强网杯2018 - nextrsa - Writeup

转载

weixin_30627341 于 2018-03-26 21:33:00 发布 5074 收藏 1
文章标签： python
原文链接：http://www.cnblogs.com/WangAoBo/p/8654120.html
版权

强网杯2018 - nextrsa - Writeup

> 原文地址：M4x@10.0.0.55
>
> 所有代码均已上传至我的github

俄罗斯套娃一样的rsa题目，基本把我见过的rsa套路出了一遍，值得记录一下

level 0

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[*] Switching to interactive mode
ok!
Firstly, please give me the proof of your work!
x=chr(random.randint(0,0xff))+chr(random.randint(0,0xff))+chr(random.randint(0,0x1f))
hashlib.sha256(x).hexdigest()[0:8]=='372c8af8'
@ x.encode('hex')=[*] Got EOF while reading in interactive
$
```

发送teamtoken后，到第0关，较简单，爆破sha256即可，此时代码如下：

```
QWB_nextrsa [master●] cat exp.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
from hashlib import sha256
#  context.log_level = "debug"

def brute(cipher):
    #  success("cipher -> {}".format(cipher))
    #  print type(cipher)
    for a in xrange(0, 0xff):
        for b in xrange(0, 0xff):
            for c in xrange(0, 0xff):
                x = chr(a) + chr(b) + chr(c)
                if sha256(x).hexdigest()[0: 8] == cipher:
                    #  success("x -> {}".format(x))
                    return x
    print "not found"

if __name__ == "__main__":
    io = remote("39.107.33.90", 9999)

    token = "icq9bae582b7f5d9ab6caed7d40150be"
    io.sendlineafter(":", token)

    io.recvuntil("=='")
    cipher = io.recvuntil("'", drop = True)
    x = brute(cipher)
    io.sendlineafter(")=", x.encode('hex'))
    success("Level 1 Clear!")

    io.interactive()
    io.close()
```

> 碰撞失败的话多次尝试即可

level 1

为了方便后续处理，先写一个解rsa的函数，可以参考python使用libnum，gmpy2快速解RSA

```
def rsa(n, p, q, e, c):
    assert n == p * q

    d = invert(e, (p - 1) * (q - 1))
    m = pow(c, d, n)

    return m
```

下一关，给了n，e，c，并且经过尝试，n，c是不变的，只有c在改变

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[*] Switching to interactive mode
ok!


input format:almost hex(m).replace("L","")

=next-rsa=
# n=0xc4606b153b9d06d934c9ff86a3be5610266387d82d11f3b4e354b1d95fc7e577
# e=0x10001
# c=0xa87fc50517b50db03a038c93c2a2c2c36de67660920da8720b787fedc3e19dd9
@ m=[*] Got EOF while reading in interactive
$
```

尝试在在线网站分解n，发现能直接分解

> p = 289540461376837531747468286266019261659
>
> q = 306774653454153140532319815768090345109

那么直接解开就可以了，此时代码如下：

```
QWB_nextrsa [master●] cat exp.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from pwn import *
from hashlib import sha256
from gmpy2 import invert
#  context.log_level = "debug"

def brute(cipher):
    #  success("cipher -> {}".format(cipher))
    #  print type(cipher)
    for a in xrange(0, 0xff):
        for b in xrange(0, 0xff):
            for c in xrange(0, 0xff):
                x = chr(a) + chr(b) + chr(c)
                if sha256(x).hexdigest()[0: 8] == cipher:
                    #  success("x -> {}".format(x))
                    return x
    print "not found"

def rsa(n, p, q, e, c):
    assert n == p * q

    d = invert(e, (p - 1) * (q - 1))
    m = pow(c, d, n)

    return m
```

```
fmt = lambda m: hex(m).replace("L", "")

if __name__ == "__main__":
    io = remote("39.107.33.90", 9999)

    token = "icq9bae582b7f5d9ab6caed7d40150be"
    io.sendlineafter(":", token)

    io.recvuntil("=='")
    cipher = io.recvuntil("'", drop = True)
    x = brute(cipher)
    io.sendlineafter(")=", x.encode('hex'))
    success("Level 1 Clear!")

    io.recvuntil("# n=")
    n = int(io.recvuntil("\n", drop = True), 16)
    io.recvuntil("# e=")
    e = int(io.recvuntil("\n", drop = True), 16)
    io.recvuntil("# c=")
    c = int(io.recvuntil("\n", drop = True), 16)
    p =  289540461376837531747468286266019261659
    q = 306774653454153140532319815768090345109
    m = fmt(rsa(n, p, q, e, c))
    io.sendlineafter("m=", m)
    success("Level 2 Clear!")

    io.interactive()
    io.close()
```

