2021年“绿城杯”网络安全大赛部分Writeup

原创

塞纳河畔的春水 于 2021-09-29 21:36:26 发布 871 收藏 5
分类专栏： CTF_Writeup 文章标签： 网络安全 算法 密码学 python pycharm

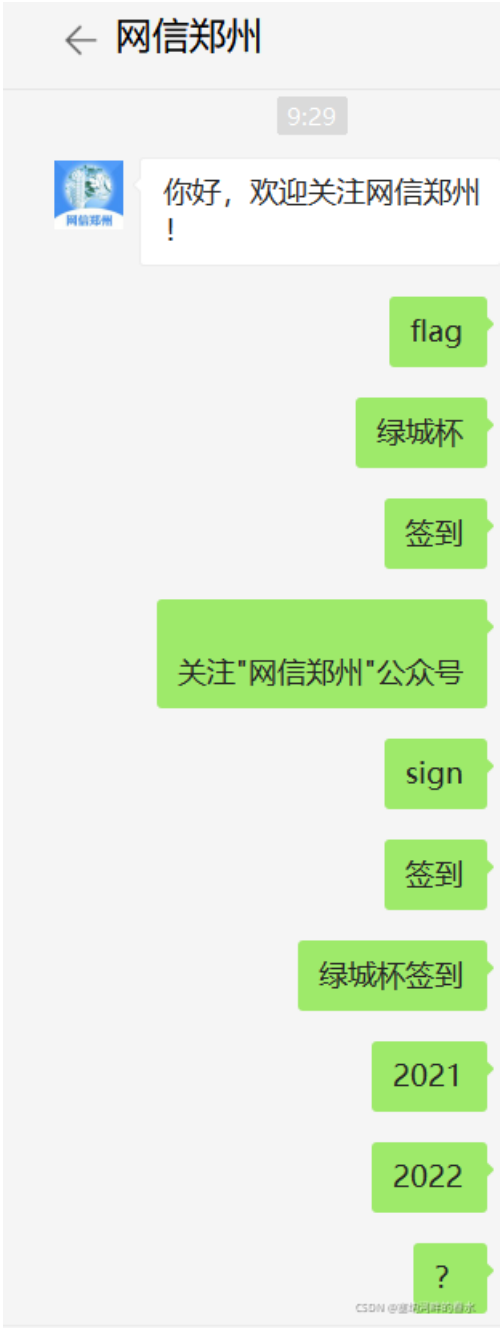
文章目录

MISC

签到

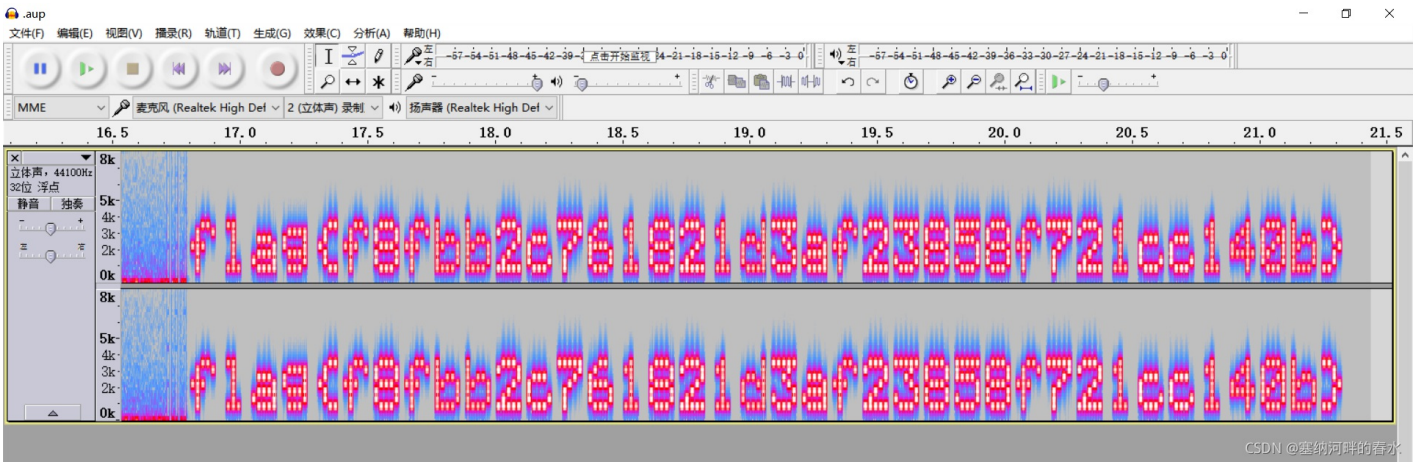
> 题目描述：关注“网信郑州”

