Web Attack HW3

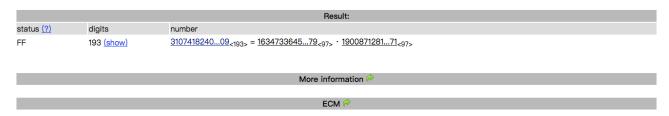
RSA_1

sol:

1. First get the information in the public key, obtaining N, e

```
popenssl rsa -pubin -inform PEM -text -noout < public-key.pem
Public-Key: (640 bit)
Modulus:
    00:ae:5b:b4:f2:66:00:32:59:cf:9a:6f:52:1c:3c:
    03:41:01:76:cf:16:df:53:95:34:76:ea:e3:b2:1e:
    de:6c:3c:7b:03:bd:ca:20:b3:1c:00:67:ff:a7:97:
    e4:e9:10:59:78:73:ee:f1:13:a6:0f:ec:cd:95:de:
    b5:b2:bf:10:06:6b:e2:22:4a:ce:29:d5:32:dc:0b:
    5a:74:d2:d0:06:f1
Exponent: 65537 (0x10001)</pre>
```

2. Use factordb to factorize the number N get p,q



3. Use formula step by step get the value of r, d and then decrypt the message.

```
q = 1900871281664822113126851573935413975471896789968515493666638539088027103802104498957191261465571
In [11]: r = (p-1)*(q-1)
In [12]: def inv_mod(a, b):
            li = [a, b]
            p, q = a, b
while p != 1 and q != 1:
               r = p % q
                li.append(r)
               p = q
q = r
            li.remove(1)
            li.reverse()
            k, m = 1, -(li[1]//li[0])
for i in range(len(li)-2):
               m0 = m
               m = -(li[i+2]//li[i+1])*m0 + k

k = m0
            return m%a
In [13]: d = inv_mod(r,e)
In [20]: n_s = '02797196da55a20737a066fae17f0f0cbcdd4lacf84c91f33ac34d7lecc77614ef23fcd53c6098aae4ddb8429f4bca3fe8f79c510f9180e5
In [21]: for i in range(len(n_s)//2):
            print(chr(int(n_s[2*i:2*(i+1)], 16)), end='')
        yqÚU¢7 fúá½ÝA¬øLó:ĀMqìÇvï#üŌ<`&äÝ,BKÊ?è÷QåQº&kFLAG_IS_WeAK_rSA
```

4. The decrypted message look like something strange and followed by "FLAG IS WeAK rSA"

RSA_2 sol:

1. Followed the step exactly like the rsa_1 but with clearly value.

```
In [28]: c = 8320829899517460417477359029820363936054002487125612689288966134574240331492986193910049266660564731664657648652621

In [29]: p = 9648423029010515676590551740010426534945737639235739800643989352039852507298491399561035009163427050370107570733633

In [30]: q = 1187484383798029703209240584865365685276091015454338090765004019070428335890920857825106304773244399223064790388751

In [31]: e = 65537

In [32]: r = (p-1)*(q-1)
n = p*q
```

2. Get the decrypted value 5577446633554466577768879988

Stego_1 sol:

According to 毛弘仁, I found the exact the same problem on the CTF. And the website also provide the source code to solve the problem. By the source code, it seems that inside the png format, there are some space could hide the information.

```
python png_filter.py stego1.png
PNG Signature: ('\x89', 'P', 'N', 'G', '\r', '\n', '\x1a', '\n')
Pos : 8
Type: IHDR
Size: 13
CRC : 5412913F
Pos : 33
Type: IDAT
Size: 10980
CRC: 98F96EEB
Pos: 11025
Type: IEND
Size: 0
CRC: AE426082
Data length in PNG file :
Decompressed data length:
                               10980
                               1920800
Flag: DrgnS{WhenYouGazeIntoThePNGThePNGAlsoGazezIntoYou}
```

Strgo_2 sol:

1. Use binwalk to checkout file, it seems that a zip file is hided behind the jpg file.

```
binwalk stego2.jpg

DECIMAL HEXADECIMAL DESCRIPTION

0 0x0 JPEG image data, JFIF standard 1.01
40804 0x9F64 Zip archive data, at least v2.0 to extract, compressed size: 32993, uncompressed size: 33783, name: got2.jpg
73941 0x120D5 End of Zip archive
```

2. split the zip file and then extract it, get "got2.jpg"

