hw2 Writeup

School/Grade: 交大資科工所 碩一
Student ID: 309551004 (王冠中)
ID: aesophor

RSA

- **Overview**:

  In this challenge, the steps I took are listed as follows:

  1. use binary search to find out the original `p` where `n = p * q1 * q2`
  2. calculate `d` where `d = e^(-1) mod (p-1)(q1-1)(q2-1)`
  3. decrypt the cyphertext `c` with the private key `(n,d)`

- **Observation**:

  - `getPrime(n)` is a function from the package `pycryptodome` which generates a random n-bit prime number.

  - for every `p` we pick, `q1` and `q2` can be deduced

    ```
    p  = getPrime(512)
    q1 = next_prime(2 * p)   # q1 depends on p
    q2 = next_prime(3 * q1)  # q2 depends on q1
    ```

  - since the domain of `p` is [2^511, 2^512 - 1], bruteforcing this entire domain can be really slow. Hence, we can employ binary search to find out the correct `p`.

- **Exploitation**:

```
left = 2 ** 511
right = 2 ** 512 - 1


while left <= right:
    mid = left + int((right - left) / 2)

    p  = mid
    q1 = next_prime(2 * p)
    q2 = next_prime(3 * q1)
    n  = p * q1 * q2

    if n > real_n:
        right = mid - 1

    elif n < real_n:
        left = mid + 1

    else:  # n == real_n
        d = inverse(e, (p - 1) * (q1 - 1) * (q2 - 1))
        m = pow(real_c, int(d), n)
        print(long_to_bytes(m))
        break
```

- **Flag**:

  flag: `FLAG{Ew9xeANumjDr6bXemHsh}`
  exploit: please see the attachment

LSB (LSB Oracle Attack)

- **Overview**:

  1. Modify the exploit: https://github.com/oalieno/Crypto-Course/blob/master/RSA/LSB-Oracle/solve2.py (https://github.com/oalieno/Crypto-Course/blob/master/RSA/LSB-Oracle/solve2.py)
  2. Change all base2-related stuff to base3

- **Flag**:

  flag: `FLAG{nF9Px2LtlNh5fJiq3QtG}`

exploit:

```
#!/usr/bin/env python3
from pwn import *
from Crypto.Util.number import *

r = remote('140.112.31.97', 30001)

n = int(r.recvline().split(b' = ')[1])
c = int(r.recvline().split(b' = ')[1])
e = 65537

inv  = inverse(3, n)
inve = pow(inv, e, n)

flag, x = 0, 0
for i in range(1024):
    r.sendline(str(c))
    m = int(r.recvline().split(b' = ')[1])
    bit = (m - x) % 3
    x = inv * (x + bit) % n
    flag += bit * pow(3, i)
    print(long_to_bytes(flag))
    c = (c * inve) % n

print(long_to_bytes(flag))
```