看雪2017CTF第十题

原创

Ericky_ 于 2017-06-30 16:24:45 发布 1317 收藏

分类专栏： Android Win 文章标签： 函数 rsa 源代码 调试 逆向分析

本文链接：https://blog.csdn.net/hk9259/article/details/74005914

对gmp和RSA不熟悉，从头到尾追了很长时间。这两天也逼着自己熟悉了很多东西，学到了不少。

1.初探

程序是X64的，不管三七二十一，先拖到IDA里面看一下，由于一开始都不知道gmp是什么东西，便一边用OD调试，一边IDA傻傻看代码，一个函数一个函数看。经过漫长而又漫长的分析，结合阅读gmp的源代码，把IDA中的函数对应上了，如下：

```
if ( (unsigned __int8)(v17 - 48) > 9u && (unsigned __int8)(v17 - 65) > 0x19u )// 如果不是数字又不是大写字母就不行
  break;
if ( ++v13 >= 70 )
{
  key2 = KEY;
  word_140052F9C = word_140051F24;
  do
  {
    key7_70 = *((_BYTE *)&KEY + v14 + 6);
    key7_70_byte[v14++] = key7_70;
  }
  while ( key7_70 );
  mpz_init_set_str(&E, (__int64)&key2, 0x10u);// 前6位处理
                                              //
  mpz_init_set_str(&P, (__int64)key7_70_byte, 0x10u);// 后64位处理
  if ( mpz_probab_prime_p((__int64)&P, 0x1F4u) )// 是否是素数
  {
    if ( mpz_probab_prime_p((__int64)&E, 0x1F4u) )
    {
      mpz_set_si(&Q, 0i64);
      mpz_init_set_str(
        &knownVar1,
        (__int64)"6248BC3AB92A33B000FDB88568F19727F92F79EB68FF6AD73203EFD20A3E331BE941C7AA288095F33BC4B255FD983114D"
                 "480EFFBEE2E313E6218A57F9CCC8189",
        0x10u);
      mpz_init_set_str(
        &knownVar2,
        (__int64)"2476A7F02588913F228923E1F36F963F29708C07B117396817A6B94C336FC77FF7D381925EB40CFED8FBE894570155E41"
                 "569B4EC69B26CB0320105A29651CB4B",
        0x10u);
      mpz_set_si(&v22, -0i64);
      mpz_mod((__int64)&v22, (__int64)&knownVar1, (__int64)&P);
      if ( !(unsigned int)sub_1400075D0((__int64)&v22, 0i64) )// ret 0 才可以
      {
        sub_1400071E0(&Q, (__int64)&knownVar1, (__int64)&P);
        if ( (signed int)mpz_cmp((__int64)&P, (__int64)&Q) <= 0 )// P<Q
        {
          mpz_sub_ui(&P, (__int64)&P, 1ui64);// //p=p-1
          mpz_sub_ui(&Q, (__int64)&Q, 1ui64);// q=q-1
          mpz_mul((__int64)&v22, (__int64)&P, (__int64)&Q);// 计算欧拉函数φ(n)=(p-1)*(q-1)
          mpz_invert((__int64)&v22, (__int64)&E, (__int64)&v22);
          v19 = mpz_cmp((__int64)&knownVar2, (__int64)&v22);// v22这里要等于2476A7F02588913F228923E1F36F963F29708C07B117396817A6B94C336FC77FF7D38192
          v20 = "z!!!\n\n";
          if ( !v19 )
            goto LABEL_16;
        }
```


2.关于gmp

GMP简介：GMP是著名的任意精度算术运算库，支持任意精度的整数、有理数以及浮点数的四则运算、求模、求幂、开方等基本运算，还支持部分数论相关运算。Maple、Mathematica等大型数学软件的高精度算术运算功能都是利用GMP实现的。