level 2

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[+] Level 2 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# n=0x92411fa0c93c1b27f89e436d8c4698bcf554938396803a5b62bd10c9bfcbf85a483bd87bb2d6a8dc00c32d8a7caf30d8899
# e=0x6f6b385dd0f06043c20a7d8e5920802265e1baab9d692e7c20b69391cc5635dbcaae59726ec5882f168b3a292bd52c97653
# c=0x2add7528efe278a70a43f97fc5af83bbaab1238364735d998de005d7feb1a8ab931c7410f0f785db455857b8154a68de318
@ m=$
```

给了n，e，c，求出m，发现**e**很大，直接尝试wiener attack

```
QWB_nextrsa [master●] python rsa-wiener-attack/RSAwienerHacker.py 0x92411fa0c93c1b27f89e436d8c4698bcf5549
--------------------------
Hacked!
42043
QWB_nextrsa [master●]
```

求出了d，就可以解出明文了，代码太长就不贴了，所有代码都放在了我的github中了

level 3

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# n=0x79982a272b9f50b2c2bc8b862ccc617bb39720a6dc1a22dc909bbfd1243cc0a03dd406ec0b1a78fa75ce5234e8c57e0aab4
# e=0x3
# c=0x381db081852c92d268b49a1b9486d724e4ecf49fc97dc5f20d1fad902b5cdfb49c8cc1e968e36f65ae9af7e8186f15ccdca
# b=0xfedcba987654321000000000000000000000000000000000000000000000000000000000000000000000000000000000000
# m=b+x (x:64bit)
@ x=[*] Got EOF while reading in interactive
$
```

已知n，c，e和明文的高位，本来看到e=3是想用低加密指数爆破的，尝试了一下没有出结果才注意到实际上我们已经有了明文的高位，这时才想起来在Coppersmith中有相关的介绍，同时也找到了一篇相关的博客，使用sage可以跑出结果，sage代码如下：

```
QWB_nextrsa [master●] cat copper.sage
# partial_m.sage

n = 0x79982a272b9f50b2c2bc8b862ccc617bb39720a6dc1a22dc909bbfd1243cc0a03dd406ec0b1a78fa75ce5234e8c57e0aab4
e = 3

m = randrange(n)
c = pow(m, e, n)

beta = 1
epsilon = beta^2/7

nbits = n.nbits()
kbits = floor(nbits*(beta^2/e-epsilon))
#mbar = m & (2^nbits-2^kbits)
mbar = 0xfedcba987654321000000000000000000000000000000000000000000000000000000000000000000000000000000000
c = 0x381db081852c92d268b49a1b9486d724e4ecf49fc97dc5f20d1fad902b5cdfb49c8cc1e968e36f65ae9af7e8186f15ccdca
print "upper %d bits (of %d bits) is given" % (nbits-kbits, nbits)

PR.<x> = PolynomialRing(Zmod(n))
f = (mbar + x)^e - c

print m
x0 = f.small_roots(X=2^kbits, beta=1)[0]  # find root < 2^kbits with factor = n
print mbar + x0
print x0
```