虽然下架了但是还是想吐槽一下/doge



[warmup]音频隐写

手速题，题目给了一个附件标题音频隐写，直接用Audacity打开，导入数据，转频谱图，拖到最后。



接着手打或者OCR都可以

> flag{f8fbb2c761821d3af23858f721cc140b}

Reverse

[warmup]easy_re

先放个解题脚本

```
v16=[ 0xF5, 0x8C, 0x8D, 0xE4, 0x9F, 0xA5, 0x28, 0x65, 0x30, 0xF4,
0xEB, 0xD3, 0x24, 0xA9, 0x91, 0x1A, 0x6F, 0xD4, 0x6A, 0xD7,
0x0B, 0x8D, 0xE8, 0xB8, 0x83, 0x4A, 0x5A, 0x6E, 0xBE, 0xCB,
0xF4, 0x4B, 0x99, 0xD6, 0xE6, 0x54, 0x7A, 0x4F, 0x50, 0x14,
0xE5, 0xEC]
v23="tallmewhy"
v24=[]
v22=[]
for i in range(256):
    v24.append(i)
    v22.append(ord(v23[i%len(v23)]))
    v6,v7=0,0
while v6<255:
    v8=v24[v6]
    v7=(v7+v22[v6]+v8)%256
    v24[v6] = v24[v7]
    v6 += 1
    v24[v7]=v8^0x37
v12=0
v9,v10,v20=0,0,0
while v12+1<42:
    v9=(v9+1)%256
    v11=v24[v9]
    v10=(v10+v11)%256
    v24[v9]=v24[v10]
    v24[v10] = v11
    v12 = v20
    v16[v20]^=v24[(v11+v24[v9])&0xff]
    v16[v12]=chr(v16[v12])
    v20 = v12 + 1
print(''.join(v16))
```

> flag{c5e0f5f6-f79e-5b9b-988f-28f046117802}

Crypto

这次密码比较简单，AK。

RSA-1

题目附件：

```
from Crypto.Util.number import *
import gmpy2
from flag import flag
assert flag[:5]==b'flag{'

m = bytes_to_long(flag)
p = getPrime(1024)
q = getPrime(1024)
n = p * q
print('n =',n)
e = 0x10001
M = 2021 * m * 1001 * p
c = pow(M,e,n)
print('c =',c)

#n = 17365231154926348364478276872558492775911760603002394353723603461898405740234715001820111548600914907
#c = 69449671088154377354289412867841194031383197134557321559250559286465369625976729418058313121306893380
```

从题目条件可知n与c有公因子p，通过gcd求出p，随后求出q。

```
p=gmpy2.gcd(n, c)
q=n//p
```

接下去常规rsa解密，完整脚本。

```
import binascii
from Crypto.Util.number import *
import gmpy2

n = 17365231154926348364478276872558492775911760603002394353723603461898405740234715001820111548600914907610
c = 69449671088154377354289412867841194031383197134557321559250559286465369625976729418058313121306893380140
p = gmpy2.gcd(n, c)
#p=150290608270992439844054823303154263794197803561695786056860615174575181277160032222859532335454486914350
q = n//p

# 公钥e
e = 65537
# 两个素数的乘积n
n = p * q
phi = (p - 1) * (q - 1)
# 逆元d
d = gmpy2.invert(e, phi)
# 明文m
m = pow(c, d, n)
m = m//(2021 * 1001 * p)
print(binascii.unhexlify(hex(m)[2:].strip("L")))
#b'flag{Math_1s_1nterest1ng_hah}'
```

