# BUUCTF Crypto练习

原创

abtgu 于 2020-09-20 16:35:52 发布 

分类专栏： CTF



本文链接：https://blog.csdn.net/weixin_43790779/article/details/108694938





## Writeup

RSAROLL

rsa2

[GUET-CTF2019]BabyRSA

[BJDCTF 2nd]rsa1

[NCTF2019]childRSA

[HDCTF2019]bbbbbbrsa

[BJDCTF2020]easyrsa

## RSAROLL

打开data文件，猜测第一行{920139713,19}，为{n,e}，其余行为rsa加密后的密文，写脚本直接解密

```
import gmpy2

n = 920139713
e = 19
p = 18443
q = 49891

d = gmpy2.invert(e,(p-1)*(q-1))
m = []
with open('data.txt') as f:
    for line in f.readlines():
        line = line.strip('\n')
        m.append(chr(pow(int(line),d,n)))

for i in m:
    print(i,end="")
```

得到flag，flag{13212je2ue28fy71w8u87y31r78eu1e2}。

## rsa2

本题考查低解密指数攻击，直接使用破解脚本求出d值

```
import RSAwienerHacker
N = 10199180977755325347027675139926474013115768232925267350179215450700615843443200914199536724196252570595004625340018888465826249653470643879151507188586089755273665689956691573129722581725063987364337631010399217064690655724283289391490205358108750251278730332274778042021088485216658671763655905815254497947 1
e = 46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378630543338219025744820916471913696072050291990620486581719410354385121760761374229374847695148230596005409978383693697403058160827702839096119563559721818480775199209220592683769588117133651069252352182651730 85
d = RSAwienerHacker.hack_RSA(e,N)

import hashlibs
flag = "flag{" + hashlib.md5(hex(d).encode('utf8')).hexdigest() + "}"
print(flag)
```

得到flag，flag{8159e6c4abdd3b94ce461ed9a1a24017}。然而，结果错误，想到python2与python3的md5结果有区别，故使用python2的在线环境运行，果然得到不同结果，flag{47bf28da384590448e0b0d23909a25a4}。

## [GUET-CTF2019]BabyRSA

给了(p+q)、(p+1)(q+1)，e，d，c，要想求m，需要知道n。将(p+1)(q+1)展开，可以发现

$$(p+1)(q+1)=$$

```
import libnum
import gmpy2

#(p+q)
x = 0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97be4400e33307573af6b25e2ea
#(p+1)(q+1)
y = 0x5248becef1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c626547203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec6605d9693078b800ee80931a7a0c84f33c851740
e = 0xe6b1bee47bd63f615c7d0a43c529d219
d = 0x2dde7fbaed477f6d62838d55b0d0964868cf6efb2c282a5f13e6008ce7317a24cb57aec49ef0d738919f47cdcd9677cd52ac2293ec5938aa198f962678b5cd0da344453f521a69b2ac03647cdd8339f4e38cec452d54e60698833d67f9315c02ddaa4c79ebaa902c605d7bda32ce970541b2d9a17d62b52df813b2fb0c5ab1a5
c = 0x50ae00623211ba6089ddfae21e204ab616f6c9d294e913550af3d66e85d0c0693ed53ed55c46d8cca1d7c2ad44839030df26b70f22a8567171a759b76fe5f07b3c5a6ec89117ed0a36c0950956b9cde880c575737f779143f921d745ac3bb0e379c05d9a3cc6bf0bea8aa91e4d5e752c7eb46b2e023edbc07d24a7c460a34a9a

n = y-x-1

m = gmpy2.powmod(c,d,n)
print(libnum.n2s(m))
```

## [BJDCTF 2nd]rsa1

给了e，c，(p-q)，($p^{2+q}2$)，要想求m，需要知道d和n。想到完全平方公式

$$(p-q)=p+q-2$$

```
import libnum
import gmpy2


e = 11936369
#p^2+q^2
x = 179078542063786171098570462583697203005792437399320455031433804285072395735579228946278501258032028541721083458411587159216647095906724535882854022692256931873918358743337544481893000613838499804988741946438301395627666863836745379213775039519799936282909933735085265938756037672822546212884981890622952397362
#p-q
y = 315727336846491218073160616031292472134662539273203814345627816025271103682524724202358487297625197899678333679875278389569117239668699651730001857141018 2

n = (x-y*y)//2
#p+q
z = gmpy2.iroot(x+2*n,2)[0]
phi = n+1-z
d = gmpy2.invert(e,phi)

c = 660989534543580069616201086306167733735190557305709608391040978282999872811856929771160935621421296018858359804496208885363036003627386875682295080286859786115268784555632401361368726219177869631474737282581754953281455446292680952288463729728457830067772744413208401833835237946643154439728046271941633683 79


m = gmpy2.powmod(c,d,n)
print(libnum.n2s(m))
```

