2020程安 Crypto HW2/lab write-up

tags: 程式安全 CTF write up

lab - COM

Source code分析

```
p  = getPrime(512)
q1 = getPrime(512)
q2 = getPrime(512)
n1 = p * q1 # n1 have factor p
n2 = p * q2 # n2 have factor p
            # n1 & n2 have common factor -> gcd(n1, n2) = p
```

- 可得知$n1, n2$有共同質因數p
  $\Rightarrow$ 因為$q1, q2$都是質數,因此$gcd(n_1, n_2) = p$

Exploit

```
from math import gcd
from Crypto.Util.number import inverse, long_to_bytes
n1 = 766872792577986409478589754248056325301363238055493060347064347461
n2 = 757516261690267982864299157435654877483360721387193271710913038554

p = gcd(n1,n2) # gcd(n1,n2) = p

c = 563640730013381318277643162879212930517677018234314052805734038771
q1 = n1 // p # p*q1 = n

phi_n = (p - 1) * (q1 - 1)

e = 65537
d = inverse(e, phi_n) # ed = 1 (mod phi(n))
assert (e*d)%phi_n == 1

m = pow(c, d, n1) # decrypt

print(long_to_bytes(m))

# FLAG{XhupkK0Kx2D0dRov3PwJ}
```

lab - STE

Source code分析

```
def pad(data, block_size):
    if len(data) > block_size - 2:
        raise ValueError("message too big")
    return b'\x00' + b'\xff' * (block_size - 2 - len(data)) + b'\x00' +
```

- 可知padding的方式固定: \x00\xff\xff...\xff\x00

```
assert len(FLAG) == 16 # flag_len = 16
...
m = bytes_to_long(pad(FLAG, 128))
```

- 知道flag長度為16
- 明文會被補成128byte
  ⇒ 想法: 知道padding又知道flag長度很小,可以暴力try

Exploit

```
from Crypto.Util.number import *

n = 485818314067959942970843878759080733133516133094096998022128495414
c = 245996174830426295782772851211739725049120378594639860469097404094

P.<x> = PolynomialRing(Zmod(n)) # 設定x在Z_n中

padding = b'\x00' + b'\xff' * (128 - 2 - 16) + b'\x00'
padding = bytes_to_long(padding) << (8 * 16) # flag = 16byte * 8 bit

f = (padding + x) ^ 3 - c   # 列出方程式 f = m^e - c = 0 (mod n)

roots = f.small_roots() # 求方程式根

if roots: #如果有解,就是明文
    root = roots[0]
    flag = long_to_bytes(root)
    print(flag)

# FLAG{0htcMXzU3a}
```

HW2 - LSB

Source code分析

```
def main():
    p = getPrime(512)
    q = getPrime(512)
    n = p * q
    e = 65537
    d = inverse(e, (p - 1) * (q - 1))

    m = bytes_to_long(pad(FLAG, 128))
    c = pow(m, e, n)
    print(f'n = {n}')
    print(f'c = {c}')

    while True:
        c = int(input())
        m = pow(c, d, n)
        print(f'm % 3 = {m % 3}')
```

- 由最後一行可知,這個server給密文會回傳解密後mod3結果

想法

- $c = m^e$ 兩邊同乘3^e
  $\Rightarrow 3^e \cdot c = 3^e \cdot m^e = (3m)^e \ (mod\ n)$
  故若拿$3^e \cdot c$傳給server, 由server解密之後得到$3m\%n$,並回傳$3m\%n \ (mod\ 3)$

- 因為$0 \leq m < n$,故可知$0 \leq 3m < 3n$,分成三種狀況討論,剛好對應三種回傳值(0,1,2)
  - $0 \leq 3m < n \Leftrightarrow 3m\%n = 3m \rightarrow 3m\%3 = 0$
  - $n \leq 3m < 2n \Leftrightarrow 3m\%n = 3m - n \rightarrow (3m - n)\%3 = -n$
  - $2n \leq 3m < 3n \Leftrightarrow 3m\%n = 3m - 2n \rightarrow (3m - 2n)\%3 = -2n$

