoCTF 2025 | browser_webshell_solvable

AUTHOR: MICHAEL CROTTY

Description

This service provides you an encrypted flag. Can you decrypt it with just N & e?

Connect to the program with netcat:

```
$ nc verbal-sleep.picoctf.net 51434
```

The program's source code can be downloaded here.

Hints

1 2 3

4,992 users solved

96% Liked

picoCTF{FLAG}

Submit Flag

```python
from sys import exit
from Crypto.Util.number import bytes_to_long, inverse
from setup import get_primes

e = 65537

def gen_key(k):
    """
    Generates RSA key with k bits
    """
    p,q = get_primes(k//2)
    N = p*q
    d = inverse(e, (p-1)*(q-1))

    return ((N,e), d)

def encrypt(pubkey, m):
    N,e = pubkey
    return pow(bytes_to_long(m.encode('utf-8')), e, N)

def main(flag):
    pubkey, _privkey = gen_key(1024)
    encrypted = encrypt(pubkey, flag)
    return (pubkey[0], encrypted)

if __name__ == "__main__":
    flag = open('flag.txt', 'r').read()
    flag = flag.strip()
    N, cypher  = main(flag)
    print("N:", N)
    print("e:", e)
    print("cyphertext:", cypher)
    exit()
```

p,q = get_primes (K // 2) Randomly generate two larger primes N=p,q; (All the values of N in this question are even when they are even, so it means that p or q is equal to 2. Since 2 is the only even number among the primes, and odd x odd = odd, it can be surmised to this conclusion)

```
┌──(kali㉿kali)-[~]
└─$ nc verbal-sleep.picoctf.net 51434
N: 21424324791670340355352740825912615881605322443460040025043126551906393363303103185581319785933572679653741350446541897985020514448755590607345511433014646
e: 65537
cyphertext: 10019856636187731162105078999763519512539281149116456595171426082633911256198170832258970683707272159712627778579275607070156834701854134198920984991902423

┌──(kali㉿kali)-[~]
└─$ nc verbal-sleep.picoctf.net 51434
N: 21803059385460817701968383374899887189795369018003478354633242497081604016622279160407748618355862232606622345932431126665321771430538325250791953223318338
e: 65537
cyphertext: 30088427504034869333057459603034108954692356866854430416236477536342068960559263648318705628785034481248130463323136591180802792435855018331861810260121
3

┌──(kali㉿kali)-[~]
└─$ nc verbal-sleep.picoctf.net 51434
N: 14568227043298167940400651179839477974643661710541459208121007233047455797508987719922662705700262265962116899428204455621971184778141708379201465657511714
e: 65537
cyphertext: 3885857458380411817194414585886367716055451396438446005451750443745793474271246329395705268333138818169162445966958107645504780470337306121131198856175173

┌──(kali㉿kali)-[~]
└─$ nc verbal-sleep.picoctf.net 51434
N: 26123071438080057417943657551736225678122533965499715342999273681317323173227965597990617946808128457575867422829653630405115544344629882050541801621608038
e: 65537
cyphertext: 19382205836835081904559460884934431065449340363922644884524774122536726469895534284968201200217685028287515946692160884659929359510703737131406000425972429 3

┌──(kali㉿kali)-[~]
└─$ nc verbal-sleep.picoctf.net 51434
N: 23786005384009216294123357648018561250071005356304022172567092128738093533008770099207720954213043585132346885592264490376509349990478998516199246565955822
e: 65537
cyphertext: 76954314398481783355667104472320502722430300125961292243041570876637302076214658662847202638540814269948540012555031172044647007526718979563981479742509 0 7
```

```
┌──(kali㉿kali)-[~]
└─$ nc verbal-sleep.picoctf.net 51434
N: 23374580624058402508368913754681660318228053806430425571827814738125271297471024660017545751173212932095053401432368768052568585379774998797738062907373862
e: 65537
cyphertext: 5810617335745653986553904987366193879975687360545866419956812346762371789594021973503228442695884675445371826795154840320824221891071123299970659895375397
```

So divide the value of N by 2 get another value

116872903120292012541844568773408301591140269032152127859139073690626356487355123300087728755866064660475267007161843840262842926898874993988690314536869 31

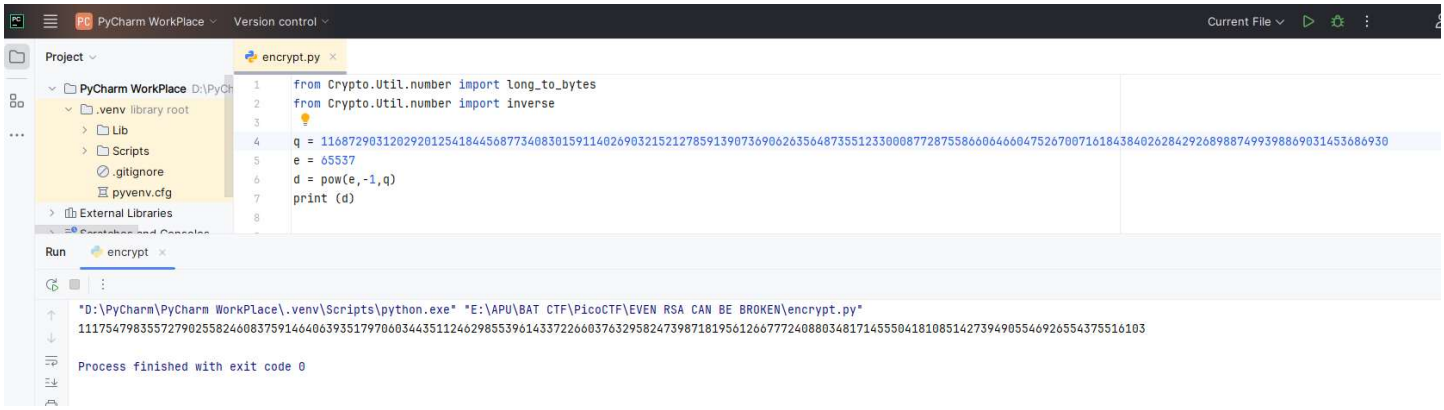
Analysing d = inverse(e,(p-1)*(q-1))
This code means to find an integer d such that (e * d) % ((p-1)*(q-1)) == 1.

So now we currently know that p = 2 p-1 = 1
q =
116872903120292012541844568773408301591140269032152127859139073690626356487355123300087728755866064660475267007161843840262842926898874993988690314536869 31

q-1=

116872903120292012541844568773408301591140269032152127859139073690626356487355123300087728755866064660475267007161843840262842926898874993988690314536869 30
604752670071618438402628429268988749939886903145368693 0

So the current equation is equivalent to d = inverse(e , q-1) i.e. find an integer d such that (e * d) %(q-1) == 1 holds
So now we can invert the value of d



The pow( ) function allows you to find the value of d. pow(e,-1,q) means to find a value of d such that (e * d) % q == 1 holds, i.e., to find the multiplicative inverse of e in the sense of modulo q, assigned to d. The value of q in the above figure is the value of q-1.
So d =
111754798355727902558246083759146406393517970603443511246298553961433722660376329582473987181956126677724088034817145550418108514273949055469265543755161 03
Now we know that the public key is (e , N) and the private key is (d, N) So we can get the hidden message.

Message = 303061272237661901533925185220017414319816026711922582267634257213580 5