理清楚函数的意思，可以来看看到底实现了什么。
介绍一下目标程序使用的一些gmp函数，如下：
void mpz_init (mpz_t x)
初始化x。任何一个mpz_t类型的变量在使用前都应该初始化。

int mpz_init_set_str (mpz_t rop, const char *str, int base)
初始化rop，并赋值rop = str，其中str是一个表示base进制整数的字符数组

int mpz_probab_prime_p (const mpz_t n, int reps)
检测n是否为素数。该函数首先对n进行试除，然后使用米勒-拉宾素性检测对n进行测试，reps表示进行检测的次数。如果n为素数，返回2；如果n可能为素数，返回1；如果n为合数，返回0。

void mpz_mul (mpz_t rop, const mpz_t op1, const mpz_t op2)
计算op1 * op2，结果保存在rop中

int mpz_invert(mpz_t rop, const mpz_t op1, const mpz_t op2)
求数论倒数函数

void mpz_init_set_si (mpz_t rop, signed long int op)
初始化rop，并将其值设置为op

int mpz_cmp (mpz_t op1, mpz_t op2)
比较函数

void mpz_mod (mpz_t r, const mpz_t n, const mpz_t d)
求模函数，返回值不为负数

把这些函数的意思弄明白了，就能发现这是一个RSA算法：

RSA算法大体可以分为三个部分：
生成密钥对
加密
解密
其中生成密钥对包括以下步骤：
随机生成两个足够大的素数
p,q
计算公共模数n
n=p*q
计算欧拉函数
φ(n)=(p−1)*(q−1)
选取一较小的与φ(n)互质的正整数e作为公共指数。则数对(n, e)为密钥对中的公钥
计算
d=e−1(modϕ(n))
则数对(n, d)为密钥对中的私钥
这里推荐2篇RSA相关文章
http://www.ruanyifeng.com/blog/2013/06/rsa_algorithm_part_one.html
http://www.ruanyifeng.com/blog/2013/07/rsa_algorithm_part_two.html

搞懂了这些之后，就可以开始写程序了

3.脚本：

```
import gmpy2

N = gmpy2.mpz(0x6248BC3AB92A33B000FDB88568F19727F92F79EB68FF6AD73203EFD20A3E331BE941C7AA288095F33BC4B25
d = gmpy2.mpz(0x2476A7F02588913F228923E1F36F963F29708C07B117396817A6B94C336FC77FF7D381925EB40CFED8FBE89
e = gmpy2.mpz(0x1)




if __name__ == '__main__':

    while e < 0x1000000:
        e = e +1
        kfi = gmpy2.mul(e,d)-1
        k = gmpy2.div(kfi,N)

        if kfi%(k+1) == 0:
            x = 1-kfi/(k+1)+N
            if x > 0 and x < 0x10000000000000000000000000000000000000000000000000000000000000000:
                print 'e:' + hex(e)
                print 'x:' + hex(x)
                continue;


    #X2 = gmpy2.mul(x,x)
    #N4 = gmpy2.mul(N,4)
    #delta = gmpy2.sub(X2,N4)

    delta_root = gmpy2.mpz(0x23c4ffb7dff4f383202beb418be684edaad4c1838af43e9ea0a731d0ea495f00)
    print 'delta_root:' + hex(delta_root)
    fenzi = gmpy2.add(x,delta_root)
    fenzi2 = gmpy2.sub(x,delta_root)
    p1 =  gmpy2.div(fenzi,2)
    p2 = gmpy2.div(fenzi2,2)
    print 'p1 =' + hex(p1)
    print 'p2 =' + hex(p2)
```

结果：

```
Ericky:Downloads heyao$ python calc.py
e:0xf552b3
x:0x13f40bc9da5c21ae87dd77a150d9feca2b17b4a84db1108fe9d4dda6a9c7d9086
delta_root:0x23c4ffb7dff4f383202beb418be684edaad4c1838af43e9ea0a731d0ea495f00
p1 =0xb182de2ac2db8735cf01b2ab4cc338c82e2806043302a3ce9efa861dc36377c3
p2 =0x8dbdde72e2e693b2aed5c769c0dcb3da83534480a80e652ffe53544cd91a18c3
```


答案：

F552B38DBDDE72E2E693B2AED5C769C0DCB3DA83534480A80E652FFE53544CD91A18C3