  $\Rightarrow$ 這樣可以獲得一個m的範圍

- 接著第二輪兩邊同乘9^e,得到$9m\%n \ (mod\ 3)$
  假設第一輪得到的範圍為$n \leq 3m < 2n \rightarrow 3n \leq 9m < 6n$,再繼續分成三種狀況討論
  - $3n \leq 9m < 4n \Leftrightarrow 9m\%n = 9m - 3n \rightarrow (9m - 3n)\%3 = 0$
  - $4n \leq 9m < 5n \Leftrightarrow 9m\%n = 9m - 4n \rightarrow (9m - 4n)\%3 = -n$
  - $5n \leq 9m < 6n \Leftrightarrow 9m\%n = 9m - 5n \rightarrow (9m - 5n)\%3 = -2n$

⇒ 又可以獲得一個m的範圍

- 多做幾輪就可以把範圍壓到更小

exploit

```
from pwn import *
from Crypto.Util.number import * # for long_to_bytes()

#r = process("./server.py")
r = remote('140.112.31.97', 30001)

n = int(r.recvline()[4:])
c = int(r.recvline()[4:])
e = 65537

def oracle(x): # return 3^k*m % 3
    r.sendline(str(x))
    remainder = int(r.recvline()[8:])
    return remainder

L = 0 # lower bound
H = 1 # higher bound
i = 0 # 第幾round

while (n * H // (3**i)) - (n * L // (3**i)) >= 2:
    i += 1
    s = c * pow(3, i*e, n) % n # 算(3^ie)c

    m = oracle(s) #傳給server,m為回傳值,i.e.(3^i)*m % 3
    L *= 3
    H *= 3

    if m == 0:
        H -= 2
    elif m == (-n % 3):
        L += 1
        H -= 1
    else:               # -2n % 3
        L += 2

print(long_to_bytes(n * L // (3**i))) # 取整數
print(long_to_bytes(n * H // (3**i))) # 取整數

r.interactive()
```

- 最後可以得到一個範圍,如果不夠小,再考慮所有可能性

```
peanut0203@ncuctf:~$ python3 sol.py
[+] Opening connection to 140.112.31.9
b'\xaa@\x87=\x17o|\xcb\x06MO\xe2\x01\x
\x07\xba#\x04h\xdbw;\xa0\xd4\xc5\x13\x
99\xa3I\xbb\xc7\x0c\x8el\xb9\xc5Qm\xa1
xa7\x00FLAG{nF9Px2LtlNh5fJiq3QtG|'
b'\xaa@\x87=\x17o|\xcb\x06MO\xe2\x01\x
\x07\xba#\x04h\xdbw;\xa0\xd4\xc5\x13\x
99\xa3I\xbb\xc7\x0c\x8el\xb9\xc5Qm\xa1
xa7\x00FLAG{nF9Px2LtlNh5fJiq3QtG~'
```

> lower bound取整之後的最後一個byte = ord('|') = 124
> upper bound取整之後的最後一個byte = ord('~') = 126
> 因此最後一byte有可能為124-126,且FLAG的pattern最後通常為'}'
> chr(125) = '}'

- 故flag: `FLAG{nF9Px2LtlNh5fJiq3QtG}`

HW2 - RSA

Source code分析

```
1  def pad(data, block_size):
2      padlen = block_size - len(data) - 2
3      if padlen < 8:
4          raise ValueError
5      return b'\x00' + bytes([random.randint(1, 255) for _ in range(padle
```

- 可知padding方式為隨機

```
1  p = getPrime(512)
2  q1 = next_prime(2 * p) # find next prime > 2p
3  q2 = next_prime(3 * q1) # find next prime > 3q1
4
5  n = p * q1 * q2
```

