Whctf 2017 -UNTITLED- Writeup

转载

baikeng3674 于 2017-09-18 09:11:00 发布 235 收藏
文章标签： python
原文链接：http://www.cnblogs.com/WangAoBo/p/7541481.html

Whctf 2017 -UNTITLED- Writeup

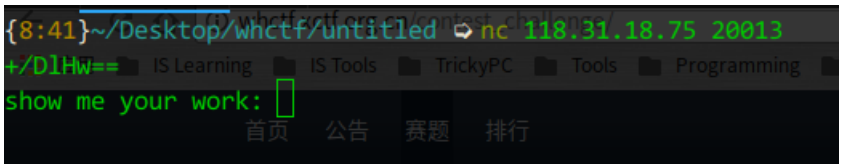
> 转载请表明出处http://www.cnblogs.com/WangAoBo/p/7541481.html

分析：

下载下来的附件是一个py脚本，如下

```python
from Crypto.Util.number import getPrime,long_to_bytes,bytes_to_long
import primefac
import time
from os import urandom
import hashlib
import sys
class Unbuffered(object):
   def __init__(self, stream):
       self.stream = stream
   def write(self, data):
       self.stream.write(data)
       self.stream.flush()
   def __getattr__(self, attr):
       return getattr(self.stream, attr)
import sys
sys.stdout = Unbuffered(sys.stdout)
def gen_args():
    p=getPrime(1024)
    q=getPrime(1024)
    n=p*q
    e=0x10001
    d=primefac.modinv(e,(p-1)*(q-1))%((p-1)*(q-1))
    return (p,q,e,n,d)
def proof():
    salt=urandom(4)
    print salt.encode("base64"),
    proof=raw_input("show me your work: ")
    if hashlib.md5(salt+proof.decode("base64")).hexdigest().startswith("0000"):
        print "checked success"
        return 1
    return 0

def run():
    if not proof():
        return
    m=int(open("/home/bibi/PycharmProjects/work/whctf/flag","r").read().encode("hex"),16)#flag{*}
    (p,q,e,n,d)=gen_args()
    c=pow(m,e,n)
    print "n:",hex(n)
    print "e:",hex(e)
    print "c:",hex(c)
    t=int(hex(m)[2:][0:8],16)
    u=pow(t,e,n)
    print "u:",hex(u)
    print "===="
    x=int(hex(m)[2:][0:8]+raw_input("x: "),16)
    print "===="
    y=int(raw_input("y: "),16)
    if (pow(x,e,n)==y and pow(y,d,n)==t):
        print "s:",hex(int(bin(p)[2:][0:568],2))
run()
```

nc连上之后如下：



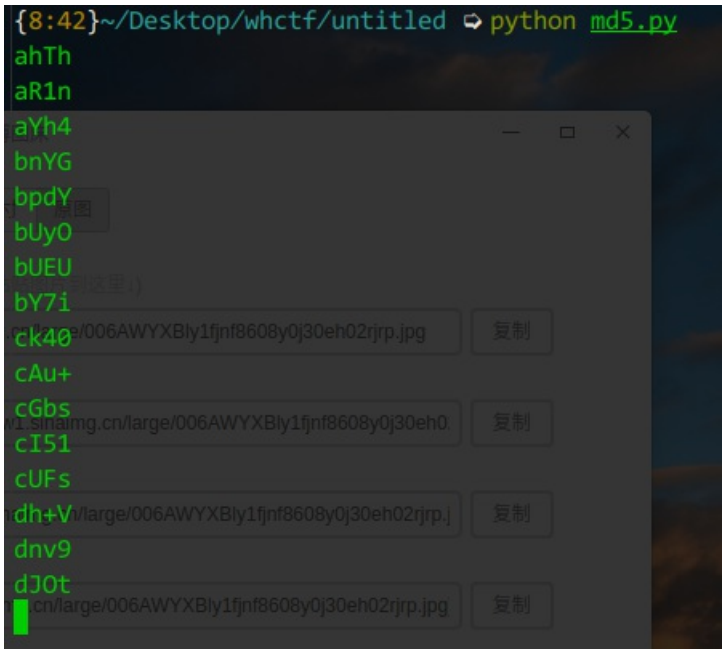
结合py代码分析，首先要通过**proof()**函数的验证，即要满足：

```
hashlib.md5(salt+proof.decode("base64")).hexdigest().startswith("0000")
```