## [NCTF2019]childRSA

直接在线分解n，得到p，q的值，脚本如下：

```
import gmpy2
import libnum

e = 0x10001
p = 178449493212694205742332078583256205058672290603652616240227340638730811945224947826121772642204629335108873832781921390308501763661154638696935732709724016546955977529088135995838497476350749621442719690722226913635772410880516639651363626821442456779009699333452616953193799328647446968707045304702547915799734431818800374360377292309248361548868909066895474518333089446581763425755389837072166970684877011663234978631869703859541876049132713490090720408351108387971577438951727337962368478059295446047962510687695047494480605473377173021467764495541590394732685140829152761532035790187269724703444386838656193674253139
q = 184084121540115307597161367011014142898823526027674354555037785878481711602257307508985022577801782788769786800015984410443717799994642236194840684557538917849420967360121509675348296203886340264385224150964642958965438801864306187503790100281099130863977710204660546799128755418521327290719635075221585824217487386227004673527292281536221958961760681032293340099395863194031788435142296085219594866635192464353365034089592414809332183882423461536123972873871477755949082223830049594561329457349537703926325152949582123419049073013144325689632055433283354999265193117288252918515308767016885678802217366700376654365502867
c = 263080183567398538953822401099688941751667312837029270021652689987737083352163389970583141577171471310832965513133340425098062298533414884610870099552038542533138276082754605927856077390919925914310803426640819620305570427848640745333807010145853156632187831301623761760947730104781593624343317872793033027180987355746054698038018731099824732582074443423306331918490405535507088865933407707530643224108890481354250257159821966006507409870764865406740909231816642815151976797459078301076847772485322786453437162636860149410814179146227249063149602499451050113017312473246016208867829672173393403938536164500771051253919826899861783424172233922170852764654711027375947199323472424826703208010631918694713183135144079973263500651879041542295577063513550524460271599725467372134514229782110557781645787821564284666268940261030533604312816446455151554713018268447543388023528460952934217182498197282055385346522129848312836424720716694948518231235528273807377986098297062257443766670825340268744834824831274915334743065522100393862560621163457858706683315137257920533021882766825506726633539377810556218601016242422166716358243114127934959656288760363447317331427594953482489703136553814072414571187435323113946977632836818529085643872826052791085
phi = (p-1)*(q-1)

d = gmpy2.invert(e,phi)
m = pow(c, d, n)
print(libnum.n2s(m))
```

## [HDCTF2019]bbbbbbrsa

直接写脚本破解：

```
from base64 import b64decode as b32decode
from gmpy2 import invert,gcd,iroot
from Crypto.Util.number import *

p = 177077389675257695042507998165006460849
n = 37421829509887796274897162249367329400988647145613325367337968063341372726061
cipher = '==gMzYDNzIjMxUTNyIzNzIjMyYTM4MDM0gTMwEjNzgTM2UTN4cjNwIjN2QzM5ADMwIDNyMTO4UzM2cTM5kDN2MTOyUTO5YDM0czM3MjM'
q = n//p

c = int(b32decode(cipher[::-1]))
print(c)

phi = (p-1)*(q-1)

for i in range(50000,70000):
    if gcd(i,phi) == 1:
        try:
            d = invert(i,phi)
            m = pow(c,d,n)
            print(long_to_bytes(m).decode())
        except:
            continue
```

## [BJDCTF2020]easyrsa

Fraction(a,b) 相当于 a/b，关于Fraction(a,b)详解请看https://blog.csdn.net/weixin_43790779/article/details/108693989 。
Derivative(f(x),x) : 当x='x'时,f(x)的导数值。
arctan(x)的导数是$1/(1+x^2)$。$\mathrm{arth}(x)$的导数是$1/(1-x^2)$

$$phi = (p-1)(q-1)$$

```
from Crypto.Util.number import long_to_bytes
import gmpy2

c = 79225478668577614598074915026542162830127761777895115493506729581018102813484022840983101477965494306892538035109948774201355372685494106526544796208586913241103671820256487884070415999430913862275431821577462029470995723896760843927064060843076570001046656966544091550063132039572928857437917151987819742055786547921231915849576652932083904537483691823331528098823124533597061478081989229167627737217266815889771038774541190437448891645293831880774991949329096439186966468769073273647513809531825178831345918108008489717191848087136943429854581030066760134519122210802527359489936926748993998260848486221458154610351
z = 32115748677623209667471622872185275070257924766015020072805267359839059393284316595882933372289732127274076434587519333300142473010344694803885168557548801202495933226215437763329280242113556524498457559562872900811602056944423967403777623306961880757613246328729616643032628964072931272085866928045973799374711846825157781056965164178505232524245809179235607571567174228822561697888645968559343608375331988097157145264357626738141646556353500994924115875748198318036296898604097000938272195903056733565880150540275369239637793975923329598716003350308259321436752579291000355560431542229699759955141152914708362494482
n = 15310745161336895413406690009324766200789179248896951942047235448901612351128459309145825547569298479821101249094161867207686537607047447968708758990950136380924747359052570549594098569970632854351825950729752563502284849263730127586382522703959893392329333760927637353052250274195821469023401443841395096410231843592101426591882573405934188675124326997277775238287928403743324297705151732524641213516306585297722190780088180705070359469719869343939106529204798285957516860774384001892777525916167743272419958572055332232056095979448155082465977781482598371994798871917514767508394730447974770329967681767625495394441

e=65537
#p+q
x = gmpy2.iroot(z+2*n,2)[0]
phi = n-x+1

d = gmpy2.invert(e,phi)
m = pow(c,d,n)
flag=long_to_bytes(m)

print(flag)
```