- 令$q1 = 2p + x,\ q2 = 3q1 + y = 6p + 3x + y$
  $\Rightarrow n = p(2p + x)(6p + 3x + y) = 12p^3 + (12x + 2y)p^2 + (3x^2 + xy)p$

想法

- 由質數定理,兩個質數的差不會超過1500,因此x跟y相比起p,都很小
  $\Rightarrow n > 12p^3$
- 可以先求p的近似值,然後算出$q1, q2$及$p \cdot q1 \cdot q2$跟n的大小關係,再往下或往上找p

```
p_est = next_prime(int(pow(n//12,1/3)))
q1 = next_prime(2 * p_est)
q2 = next_prime(3 *q1)

if (n <= p_est*q1*q2):
    for p in range(p_est, 0, -1):
        if (is_prime(p)):
            q1 = next_prime(2 * p)
            q2 = next_prime(3 *q1)
            if (p*q1*q2 == n):
                print(f'p is {p}')
                break
else:
    while(True):
        p = next_prime(p_est)
        q1 = next_prime(2 * p)
        q2 = next_prime(3 *q1)
        if (p*q1*q2 == n):
            print(f'p is {p}')
            break
```

- 找到p之後就可以利用e計算d跟$\phi(n)$,最後利用d求明文

Exploit

```python
from Crypto.Util.number import *
from gmpy2 import *

n = 220017788745427743154843924811157115392811047407235178284613606119
c = 106738266822232052382413255561332423985743815185522531628217640245
e = 65537
# n = p * q1 * q2 = p(2p+x)(6p+3x+y) = 12p^3 + 12p^2x + 2p^2y + 3px^2 +
#                                    = 12p^3 + (12x+2y)p^2 + (3x^2+xy)p
#                                     > 12p^3

'''find p'''
p_est = next_prime(int(pow(n//12,1/3)))
q1 = next_prime(2 * p_est)
q2 = next_prime(3 *q1)
if (n <= p_est*q1*q2):
    for p in range(p_est, 0, -1):
        if (is_prime(p)):
            q1 = next_prime(2 * p)
            q2 = next_prime(3 *q1)
            if (p*q1*q2 == n):
                print(f'p is {p}')
                break
else:
    while(True):
        p = next_prime(p_est)
        q1 = next_prime(2 * p)
        q2 = next_prime(3 *q1)
        if (p*q1*q2 == n):
            print(f'p is {p}')
            break

'''calculate d'''
phi_n = (p-1) * (q1-1) * (q2-1)
d = inverse(e, phi_n)

'''decrypt c'''
m = pow(c, d, n)
print(f'm is {long_to_bytes(m)}')
```

```
p is 122393639688623016550326718894086783363651977652907222495887682276491406899488728167253064168252425926545908260284435352973447178087243161450043008604209 99
m is b'\xdfM\xb5\x94$\']\x87\x05\xc1\xad\x9a\xbe\xc4o\x1bL\xa2>\xab\x89\xfc\x13\x91v\xd8S\\f\x1d\x86Q[%\xba\xc3||\xcf\xe9\xd0\xa9\xb4D\x1e\xc9m\x98q$_\xff\x9e\x82\x84\xd1-\xb7\n\xf4?\xfd"\x90\x1c\x0fCix\xf7xh4\xc2aV\x89\xd4\'p\xe3\xff\x11\xe7MR\xa0\xd9DD\xf0\xe8i\x89\x1fs\xef\xafhL\x01<\xf9w\xa8\x17`\x1aQ\xa6\x1f>a\x9e\xfbp\x9d\x12^\x07\x9d\xae\x06\x8bZ`ea\xc2~\xf4\x15\xc5\x03K+?\xce\x8e\x05\x03\x05\xe3\xde\x0e;\xcc\x96wwY\xfe?\x9f\xeeH\xf0cy\xe8)\xf5\x01\xc4\x00FLAG{Ew9xeANumjDr6bXemHsh}'
```