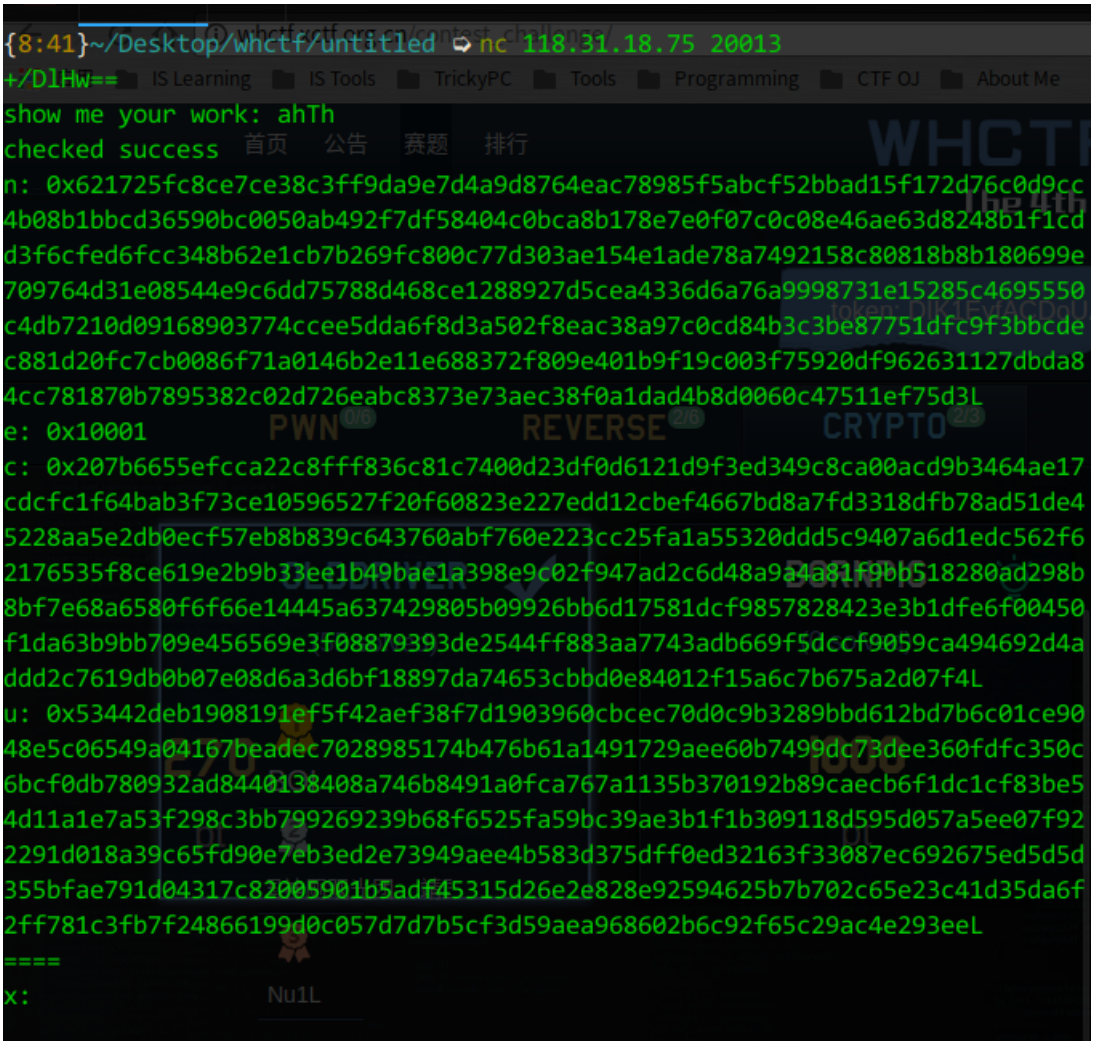
写一个脚本爆破，使盐值和输入内容的base64解码的md5开头四位为0000

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

import hashlib
import string

#base64编码后的范围
dic = string.ascii_letters + string.digits + "+/"

#盐值
salt = '+/DlHw=='.decode('base64')

