RSA_总结

原创

IFpop 于 2019-01-23 11:16:31 发布 752 收藏

分类专栏： CTF

本文链接：https://blog.csdn.net/qq_42956710/article/details/86606920

RSA(总结)

签名

签名$sig(m)=m\ (mod\ n)$，将其发个收信方，$(sig(m))=m\ (mod\ n)=m(mod\ n)$

但是如果源数据太大，用私钥进行运算花时间很长，所以现在一般会用hash函数进行处理

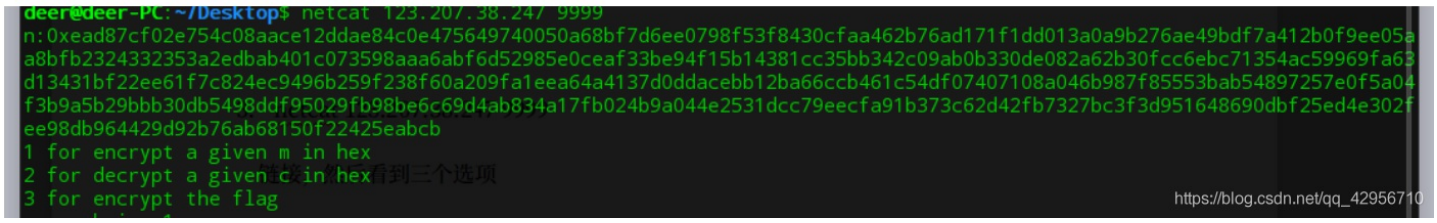
签名$sig(m)=hash(m)\ (mod\ n)$，将其发个收信方，$(sig(m))=hash(m)\ (mod\ n)$

攻击方法

选择密文攻击

> 密码分析者事先任意搜集一定数量的密文，让这些密文透过被攻击的解密算法解密，透过未知的密钥获得解密后的明文。由此能够计算出加密者的私钥或者分解模数，运用这些信息，攻击者可以恢复所有的明文

- 给你解密和加密选择给你被加密的密文
- 给你被加密的密文



分析过程：

用加密算法加密一个较为简单的数，这里我选择的是 x = 2,密文记为r,由RSA原理可知：

$$r\ (mod\ n) = x$$

基于分解模数的RSA攻击

yafu

> yafu 基本命令：factor(n)

factordb

> 一个在线分解网站

n比较小的话，可以选择暴力破解

sage有专门因式分解的函数

公共因子攻击

> 一般有两个n值，求出其公共因子

如果n1 = p*q1*,*n2* = *p*q2，可以求出gcd(n1,n2)的公因子

识别此类题目，通常会发现题目给了若干个n，均不相同，并且明文都没什么联系，e也一般取65537

```
'''
例子：factor跑不出来，通过求公因子，然后分别求出p,q
'''
n1=9051013965404084482870087864821455535159008696042953021965631089095795348830954383127323853272528967729311045179605407693592665683311660581204886571146327720288455874927281128121117323579691204792399913106627543274457036172455814805715668293705603675386878220947722186914112990452722174363713630297685159669328951520891938403452797650685849523658191947411429068829734053745180460758604283051344339641429819373112365211739216160420494167071996438506850526168389386850499796102003625404245645796271690310748804327
n2=1322594839617960381606204641871721479266851241362509156999752436424399599196101889415005920782409383742045137524055031005020939896450631851899162014257592662378041153225723070198582162942572203060872203557069047417125923815394709531030352283197166466606754264903446162172565623486900550129342397518470192972917007728025143621616729305856003008900614022430754256795711817872069827124772614325795379812780557553445737670769517933120624802750045646575902637198160335641394971099420737017550118731532053662385856657 43
```