使用sagemath运行即可得到缺失的明文

> 可以使用在线运行sage的网站

level 4

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[+] Level 4 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# n=0x78e2e04bdc50ea0b297fe9228f825543f2ee0ed4c0ad94b6198b672c3b005408fd8330c36f55d36fb129d308c23e5cb8f4d
# e=0x10001
# nextprime(p)*nextprime(q)=0x78e2e04bdc50ea0b297fe9228f825543f2ee0ed4c0ad94b6198b672c3b005408fd8330c36f5
# c=0x1c3588ac81ec3d1b439cfd2d5e6e8a5a95c8f95aaeff1b0ba49276ade80435323f307a17006ae2ffb4ca321e54387d9b33e
@ m=$
```

这一步的解决方式第一次遇到，比赛结束后也听许多表哥说了他们的方法，一个比一个精彩，这里介绍一下广外表哥和kira大佬的两种方法

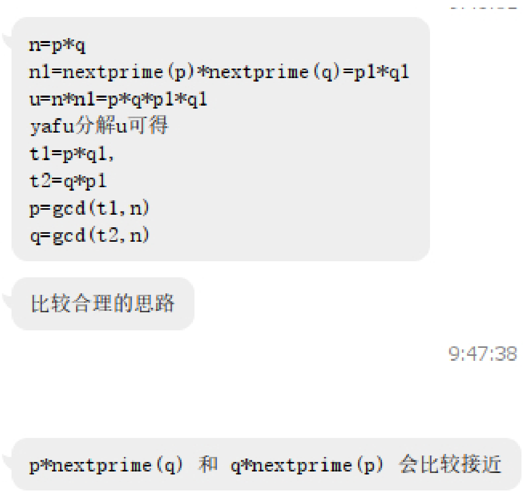
> - pq = n
> - (p + x)(q + y) = n'
>
> -> xy + py + qx = t(t = n' - n)
>
> -> **xq^2^ + (xy - t)q + ny = 0**
>
> 则该方程有素数接即可，可爆破x，y

实现脚本

```
QWB_nextrsa [master●] cat next.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'


from gmpy2 import is_prime as prime
from gmpy2 import iroot

n = 152604733989160716862877523402491582793430130771456677945147176923529418734666906862884422047145858
nn = 152604733989160716862877523402491582793430130771456677945147176923529418734666906862884422047145858

#  print nn > n
t = nn - n
f1 = lambda x, y: pow(x * y - t, 2) - 4 * n * x * y
f2 = lambda x, y, s: (t - x * y - s) / (2 * x)

for x in xrange(1, 3000):
    for y in xrange(1, 3000):
        print x, y
        if f1(x, y) >= 0:
            s, b = iroot(f1(x, y), 2)
            if b:
                if prime(f2(x, y, int(s))):
                    print "Success"
                    print f2(x, y, int(s))
                    exit()
```

一分钟左右即可爆破出q

另一种方法：



n=p*q
n1=nextprime(p)*nextprime(q)=p1*q1
u=n*n1=p*q*p1*q1
yafu分解u可得
t1=p*q1,
t2=q*p1
p=gcd(t1,n)
q=gcd(t2,n)

比较合理的思路

9:47:38

p*nextprime(q) 和 q*nextprime(p) 会比较接近

虽然比较麻烦，但分解那一步秒出

level 5

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[+] Level 4 Clear!
[+] Level 5 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# n=0x1daf9fab45ff83e751bf7dd1b625879b3a8c89d4a086e0806b31e2a2cc1c4c1bc8694db643acc4911f3d143c1951f006df9
# e=0x10001
# c=0xdcb90409d48b54a73e408d16f5df6d4cc49183cd47eb8ccbc27837fecdf902233979895e8d789a30ca13ff0d9f452321c62
@ m=$
```