#先尝试爆破4位
for a in dic:
    for b in dic:
        for c in dic:
            for d in dic:
                proof = a + b + c + d
                try:
                    if hashlib.md5(salt + proof.decode("base64")).hexdigest().startswith("0000"):
                        print proof
                        exit(0)
                except:
                    pass
```

很快就爆破出多组结果



随便找一组提交，通过了**proof**函数的验证，得到了**n, e, c, u**



再分析题目给的py脚本，可以看出

c即为**flag**经过**RSA**加密的密文，其中**n**和**e**已知

u为m的前8位（根据flag形式，即为f）经过RSA加密后的值

```
x=int(hex(m)[2:][0:8]+raw_input("x: "),16)

y=int(raw_input("y: "),16)

pow(x,e,n)==y and pow(y,d,n)==t
```

以上三行连起来分析，很容易得出当我们输入的**x为空，y为u**时，即可通过最后一步if的验证，从而得到**p的前568位**（输入y时记得去掉最后的L）

```
c: 0x207b6655efcca22c8fff836c81c7400d23df0d6121d9f3ed349c8ca00acd9b3464ae17
cdcfc1f64bab3f73ce10596527f20f60823e227edd12cbef4667bd8a7fd3318dfb78ad51de4
5228aa5e2db0ecf57eb8b839c643760abf760e223cc25fa1a55320ddd5c9407a6d1edc562f6
2176535f8ce619e2b9b33ee1b49bae1a398e9c02f947ad2c6d48a9a4a81f9bb518280ad298b
8bf7e68a6580f6f66e14445a637429805b09926bb6d17581dcf9857828423e3b1dfe6f00450
f1da63b9bb709e456569e3f08879393de2544ff883aa7743adb669f5dccf9059ca494692d4a
ddd2c7619db0b07e08d6a3d6bf18897da74653cbbd0e84012f15a6c7b675a2d07f4L
u: 0x53442deb1908191ef5f42aef38f7d1903960cbcec70d0c9b3289bbd612bd7b6c01ce90
48e5c06549a04167beadec7028985174b476b61a1491729aee60b7499dc73dee360fdfc350c
6bcf0db780932ad8440138408a746b8491a0fca767a1135b370192b89caecb6f1dc1cf83be5
