

XCTF—RSA256

原创

loading... 于 2020-10-13 20:39:32 发布 719 收藏 1

分类专栏: [CTF](#)

版权声明: 本文为博主原创文章, 遵循 [CC 4.0 BY-SA](#) 版权协议, 转载请附上原文出处链接和本声明。

本文链接: https://blog.csdn.net/qq_41137110/article/details/109060191

版权



[CTF 专栏收录该内容](#)

11 篇文章 0 订阅

订阅专栏

题目地址

题目名称: RSA256

题目描述:

被潘汉年按时来到上海百老汇大厦(今上海大厦), 叩开了袁殊临时下榻处的房门。袁殊说明自己当前身份和处境后, 突然话锋一转, 问潘汉年: “你到我这里来, 恐怕已经被日本特务注意到了。我应该怎么向他们解释呢?”潘汉年说: “你就将计就计在敌伪内部站住脚, 取得合法地位。同时搜集敌伪情报向我提供。”“要是岩井要求我将你介绍给他怎么办?”“那我就用胡越明的化名同岩井见面, 就说我愿意和你在香港合作搞情报。”时隔不久, 潘汉年接到袁殊通知, 通知内容为: RSA256.tar.gz, 要他在上海虹口区一家日本人开的餐馆里, 和岩井会见。请以暗号形式告知我方人员前往保护潘汉年的安全。(答案为flag{XXX}形式)

题目附件: [附件1](#)

WriteUp:

解题思路:

下载附件, 解压得到RSA256文件夹, 里面有两个文件flllllag.txt和gy.key

名称	修改日期	类型	大小
flllllag.txt	2019/3/25 13:10	文本文档	1 KB
gy.key	2019/3/25 13:10	注册表项	1 KB

给出了公钥文件gy.key和密文文件flllllag.txt, 就是常规的RSA解密, 有多种方法

方法一: 利用RsaCtfTool工具 (kali虚拟机中)

已知公钥 (自动求私钥) -publickey, 密文-uncipherfile

命令: `python3 RsaCtfTool.py --publickey 公钥文件 --uncipherfile 加密的文件`

```

root@kali:/home/zww/ctf/ctf_tools/RsaCtfTool#
root@kali:/home/zww/ctf/ctf_tools/RsaCtfTool#
root@kali:/home/zww/ctf/ctf_tools/RsaCtfTool# python3 RsaCtfTool.py --publickey /var/run/vmblock-fuse/blockdir/cQkNeU/gy.key --uncipherfile /var/run/vmblock-fuse/blockdir/hwbsgQ/fllllllag.txt
private argument is not set, the private key will not be displayed, even if recovered.

[*] Testing key /var/run/vmblock-fuse/blockdir/cQkNeU/gy.key.
Can't load ecm because sage is not installed
Can't load boneh_durfee because sage is not installed
Can't load ecm2 because sage is not installed
Can't load qicheng because sage is not installed
Can't load roca because sage is not installed
Can't load smallfraction because sage is not installed
[*] Performing factordb attack on /var/run/vmblock-fuse/blockdir/cQkNeU/gy.key.

Results for /var/run/vmblock-fuse/blockdir/cQkNeU/gy.key:

Unciphered data :
HEX : 0x0002638b4cc27586c6be00666c61677b5f326f21395f4354465f4543554e5f7d
INT (big endian) : 4220722534205764009434028013523762457981424524987027846762890641397407613
INT (little endian) : 56707497178957223025740311097346441015153867321342088366492533572570498400768
STR : b'\x00\x02c\x8bL\xc2u\x86\xc6\xbe\x00flag{_2o!9_CTF_ECUN_}'
root@kali:/home/zww/ctf/ctf_tools/RsaCtfTool#

```

解密结果 (包括16进制, int, 字符串)

flag

https://blog.csdn.net/qq_41137110

直接解出明文

方法二：利用公钥文件用openssl工具解出e、n，然后python3脚本解出明文

1、解出e、n

方式1:

打开kali虚拟机，用openssl解出e、n

命令：`openssl rsa -pubin -text -modulus -in warmup -in gy.key`

```

root@kali:~/桌面/RSA256# openssl rsa -pubin -text -modulus -in warmup -in gy.key
RSA Public-Key: (256 bit)
Modulus:
 00:a9:bd:4c:7a:77:63:37:0a:04:2f:e6:be:c7:dd:
 c8:41:60:2d:b9:42:c7:a3:62:d1:b5:d3:72:a4:d0:
 89:12:d9
Exponent: 65537 (0x10001)
Modulus=A9BD4C7A7763370A042FE68EC7DDC841602DB942C7A362D1B5D372A4D08912D9
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAKm9THp3YzcKBC/mvsfdyEFgLbLcX6Ni
0bXTcqTQiRLZAgMBAE=
-----END PUBLIC KEY-----
root@kali:~/桌面/RSA256#

```

https://blog.csdn.net/qq_41137110

方式2:

用脚本从公钥文件中解出n、e

```

# -*- coding: cp936 -*-
from Crypto.PublicKey import RSA

#1. 从公钥文件中分解n和e
public_key = RSA.importKey(open(r"G:\ctf\CTF题目\8eec4a4af1e14eb08648c8fda7660a0f\8eec4a4af1e14eb08648c8fda7660a0f\RSA256\gy.key", 'rb').read())
n = public_key.n
e = public_key.e
print('N:',n)
print('E:',e)

