安恒月赛-dasctf 部分writeup

原创

逃课的小学生 于 2020-04-26 16:56:28 发布 2919 收藏 1
分类专栏： ctf crypto re

1.crypto-not rsa

题目很简单，r是一个随机数，g=n+1，已知(pow(g,m,n*n)*pow(r,n,n*n))%(n*n)=c，即密文c，求解明文m。我们根据二项式定理易知可将pow(g,m,n*n)=pow(n+1,m,n*n)=pow(m*n+1,n*n)。由于n=p*q，所以我们知道phi(n*n)=p*(p-1)*q*(q-1)。我们对加密公式两侧同时加(p-1)*(q-1)次幂，可化简为(pow((m*n+1),(p-1)*(q-1),n*n))%(n*n)=pow(c,(p-1)*(q-1),n*n)从而将r化简掉。我们再借助二项式定理化简可得((p-1)*(q-1)*m*n+1)%(n*n)=pow(c,(p-1)*(q-1),n*n)。((p-1)*(q-1)*m*n)%(n*n)=pow(c,(p-1)*(q-1),n*n)-1，((p-1)*(q-1)*m*n)-(pow(c,(p-1)*(q-1),n*n)-1)=k*n*n。这时我们发现(pow(c,(p-1)*(q-1),n*n)-1)%n=0。于是我们做进一步化简可得((p-1)*(q-1)*m)%n=(pow(c,(p-1)*(q-1),n*n)-1)/n，m=((pow(c,(p-1)*(q-1),n*n)-1)/n)乘以(p-1)*(q-1)模n的逆，可将m求解，具体代码如下：

```
import gmpy2
c=290889110547115092522156152310151629980425794259179144349623762434771767574480537226024226722517583320523
n=640101395461244581816550728987058004135856925881761328214285288196588479998894153591093966406850336730334
p= 800063369653457251577746180595049928418410402079982494166784357805777989378l9
q = 80006336965345725157774618059504992841841040207998249416678435780577798937447
assert p*q==n
phi=(p-1)*(q-1)
c1=pow(c,phi,n*n)-1
c2=c1/n
m=(c2*gmpy2.invert(phi,n))%n
print hex(m)[2:].decode("hex")
```

2.crypto-ComplexEncode

首先贴上题目

```
from Crypto.Util.number import *
from flag import FLAG,false_flag
import gmpy2
```

```
import random
import hashlib
import base64

def rsaEncode(msg):
    f=open("out","a")
    while True:
        ran=random.randint(3, 12)
        if isPrime(ran):
            break
    e=ran*getPrime(30)
    p = getPrime(1024)
    q = getPrime(1024)
    while (not gmpy2.gcd((p-1)*(q-1),e)==ran or p*q<pow(msg,ran)):
        p = getPrime(1024)
        q = getPrime(1024)
    n=p*q
    assert(pow(msg,ran)<n)
    print("rsaEncode_init_finish!")
    dsaEncode(p)
    f.write("rsaEncode n:"+hex(n)+"\n")
    f.write("rsaEncode e:"+hex(e)+"\n")
    c=pow(msg,e,n)
    f.write("rsaEncode flag is:"+hex(c)+"\n")
    f.close()

def rsaEncode2(m):
    m=int.from_bytes(m.encode(),'big')
    f=open("out","w")
    while True:
        ran=random.randint(20, 50)
        if isPrime(ran):
            break
    e=ran*getPrime(30)
    p = getPrime(1024)
    q = gmpy2.next_prime(p)
    while (not gmpy2.gcd((p-1)*(q-1),e)==ran or p*q>pow(m,ran)):
        p = getPrime(1024)
        q = gmpy2.next_prime(p)
    n=p*q
    assert(pow(m,ran)>n)
    c=pow(m,e,n)
    f.write("n2:"+hex(n)+"\n")
    f.write("e2:"+hex(e)+"\n")
    f.write("rsaEncode2 re is:"+hex(c)+"\n")
    f.close()
    print("rsaEncode2_finish!")
    return pow(m,ran,n)

def dsaEncode(p):
    key=genkey(p)
    (r,s,k,q)=sign(rsaEncode2(false_flag),key)
    sig= r.to_bytes(205, 'big') + s.to_bytes(205, 'big') + k.to_bytes(205, 'big')+ q.to_bytes(205, 'big')
    f=open("out","a")
    f.write("dsaEncode :"+base64.b64encode(sig).decode()+"\n")
    f.close()

def genkey(x):
    # DSA
    N=1024
```

```
    N=1024
    L=2048-N-1
    while True:
        q = getPrime(N)
        if q-1>x:
            break
    assert(q-1>x)
    while True:
        t = random.getrandbits(L)
        p = (t * 2*q + 1)
        if isPrime(p):
            break
    e = (p-1) // q
    g = pow(2, e, p)
    y = pow(g, x, p)
    print("genkey_finish!")
    return {'y':y, 'g':g, 'p':p, 'q':q, 'x':x}

def sign(m, key):
    g, p, q, x = key['g'], key['p'], key['q'], key['x']
    k = random.randint(1, q-1)
    Hm = int.from_bytes(hashlib.md5(str(m).encode('utf-8')).hexdigest().encode(), 'big')
    r = pow(g, k, p) % q
    s = (inverse(k, q) * (Hm + x*r)) % q
    return (r, s, k, q)

if __name__ == "__main__":
    msg=int.from_bytes(FLAG.encode(),'big')
    rsaEncode(msg)
```