低加密指数攻击

> 特点：m和e都比较小，e一般取3

$$m < n, m=$$

有可能稍微大一点，$m = k* n+$ ，枚举k进行求解

工具：gmpy2

```
#gmpy2的应用
#!usr/env/python3
import gmpy2
for k in range(10):
    gmpy2.iroot(c+k*n,e)
```

$$c= m \ (mod\ n)$$

Rabin算法

> 算是有固定模板，辨别特点，e = 2
> 这里只关注解密方法,具体看这个https://en.wikipedia.org/wiki/Rabin_cryptosystem

```
def rabin_decrypt(c, p, q, e=2):
    n = p * q
    mp = pow(c, (p + 1) / 4, p)  #整除 用//
    mq = pow(c, (q + 1) / 4, q)
    yp = gmpy2.invert(p, q)
    yq = gmpy2.invert(q, p)
    r = (yp * p * mq + yq * q * mp) % n
    rr = n - r
    s = (yp * p * mq - yq * q * mp) % n
    ss = n - s
    return (r, rr, s, ss)
```

工具

gmpy和gmpy2

```
#gmpy2   在python3中安装：
sudo apt-get install libmpfrc++-dev
sudo apt-get install libmpc-dev
sudo pip3 install gmpy2
#gmpy2   在python2.7中安装：
sudo pip install python-gmpy2
#如果还是不行，请用aptitude
```

安装不成功的看这个安装教程

（这个里面也有一些重要库的安装过程）有兴趣可以看看这个CTF-rsa-tool

使用教程：https://gmpy2.readthedocs.io/en/latest/mpz.html

分解大整数n

- factordb
- yafu

libnum

> 是python2.7里面的一个库,安装使用请点击libnum

安装：

```
git clone https://github.com/hellman/libnum
cd libnum
python setup.py install
```

我们在RSA中常用函数有：

```
import libnum
libnum.n2s(n)  #Number to string
libnum.b2s(b)  #Binary to string
libnum.gcd(num1,num2) #求最大公约数
libnum.lcm(num1,num2) #最小公倍数
libnum.s2n(s)  #string to Number
libnum.s2b(s) #string to Binar
```

CTF_RSA_tool

> 一个集成工具

RSAtools

> 已知n,e,p,q，可以求出d

pycrypto

> 一个有着多种解密算法的python库

安装：

```
deepin:
sudo pip3 install pycrypto
```

那个链接里面有这个库的各种用法，想用的可以玩一下

openssl

> openssl则是SSL的实现版，openssl还包含了公钥私钥的生成、摘要生成等各种工具

linux里面应该是自带的

用法：

```
#提取公钥信息
openssl rsa -pubin -text -modulus -in [公钥.pem]
#用私钥进行解密
openssl rsautl -decrypt -in [密文.enc]-inkey [私钥.pem]
#其他命令可以看看openssl文档，点击上面那个openssl
```

rsatool

> 另一款强大的rsa解密工具，主要用来生成私钥，具体用法看writeup

rsa

> 这是一个纯python实现的库，不依赖底层文件，优点是部署容易，缺点是速度比较慢

这里有其他方法

常见题型

已知p、q、e求解d

思路：

由于

$$* d \equiv 1 \ (mod \ (n))$$

方法：

```python
#RSA实践
#三种解法
'''
第一种gmpy2解法
mpz:Multiple-precision Integers（多精度型整数）
gmpy2.invert():求模逆
'''
import gmpy2
p =gmpy2.mpz(473398607161)
q =gmpy2.mpz(4511491)
e =gmpy2.mpz(17)
fn= (p - 1) * (q - 1)
d = gmpy2.invert(e, fn)
print("d is:")
print (d)

'''
直接用扩展欧几里得法
'''
def computeD(fn, e):
    (x, y, r) = ext_euclid(fn, e)
    #y maybe < 0, so convert it
    if y < 0:
        return fn + y
    return y

def ext_euclid(a, b):
    if b == 0:
        return (1, 0, a)
    else:
        x, y, q = ext_euclid(b, a % b)
        # q = gcd(a, b) = gcd(b, a%b)
        x, y = y, (x - (a // b) * y)  # '//'代表整除， 可以用python演示一下.
        return (x, y, q)

p = 473398607161
q = 4511491
e = 17

n = p * q
print("n："+str(n))
fn = (p - 1) * (q - 1)
print("f(n):"+str(fn))
d = computeD(fn, e)
print(d)

'''
用工具解密——RSAtools
ps:用这个工具记得将e转化为hex形式
'''
```

已知c、n、e求解明文/已知 c ,q,p,e

根据上述分解大数工具，我们可以求出 p ,q 利用 $(n)=(n-1)$