南邮ctf mysql_南邮ctf知识点汇总 - Crypto篇

原创

编辑部的宋姑娘 于 2021-02-05 07:35:28 发布 44 收藏
文章标签： 南邮ctf mysql

easy!:

bmN0Znt0aGlzX2lzX2Jhc2U2NF9lbmNvZGV9

base64解码获得flag

```
import base64

encoded = "bmN0Znt0aGlzX2lzX2Jhc2U2NF9lbmNvZGV9"

base64.b64decode(encoded)
```

Keyboard:

按顺序在键盘上比划一下获得flag。。。

异性相吸：

提示说xor，直接xor明文密文获得flag

```
#coding=utf-8

encrypted=[]

with open("./biubiubiu/密文.txt".decode('utf-8')) as f:

while True:

c = f.read(1)

if not c:

break

encrypted.append(c)

plain=[]

with open("./biubiubiu/明文.txt".decode('utf-8')) as f:

while True:

c = f.read(1)

if not c:

break

plain.append(c)
```

```
flag=""
for i in range(len(encrypted)):
flag+=chr(ord(encrypted[i])^ord(plain[i]))
print flag
```

Wiener Wiener Chicken Dinner:

RSA wiener attack:

```
import math
def continued_fractions_expansion(numerator,denominator):#(e,N)
result=[]
divident=numerator%denominator
quotient=numerator/denominator
result.append(quotient)
while divident!=0:
numerator=numerator-quotient*denominator
tmp=denominator
denominator=numerator
numerator=tmp
divident=numerator%denominator
quotient=numerator/denominator
result.append(quotient)
return result
def convergents(expansion):
convergents=[(expansion[0],1)]
for i in range(1,len(expansion)):
numerator=1
denominator=expansion[i]
for j in range(i-1,-1,-1):
numerator+=expansion[j]*denominator
if j==0:
break
tmp=denominator
```

```
denominator=numerator
numerator=tmp
convergents.append((numerator,denominator))#(k,d)
return convergents
def newtonSqrt(n):
approx = n/2
better = (approx + n/approx)/2
while better != approx:
approx = better
better = (approx + n/approx)/2
return approx
def wiener_attack(cons,e,N):
for cs in cons:
k,d=cs
if k==0:
continue
phi_N=(e*d-1)/k
#x**2-((N-phi_N)+1)*x+N=0
a=1
b=-((N-phi_N)+1)
c=N
delta = b*b - 4*a*c
if delta<=0:
continue
x1= (newtonSqrt(delta)-b)/(2*a)
x2=-(newtonSqrt(delta)+b)/(2*a)
if x1*x2==N:
return [x1,x2,k,d]
N=106304532128384446834453116899277852065119216210094853399153909744703144009006819183581
e=837165022918376318972691589160491375229372195625940137121740685432530132860541010174721
```

```
expansion=continued_fractions_expansion(e,N)

cons=convergents(expansion)

p,q,k,d=wiener_attack(cons,e,N)

print p

print q

print k

print d
```

获得p,q,k,d:

```
426460218401424758792499353633568921703212366165520538400311721231982037660757400558957 7

249271860636530815412836983376370530735257605974536868364048436665863131780986814967166 9

455969807867626572251143627963672752640210190604308383763232020961269569097 18

578997638017222610628912905035598359045719465572587611544225461048240946708 43
```

把d，p，q带入rsa，

```
privatekey = RSA.construct((long(n), long(e), long(d), long(p), long(q)))

from Crypto.PublicKey import RSA

from Crypto.Cipher import PKCS1_v1_5 as Cipher_pkcs1_v1_5

import base64

from Crypto.Hash import SHA

from Crypto import Random

privatekey=RSA.construct((106304532128384446834453116899277852065119216210094853399153909744
837165022918376318972691589160491375229372195625940137121740685432530132860541010174727 6

cipher = Cipher_pkcs1_v1_5.new(privatekey)

cipher_text =
'AGgt1h6dudnkeoCr7SFclkYYsYa65KZ8V29bbgbf+BDyjnyx5stCYjcyktat73aHs2EOaMgwGUwj3HwPTvT+T5L

flag=cipher.decrypt(base64.b64decode(cipher_text),837165022918376318972691589160491375229372195 6

print flag
```

获得flag

Baby RSA:

尝试分解n：

总结两种方式：

Search | Sequences | Report results | Factor tables | Status | Downloads | Login

1690370599734964619570437537694185541469152338771967999999999999 Factorize! (?)

Result:

status (?)	digits	number
FF	65 (show)	1690370599...99<65> = 157817387176484486971605217 1<28> · 1071092754719511397317504706621514626 9<38>

2. msieve分解：

msieve153.exe 0x291733BAB061EF9C599139CB3E40A5C762B6F448FFFFFFFFFFFFFF -v

```
Msieve v. 1.53 (SVN 1005)
Sun Dec 23 19:45:46 2018
random seeds: 4421a080 3902936c
factoring 1690370599734964619570437537694185541469152338771967999999999999999 (65 digits)
searching for 15-digit factors
commencing quadratic sieve (65-digit input)
using multiplier of 11
using generic 32kb sieve core
sieve interval: 12 blocks of size 32768
processing polynomials in batches of 17
using a sieve bound of 122849 (5838 primes)
using large prime bound of 6756695 (22 bits)
using trial factoring cutoff of 23 bits
polynomial 'A' values have 8 factors

sieving in progress (press Ctrl-C to pause)
6154 relations (2706 full + 3448 combined from 29596 partial), need 5934
6154 relations (2706 full + 3448 combined from 29596 partial), need 5934
sieving complete, commencing postprocessing
begin with 32302 relations
reduce to 9148 relations in 2 passes
attempting to read 9148 relations
recovered 9148 relations
recovered 7435 polynomials
attempting to build 6154 cycles
found 6154 cycles in 1 passes
distribution of cycle lengths:
   length 1 : 2706
   length 2 : 3448
largest cycle: 2 relations
matrix is 5838 x 6154 (0.8 MB) with weight 167383 (27.20/col)
sparse part has weight 167383 (27.20/col)
filtering completed in 3 passes
matrix is 5437 x 5501 (0.7 MB) with weight 146293 (26.59/col)
sparse part has weight 146293 (26.59/col)
commencing Lanczos iteration
memory use: 0.8 MB
lanczos halted after 88 iterations (dim = 5435)
recovered 64 nontrivial dependencies
p28 factor: 1578173871764844869716052171
p38 factor: 10710927547195113973175047066215146269
elapsed time 00:00:14
```


获得

p1=1578173871764844869716052171

p2=10710927547195113973175047066215146269

已知p1,p2,n,e,求d，并且解密获得flag：

import gmpy2

p1=157817387176484486971605217 1

p2=107109275471951139731750470662151462 69

n=0x291733BAB061EF9C599139CB3E40A5C762B6F448FFFFFFFFFFFFFF

e=0x10001

phi_n=(p1-1)*(p2-1)

d=gmpy2.invert(e,phi_n)

print hex(d)

c=0x237200C0F72B97DB55BA37C7AACBB61A26A0CB47D294726259C4DF

m=pow(c,d,n)

m_hex=hex(m)[2:]

m_str = str(bytearray.fromhex(m_hex))

print m_str

简单的方法：

rsa-tool 2 by te! 写入数据直接获得答案