4d11a1e7a53f298c3bb799269239b68f6525fa59bc39ae3b1f1b309118d595d057a5ee07f92
2291d018a39c65fd90e7eb3ed2e73949aee4b583d375dff0ed32163f33087ec692675ed5d5d
355bfae791d04317c82005901b5adf45315d26e2e828e92594625b7b702c65e23c41d35da6f
2ff781c3fb7f24866199d0c057d7d7b5cf3d59aea968602b6c92f65c29ac4e293eeL
====
x:
====
y: 0x53442deb1908191ef5f42aef38f7d1903960cbcec70d0c9b3289bbd612bd7b6c01ce90
48e5c06549a04167beadec7028985174b476b61a1491729aee60b7499dc73dee360fdfc350c
6bcf0db780932ad8440138408a746b8491a0fca767a1135b370192b89caecb6f1dc1cf83be5
4d11a1e7a53f298c3bb799269239b68f6525fa59bc39ae3b1f1b309118d595d057a5ee07f92
2291d018a39c65fd90e7eb3ed2e73949aee4b583d375dff0ed32163f33087ec692675ed5d5d
355bfae791d04317c82005901b5adf45315d26e2e828e92594625b7b702c65e23c41d35da6f
2ff781c3fb7f24866199d0c057d7d7b5cf3d59aea968602b6c92f65c29ac4e293ee
s: 0xb447abcd768378f05675b98f4724e934b1a7251749b14b11d3af19d3a47e98dbf90b94
a77a01ab76e6a7f99d5b79cfce8e9edfcc7b626ed0f1699d743fa78bd73ff4a03f904bdeL
```

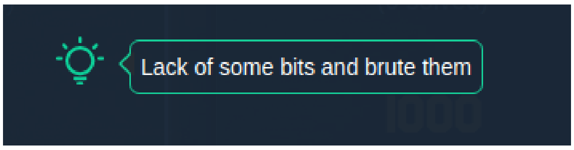
至此整理一下我们得到的信息：

- flag加密后的密文c
- 加密所用到的n和e
- p的前568位

很容易联想到恢复p，从而算出q，再解RSA就能拿到flag

步骤：

这个时候想到了国赛的一道类似题目**Partial**，也是知道n，e和高位p，需要恢复p，因此也选用相同的方法**Coppersmith Attack**(https://github.com/Gao-Chuan/RSA-and-LLL-attacks#factoring-with-high-bits-known)，但**Coppersmith Attack**方法需要我们最少知道576位p，已知568位，差了两个16进制数，根据官方给的hint

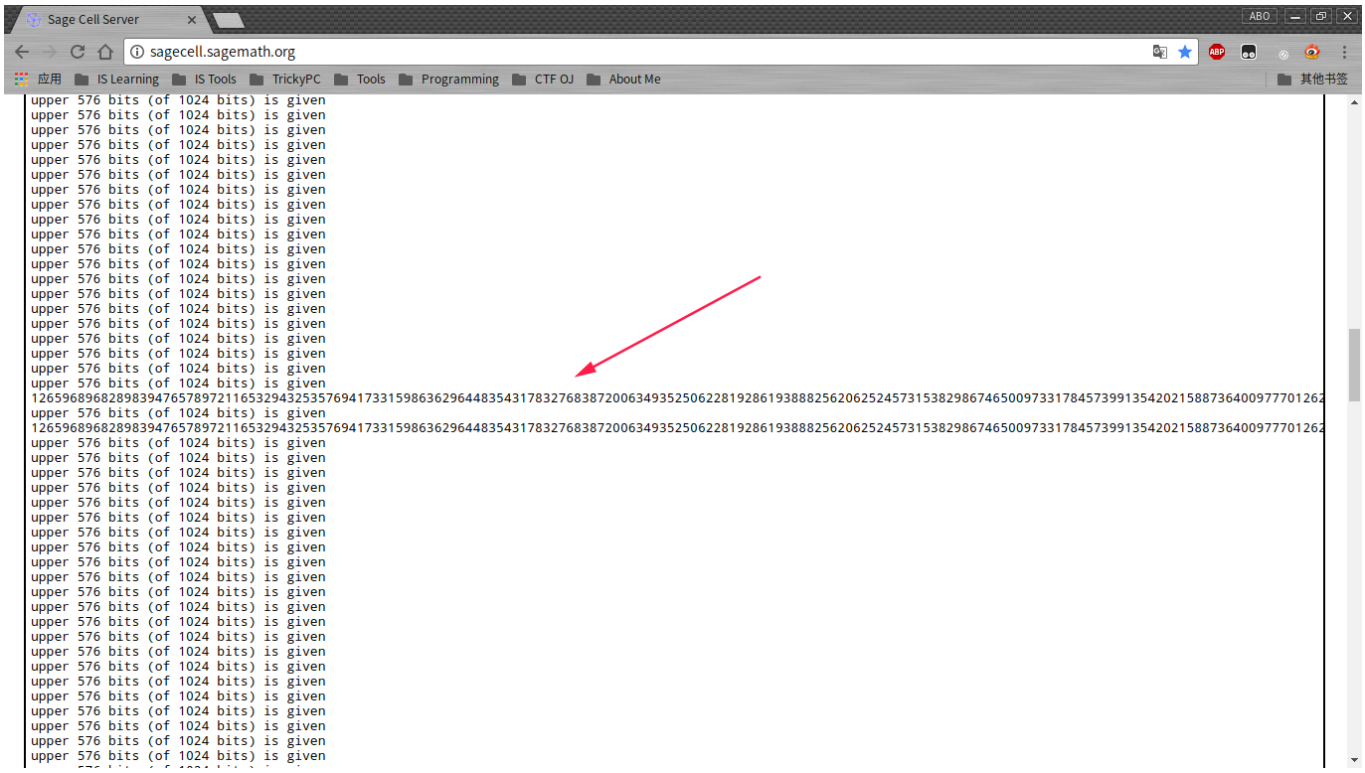


很明显我们需要爆破出要补上的两位

在强大的sagemath上写了一个爆破的代码，在线运行（http://sagecell.sagemath.org/）