[warmup]加密算法

完整脚本。

```
str1 = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
cipher_text = 'aoxL{XaaHKP_tHgwpc_hN_ToXnnht}'
a = 37
b = 23
m = 52
for i in cipher_text:
    for j in range(33,128):
        if(chr(j)in str1):
            addr= str1.find(chr(j))
            cipher_text = str1[(a * addr + b) % m]
        else:
            cipher_text = chr(j)
        if(cipher_text==i):
            print(chr(j),end='')
```

RSA2-PLUS

题目附件：

```
from Crypto.Util.number import *
import gmpy2
from flag import flag
assert flag[:5]==b'flag{'

m1 = bytes_to_long(flag[:20])
p  = getPrime(512)
p1 = gmpy2.next_prime(p)
q  = getPrime(512)
q1 = gmpy2.next_prime(q)
n1 = p*q*p1*q1
print('n1 =',n1)
e = 0x10001
c1 = pow(m1,e,n1)
print('c1 =',c1)


m2 = bytes_to_long(flag[20:])
p2 = getPrime(1024)
q2 = getPrime(1024)
print('p2+q2 =',p2+q2)
print('q2*q2 =',p2*q2)
n2 = p2*p2*q2*q2*q2
print('n2 =',n2)
c2 = pow(m2,e,n2)
print('c2 =',c2)

#n1 = 6348779979606280884589422188738902470575876294643492831465947360363568026280963989291591157710389629
#c1 = 6201882078995455673376327652982610102807874783073703018551044780440620679217833227711395689114659144

#p2+q2 = 2747731467611384627081375823090973864377938917936913830338565243030108112941019334548244850105214
#p2*q2 = 1851472427003096217256696594172322438637407629423265225870108578101877617284335592056603515733157
#n2 = 4058822704559530408036038504108223850704429273134446581529603290563352555694378761071265167546081076
#c2 = 2559109016854482176174602417872466083959094819045132922748116857649071724229452073986560206108255875
```

Part1脚本

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

n = 6348779979606280884589422188738902470575876294643492831465947360363568026280963989291591157710389629216
c = 6201882078995455673376327652982610102807874783073703018551044780440620679217833227711395689114659144506
e = 0x10001

def fermat_factorization(n):
    factor_list = []
    get_context().precision = 2048
    sqrt_n = int(sqrt(n))
    c = sqrt_n
    while True:
        c += 1
        d_square = c**2 - n
        if is_square(d_square):
            d_square = mpz(d_square)
            get_context().precision = 2048
            d = int(sqrt(d_square))
            factor_list.append([c+d,c-d])
        if len(factor_list)==2:
            break
    return factor_list

factor_list = fermat_factorization(n)
[X1,Y1] = factor_list[0]
[X2,Y2] = factor_list[1]
assert X1*Y1 == n
assert X2*Y2 == n
p1 = gcd(X1,X2)
q1 = X1 // p1
p2 = gcd(Y1,Y2)
q2 = Y1 // p2

Fai = (p1-1)*(q1-1)*(p2-1)*(q2-1)
d = invert(e,Fai)
print long_to_bytes(pow(c,d,n))

#flag{Euler_funct1ons
```

Part2脚本

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

p2_add_q2 = 274773146761138462708137582309097386437793891793691383033856524303010811294101933454824485010521
p2_mul_q2 = 185147242700309621725669659417232243863740762942326522587010857810187761728433559205660351573311
n2 = 405882270455953040803603850410822385070442927313444658152960329056335255569437876107126516754608107687
c2 = 255910901685448217617460241787246608395909481904513292274811685764907172422945207398656020610825587597

q2 = n2 // (p2_mul_q2 * p2_mul_q2)
#print(q2)
p2 = p2_mul_q2 // q2
#print(p2)
phi = p2 * (p2 - 1) * q2 * q2 * (q2-1)
e = 0x10001
d = gmpy2.invert(e, phi)
m = pow(c2, d, n2)
print(long_to_bytes(m))
#b'_1s_very_interst1ng}'
```

合在一起

> flag{Euler_funct1ons_1s_very_interst1ng}