```

运行结果

```

C:\Users\zww>python C:\Users\zww\Desktop\rsa.py
N: 76775333340223961139427050707840417811156978085146970312315886671546666259161
E: 65537

```

2、e为65537，n还比较短，用python先转换成10进制

```
C:\Users\zww>python
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 0xA9BD4C7A7763370A042FE6BEC7DDC841602DB942C7A362D1B5D372A4D08912D9
76775333340223961139427050707840417811156978085146970312315886671546666259161
```

对n进行因数分解

方式1: 登录网站<http://factordb.com/>,解出p和q

Search	Sequences	Report results	Factor tables	Status
76775333340223961139427050707840417811156978085146970312315886671546666259161				Factorize! (2)
Result:				
digits	number			
77 (show)	7677533334...61<77> = 273821108020968288372911424519201044333<39> · 280385007186315115828483000867559983517<39>			

方式2: 使用yafu工具

(常用于比较大的整数分解)自动整数因式分解, 在RSA中, 当p、q的取值差异过大或过于相近的时候, 使用yafu可以快速的把n值分解出p、q值!

用法:

```
PS G:\ctf\CTF工具\yafu-1.34> .\yafu-x64.exe
factor(76775333340223961139427050707840417811156978085146970312315886671546666259161)

fac: factoring 76775333340223961139427050707840417811156978085146970312315886671546666259161
fac: using pretesting plan: normal
fac: no tune info: using qs/gnfs crossover of 95 digits
div: primes less than 10000
fmt: 1000000 iterations
rho: x^2 + 3, starting 1000 iterations on C77
rho: x^2 + 2, starting 1000 iterations on C77
rho: x^2 + 1, starting 1000 iterations on C77
pml: starting B1 = 150K, B2 = gmp-ecm default on C77
ecm: 30/30 curves on C77, B1=2K, B2=gmp-ecm default
ecm: 74/74 curves on C77, B1=11K, B2=gmp-ecm default
ecm: 149/149 curves on C77, B1=50K, B2=gmp-ecm default, ETA: 0 sec

starting SIQS on c77: 76775333340223961139427050707840417811156978085146970312315886671546666259161
==== sieving in progress (1 thread): 36224 relations needed ====
==== Press ctrl-c to abort and save state ====
36340 rels found: 18493 full + 17847 from 189308 partial, (2203.00 rels/sec)

SIQS elapsed time = 95.9716 seconds.
Total factoring time = 115.8173 seconds

***factors found***

P39 = 280385007186315115828483000867559983517
P39 = 273821108020968288372911424519201044333

ans = 1

PS G:\ctf\CTF工具\yafu-1.34>
```

到此, 已经获取到RSA的全部参数

p = 273821108020968288372911424519201044333

q = 280385007186315115828483000867559983517

n=76775333340223961139427050707840417811156978085146970312315886671546666259161

e=65537

3、使用python3代码解出明文

```
# -*- coding: cp936 -*-
import base64
from Crypto.PublicKey import RSA
def egcd(a,b):
    if a==0:
        return (b,0,1)
    else:
        g,y,x=egcd(b%a,a)
        return (g,x-(b//a)*y,y)
def modinv(a,m):
    g,x,y=egcd(a,m)
    if g!=1:
        raise Exception('modular inverse does not exist')
    else:
        return x%m
p = 273821108020968288372911424519201044333
q = 280385007186315115828483000867559983517
n = p*q
e = 65537
d=modinv(e,(p-1)*(q-1))#RSA私钥

with open(r"fl1111lag.txt" , "rb") as f:
    s=f.read().hex()#bytes转16进制字符串
c=int(s,16);#密文, 16进制转成int型
#解出明文
m=pow(c,d,n)#得到的是10进制数据
hex=hex(m)#得到16进制数据, 最后转字符串就行了
print(hex)#输出16进制数据
#因为base16编码后的字母组成是[0-9A-F],所以要转成大写, 否则会提示"Non-base16 digit found"
#还可以写成flag=base64.b16decode(hex[2:], True)或者修改python库base64源码里的b16decode()函数第二个参数为True
flag=base64.b16decode(hex[2:].upper())
print(flag)#输出解码后的字符串
```

运行结果如下:

```
C:\Users\zww>python C:\Users\zww\Desktop\rsa.py
0x2638b4cc27586c6be0666c61677b5f326f21395f4354465f4543554e5f7d
Traceback (most recent call last):
  File "C:\Users\zww\Desktop\rsa.py", line 31, in <module>
    flag=base64.b16decode(hex[2:].upper())
  File "E:\anaconda3\lib\base64.py", line 268, in b16decode
    return binascii.unhexlify(s)
binascii.Error: Odd-length string
```

长度为61位, 不是偶数, 所以报错, 因为flag前的位数无意义, 只要提取红框里的字符串解码就ok

可以看出flag字符串16进制标识, 直接base16解码得到flag:

flag{_2o!9_CTF_ECUN_}

提交时需要去掉_(真坑)