下意识的就上yafu了（p，q相差过大或过小），秒解

```
QWB_nextrsa [master●] yafu "factor(@)" -batchfile ./n_yafu



=== Starting work on batchfile expression ===
factor(8953391589573037684542938831731813546596371535331966829603746043683226157169876411642055492211298
=============================================
fac: factoring 895339158957303768454293883173181354659637153533196682960374604368322615716987641164205549
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 C317
rho: x^2 + 2, starting 1000 iterations on C317
rho: x^2 + 1, starting 1000 iterations on C317
pm1: starting B1 = 150K, B2 = gmp-ecm default on C317
Total factoring time = 6.6791 seconds


***factors found***

P9 = 743675299
P309 = 1203938278118477306658929226010478740748974578397549658241875537092865868759999841226682384701780

ans = 1

eof; done processing batchfile
```

level 6

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[+] Level 4 Clear!
[+] Level 5 Clear!
[+] Level 6 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# n=0x7003581fa1b15b80dbe8da5dec35972e7fa42cd1b7ae50a8fc20719ee641d6080980125d18039e95e435d2a60a4d5b0aaa4
# e=0x3
# c=0xb2ab05c888ab53d16f8f7cd39706a15e51618866d03e603d67a270fa83b16072a35b5206da11423e4cd9975b4c03c9ee0d7
@ m=$
```

e只有3，这次应该是低加密指数攻击了

```
QWB_nextrsa [master●] cat smallE.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from gmpy2 import iroot

n=0x7003581fa1b15b80dbe8da5dec35972e7fa42cd1b7ae50a8fc20719ee641d6080980125d18039e95e435d2a60a4d5b0aaa42d
e=0x3
c=0xb2ab05c888ab53d16f8f7cd39706a15e51618866d03e603d67a270fa83b16072a35b5206da11423e4cd9975b4c03c9ee0d78a

i = 0
while True:
    if iroot(c + i * n, 3)[1] == True:
        print "Success!"
        print iroot(c + i * n, 3)
        break
    i += 1
QWB_nextrsa [master●] python smallE.py
Success!
(mpz(1040065794283452835234332386718771782674284350646994660717501540629408351835476084209765388377794921
```

level 7

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[+] Level 4 Clear!
[+] Level 5 Clear!
[+] Level 6 Clear!
[+] Level 7 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# n1=0xb4e9991d2fac12b098b01118d960eb5470261368e7b1ff2da2c66b4302835aa845dd50a4f749fea749c6d439156df6faf8
# e1=0x10001
# c1=0x3a10c58ed3e8f9eade48dad7d36518dabeeca3d169c848f3b4b2bb027220e13d8b071c55046b14213e966ad9c381e5cad9
# n2=0xc31344c753e25135d5eed8febaa57dd7020b503a5569bdd4ae6747b5c36436dc1c4d7ead77bfc1034748bcc630636bae1c
# e2=0x10001
# c2=0xbefa7d62f15cafc81d098fdd524411537e948d83266ef22848f44d2e43d1f1388a26bb21c8fb08b571c7cbd6630d6f2b40
@ m1=$
```

给了n1，c1，n2，c2，且n无法分解，尝试公约数分解

```
QWB_nextrsa [master●] cat gcdN.py
#!/usr/bin/env python
# -*- coding: utf-8 -*-
__Auther__ = 'M4x'

from libnum import gcd

n1=0xb4e9991d2fac12b098b01118d960eb5470261368e7b1ff2da2c66b4302835aa845dd50a4f749fea749c6d439156df6faf8d1
#  e1=0x10001
#  c1=0x3a10c58ed3e8f9eade48dad7d36518dabeeca3d169c848f3b4b2bb027220e13d8b071c55046b14213e966ad9c381e5cad
n2=0xc31344c753e25135d5eed8febaa57dd7020b503a5569bdd4ae6747b5c36436dc1c4d7ead77bfc1034748bcc630636bae1c8f
# e2=0x10001
# c2=0xbefa7d62f15cafc81d098fdd524411537e948d83266ef22848f44d2e43d1f1388a26bb21c8fb08b571c7cbd6630d6f2b40

print "p -> {}".format(gcd(n1, n2))
print "q1 -> {}".format(n1 / gcd(n1, n2))
print "q2 -> {}".format(n2 / gcd(n1, n2))
QWB_nextrsa [master●] python gcdN.py
p -> 1725568696754776279984980552098360717842471500051715632277468961561228721883664092077858616916298226
q1 -> 132351070426725062043554691080648210190952108157658335988407251230007075283172499240825840919032041
q2 -> 142712204088308994057536283419724413794506016166476894328600394909477811164746138340181564452439035
```