阅读题目我们发现RSA1中的p就是dsa的密钥x，而在dsa中我们易知$s \equiv (H(m)+xr)k^{-1} mod q$,而我们从base64的密文已知(r,s,k,q)。而我们在RSA2可以容易的解出m。于是我们倒着去做，十分容易解出原文。这里注意在RSA解密中e和phi不互质，所以我们要注意先求解$m^{gcd(e,phi)}$，再解m。在dsa中直接是对RSA2中的$m^{gcd(e,phi)}$做hash，所以不需要求解，而在gcd(e,phi)较小，我们可以爆破求解m。同时在RSA2中n可以使用yafu轻松分解。代码如下

```
import gmpy2
import base64
import hashlib
n2=0x431a834246a5969d460cee6b7db01ec4e05daffd60d6674fd1cf3ab96544ad788df5ef729eba08fce3d6c237b47b7cbda093fb
e2=0x57e6b0c63
rsaEncode2=0x174e74a04d09ee859030c9fa292c266f9de06bf833dcffc5d24a4382251620cac743cf2d500428d5784fc271e61096
dsaEncode="AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
n=0x769bf99a95e9f446788300bfc679b04f26498d26ce1f1384ef22a4bc0606d2fdcf4af86b8e9b26003f561532613a8b7bcb09c00
e=0x12a316381
flag=0xa97f8fd2cbf4aa1cae331dc5261c93a8da3c0a4aadd33cf204cdb2e38e3afdd16aed1a23cd911f96ddc7152b14120949235c
p2 = 92038254802992589334836021211021722416892343074988502820738425960842862985591620926198422759102071572 8
q2 = 92038254802992589334836021211021722416892343074988502820738425960842862985591620926198422759102071572 8
assert p2*q2==n2
phi=(p2-1)*(q2-1)
g2=gmpy2.gcd(e2,phi)
d2=gmpy2.invert(e2/g2,phi)
m2=pow(rsaEncode2,d2,n2)
Hm=int(hashlib.md5(str(m2).encode()).hexdigest().encode("hex"),16)
key=base64.b64decode(dsaEncode)
r=int(key[:205].encode("hex"),16)
s=int(key[205:410].encode("hex"),16)
k=int(key[410:615].encode("hex"),16)
q=int(key[615:820].encode("hex"),16)
x=(gmpy2.invert(r,q)*(k*s-Hm))%q
p=n/x
phi=(p-1)*(x-1)
g=gmpy2.gcd(e,phi)
d=gmpy2.invert(e/g,phi)
m=pow(flag,d,n)
for i in xrange(1000000):
 ss=gmpy2.iroot(i*n+m,g)
 if ss[1]:
  print ss[0]
  print hex(ss[0])[2:].decode("hex")
  break
```

reverse-入门逆向

```
 7 _alloca(0x10u);
 8 __main();
 9 std::operator<<<std::char_traits<char>>((int)&std::cout, "input your key: ");
10 std::operator>><char,std::char_traits<char>>(&std::cin, &v5);
11 for ( i = 0; i < strlen(&v5); ++i )
12   v6[i - 112] = (v6[i - 112] ^ 6) + 1;
13 if ( !strcmp(&v5, "akhb~chdaZrdaZudqduvdZvvv|") )
14   std::operator<<<std::char_traits<char>>((int)&std::cout, "yes!you are right");
15 else
16   std::operator<<<std::char_traits<char>>((int)&std::cout, "try again");
17 system("PAUSE");
18 return 0;
19}
```


我们发现加密运算c[i]=m[i]^6+1和密文"akhb~chdaZrdaZudqduvdZvvv|"，求解即可，代码如下

```
c="akhb~chdaZrdaZudqduvdZvvv|"
m=""
for i in c:
 m=m+chr((ord(i)-1)^6)
print m
```

reverse-encrypt3

```
v3 = std::operator<<<std::char_traits<char>>(&std::cout, "Please input your magic number", a3);
std::ostream::operator<<(v3, &std::endl<char,std::char_traits<char>>);
std::istream::operator>>(&std::cin, &v11);
if ( v11 <= 9 )
 return 0xFFFFFFFFLL;
v15 = operator new[](v11);
for ( i = 0; i < v11; ++i )
 std::operator>><char,std::char_traits<char>>(&std::cin, i + v15);
v6 = std::operator<<<std::char_traits<char>>(&std::cout, "You have input sth.", v5);
std::ostream::operator<<(v6, &std::endl<char,std::char_traits<char>>);
v8 = std::operator<<<std::char_traits<char>>(&std::cout, "I will mix them with the enc_flag!", v7);
std::ostream::operator<<(v8, &std::endl<char,std::char_traits<char>>);
for ( j = 0; j <= 37; ++j )
{
 v10 = 0;
 for ( k = 0; k < v11; ++k )
  v10 ^= *(_BYTE *)(k + v15);
 putchar(v10 ^ (unsigned __int8)*(&v16 + j));
}
std::operator<<<std::char_traits<char>>(&std::cout, "\nMaybe you have found the `flag`\n", v9);
return 0LL;
}
```


我们会发现在这里是将我们的输入按字符做异或获得最后结果v10，然后将v10按字符和一串密文异或可能获得明文，这里我们尝试将0-128和密文做异或，找到解，代码如下：

```
c="$.#%9! qwps#&wtu!z$pws$ spwp&rqt{r!'{?"
for j in xrange(128):
 m=""
 for i in c:
  m=m+chr((ord(i)^j))
 print m
```