0yster HGAME 2019 week-3 writeup

又苟了一周，流下既羞耻又快乐的泪水。

web-sqli1

手注，但是可能是防止sqlmap一把梭的情况加了截取字符串md5前4位做验证码的判断。

所以首先要计算code。

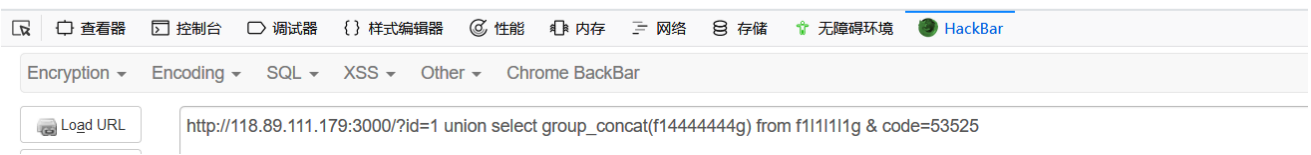
```
#coding=utf-8
import md5
for i in range(0,1000000):
    m1=md5.new()
    i=str(i)
    m1.update(i)
    code=m1.hexdigest()
    if code[0:4]=='0815':
        break

print i
print code
```

首先通过加'判断应该是数字型的注入无需闭合

然后通过order by 确定只有一个字段，于是union select 1 在1的位置上做手脚一路爆出数据库，表名，字段名和值。

substr(md5($_GET["code"]),0,4) === 0f8e
array(1) { ["word"]=> string(7) "welcome" } array(1) { ["word"]=> string(26) "hgame{sql1_1s_iNterest1ng}" }



crypto-babyRSA

rsa初体验，因为目的是学所以数学渣花了很久研究rsa算法的来龙去脉和基础概念，然后自己照着公式写了一个算d的脚本，但是跑出来的m转成16进制再转成字符串始终是乱码。然后我头铁地又换了种方式写发现还是一样的结果。去请教学长才知道到e和phi并不互素，一下子有点懵，因为看到的介绍正常的加密流程是必须挑选与phi互素的e。

谷歌后初步判断是Rabin加密。再谷歌发现一篇生动且思路简单清晰的文章。

http://blog.sina.com.cn/s/blog_64370f500100lhqz.html

但是实现起来无从下手，所以解题脚本是找的改了数据直接跑出flag

```
#-*- coding:utf-8 -*-
from Crypto.Util.number import *
import sympy
def gcd(a,b):
    if a < b:
        a,b = b,a
    while b != 0:
        tem = a % b
        a = b
        b = tem
    return a

def invalidExponent(p,q,e,c):
    phiN = (p - 1) * (q - 1)
    n = p * q
    GCD = gcd(e, phiN)
    if (GCD == 1):
        return "Public exponent is valid....."
    d = inverse(e//GCD,phiN)
    c = pow(c, d, n)
    plaintext = sympy.root(c, GCD)
    plaintext = long_to_bytes(plaintext)
    return plaintext

def main():
    p = 583800044303078033678069964607731236037903057890983844889520562066157682745 27
    q = 818595269757200606493800981936716128012005050291270765394576801554876696228 67
    e = 12
    c =
20608721532369020246787892668194449176965915672645869081591928616363088644729157051019617158562614360898838461518592175240938078800647657633741013644746 0

    plaintext = invalidExponent(p,q,e,c)
    print plaintext
main()
```