level 8

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[+] Level 4 Clear!
[+] Level 5 Clear!
[+] Level 6 Clear!
[+] Level 7 Clear!
[+] Level 7 Clear!
[+] Level 8 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# c1=pow(m,e1,n),c2=pow(m,e2,n)
# n=0xace2aa1121d22a2153389fba0b5f3e24d8721f5e535ebf5486a74191790c4e3cdd0316b72388e7de8be78483e1f41ca5c93
# e1=0xac8b
# c1=0x9e84763bdbe246fad0a9cd52fda6233e6128a6210efaf3e6dea4fe272f78ad1f8f5cc7022f62f4f542341128e42d6fd10e
# e2=0x1091
# c2=0x9817fdc7b31a8f9cde1794096d3aa2bc6fe06fe34d4b7c9ca9a77982adf67fd4a7e636659553f4168a16757dc3a75e54ff
@ m=$
```

一个n，多组c，e，采用共模攻击

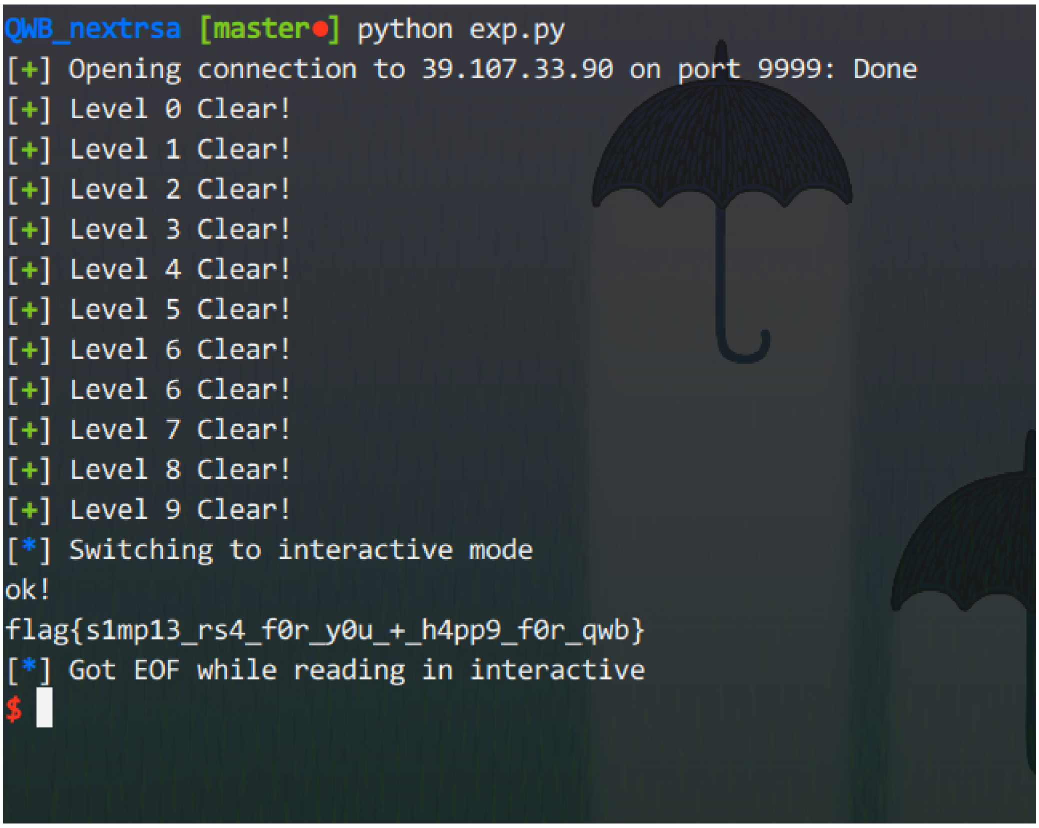
共模攻击写在最后的脚本里了

level 9