```
n = 0x621725fc8ce7ce38c3ff9da9e7d4a9d8764eac78985f5abcf52bbad15f172d76c0d9cc4b08b1bbcd36590bc0050ab492f7df58404c0bca8b178e7e0f07c0c08e46ae63d8248b1f1cdd3f6cfed6fcc348b62e1cb7b269fc800c77d303ae154e1ade78a7492158c80818b8b180699e709764d31e08544e9c6dd75788d468ce1288927d5cea4336d6a76a9998731e15285c4695550c4db7210d09168903774ccee5dda6f8d3a502f8eac38a97c0cd84b3c3be87751dfc9f3bbcdec881d20fc7cb0086f71a0146b2e11e688372f809e401b9f19c003f75920df962631127dbda84cc781870b7895382c02d726eabc8373e73aec38f0a1dad4b8d0060c47511ef75d3
p = 0xb447abcd768378f05675b98f4724e934b1a7251749b14b11d3af19d3a47e98dbf90b94a77a01ab76e6a7f99d5b79cfce8e9edfcc7b626ed0f1699d743fa78bd73ff4a03f904bde

import string
dic = string.digits + "abcdef"

for a in dic:
    for b in dic:
        pp = hex(p) + a + b
        #p需要用0补全到1024位
        pp += '0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000'
        #要加的数字与补全p时0的个数有关
        pp = int(pp, 16)
        p_fake = pp+0x10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
        pbits = 1024
        kbits = pbits-576
        pbar = p_fake & (2^pbits-2^kbits)
        print "upper %d bits (of %d bits) is given" % (pbits-kbits, pbits)
        PR.<x> = PolynomialRing(Zmod(n))
        f = x + pbar
        try:
            x0 = f.small_roots(X=2^kbits, beta=0.4)[0]  # find root < 2^kbits with factor >= n^0.4
            print x0 + pbar
        except:
            pass
```

爆破出了p如下：



现在知道了n, e, c, p，解开RSA就可以了，python脚本如下（当然把解RSA的过程写在sage代码中也是可以的）

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

p = 
126596896828983947657897211653294325357694173315986362964483543178327683872006349352506228192861938882562062524573153829867465009733178457399135420215887364009777012624212242069216745138202953735034716032666189414323613790242613717531697843979604409625853777348356827810939640137432278820298916431800157020739
n = 
0x621725fc8ce7ce38c3ff9da9e7d4a9d8764eac78985f5abcf52bbad15f172d76c0d9cc4b08b1bbcd36590bc0050ab492f7df58404c0bca8b178e7e0f07c0c08e46ae63d8248b1f1cdd3f6cfed6fcc348b62e1cb7b269fc800c77d303ae154e1ade78a7492158c80818b8b180699e709764d31e08544e9c6dd75788d468ce1288927d5cea4336d6a76a9998731e15285c4695550c4db7210d09168903774ccee5dda6f8d3a502f8eac38a97c0cd84b3c3be87751dfc9f3bbcdec881d20fc7cb0086f71a0146b2e11e688372f809e401b9f19c003f75920df962631127dbda84cc781870b7895382c02d726eabc8373e73aec38f0a1dad4b8d0060c47511ef75d3L
e = 0x10001
c = 
0x207b6655efcca22c8fff836c81c7400d23df0d6121d9f3ed349c8ca00acd9b3464ae17cdcfc1f64bab3f73ce10596527f20f60823e227edd12cbef4667bd8a7fd3318dfb78ad51de45228aa5e2db0ecf57eb8b839c643760abf760e223cc25fa1a55320ddd5c9407a6d1edc562f62176535f8ce619e2b9b33ee1b49bae1a398e9c02f947ad2c6d48a9a4a81f9bb518280ad298b8bf7e68a6580f6f66e14445a637429805b09926bb6d17581dcf9857828423e3b1dfe6f00450f1da63b9bb709e456569e3f08879393de2544ff883aa7743adb669f5dccf9059ca494692d4addd2c7619db0b07e08d6a3d6bf18897da74653cbbd0e84012f15a6c7b675a2d07f4L

import libnum
import gmpy2

q = n / p
assert n == p * q

d = gmpy2.invert(e, (p - 1) * (q - 1))
m = pow(c, d, n)
print libnum.n2s(m)
```

> python 解RSA的姿势（http://www.cnblogs.com/WangAoBo/p/7513811.html）

运行，即可得到flag

```
{9:06}~/Desktop/whctf/untitled ➪ python rsa.py
flag{rs4_y0ok_s0_m2ch_1n_c7f_qu4ls_c0mp7t1t10n}
{9:06}~/Desktop/whctf/untitled ➪
```

转载于:https://www.cnblogs.com/WangAoBo/p/7541481.html