```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 0 Clear!
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[+] Level 4 Clear!
[+] Level 5 Clear!
[+] Level 6 Clear!
[+] Level 6 Clear!
[+] Level 7 Clear!
[+] Level 8 Clear!
[*] Switching to interactive mode
ok!
=next-rsa=
# c1=pow(m,e,n1),c2=pow(m,e,n2),c3=pow(m,e,n3)
# e=0x3
# n1=0x43d819a4caf16806e1c540fd7c0e51a96a6dfdbe68735a5fd99a468825e5ee55c4087106f7d1f91e10d50df1f2082f0f32
# c1=0x5517bdd6996b54aa72c2a9f1eec2d364fc71880ed1fa8630703a3c38035060b675a144e78ccb1b88fa49bad2ed0c6d5ad0
# n2=0x60d175fdb0a96eca160fb0cbf8bad1a14dd680d353a7b3bc77e620437da70fd9153f7609efde652b825c4ae7f25decf14a
# c2=0x3288e3ea8c74fd004e14b66a55acdcbcb2e9bd834b0f543514e06198045632b664dac3cf8578cde236a16bef4a1246de69
# n3=0x280f992dd63fcabdcb739f52c5ed1887e720cbfe73153adf5405819396b28cb54423d196600cce76c8554cd963281fc4b1
# c3=0xb0c5ee1ac47c671c918726287e70239147a0357a9638851244785d552f307ed6a049398d3e6f8ed373b3696cfbd0bce1ba
@ m=[*] Got EOF while reading in interactive
$
```

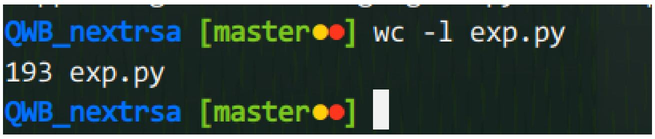
e = 3，给了三组数据，使用广播攻击，广播攻击也写在最后的脚本里了

flag



```
QWB_nextrsa [master●] python exp.py
[+] Opening connection to 39.107.33.90 on port 9999: Done
[+] Level 0 Clear!
[+] Level 1 Clear!
[+] Level 2 Clear!
[+] Level 3 Clear!
[+] Level 4 Clear!
[+] Level 5 Clear!
[+] Level 6 Clear!
[+] Level 6 Clear!
[+] Level 7 Clear!
[+] Level 8 Clear!
[+] Level 9 Clear!
[*] Switching to interactive mode
ok!
flag{s1mp13_rs4_f0r_y0u_+_h4pp9_f0r_qwb}
[*] Got EOF while reading in interactive
$ 
```

最终脚本



```
QWB_nextrsa [master●●] wc -l exp.py
193 exp.py
QWB_nextrsa [master●●] 
```

太长，就不贴出来了，放到github上了

reference

https://err0rzz.github.io/2017/11/14/CTF%E4%B8%ADRSA%E5%A5%97%E8%B7%AF/#%E5%85%B1%E6%

http://www.cnblogs.com/pcat/p/7508205.html

https://github.com/pablocelayes/rsa-wiener-attack

https://github.com/mimoo/RSA-and-LLL-attacks

more

初次之外，还见过已知p高位的，可以参考whctf-untitled

以及修复证书的，参考Jarvis OJ 600分的RSA

转载于:https://www.cnblogs.com/WangAoBo/p/8654120.html