

# WRITE UP FREEPASS POROS 2024

## JersYY

Steven Anthony  
(235150201111035)  
Teknik Informatika

# Table of Content

Freepass POROS 2024 :

1. Bonus
  - a. wassup
2. Crypto
  - a. Patience
  - b. Acaruto
3. Forensic
  - a. dont\_open it
4. Web
  - a. messi

# BONUS

wassup

Deskripsi soal:

Challenge

13 Solves

×

wassup

0

for those who confused, the template for all challenges's flag is *freepass{fl4g}*

FLAG = *freepass{thi5\_1s\_4\_b0nus}*

Flag


Submit

Chall ini hanya memberi template flag dalam freepass ini  
flag: `freepass{thi5_1s_4_b0nus}`

# CRYPTOGRAPHY

## Patience

Deskripsi Soal :



Challenge 5 Solves

patience

100

Download chall.py Download output.txt

Flag Submit

Pada chall ini diberikan 2 file

Output.txt :

```
n =
2707066822078432357828817765137972038985124672672911322763998
0014076927289047934775811165694097995160575719001058206347597
3227944354502793846459514670851808221800475202824163175619599
6325873413503085603473523258857872264117027754650975637062243
5353940264961543756671025475490937797659821078709335380276887
2785768384421028370558231931083961136432876601919351650201353
4235986053631365237778601098914327884844088630855245133660188
4395415694581562870178647008782055229634214236363051235967968
9833673336446284679240732402159414606852605238599591964375838
0521266065370550790290308479925531693905005026760415565790467
5685805914326713927882142788618195581120047229740147046106849
2522532776630380541370420288558035193934214060990347547513331
1513473674160226077663010710511835659573061905206232942452585
8606141133055684444187647441707105673354997344420375325955466
9933287998296292615420336491293731265582609385219385780060162
8938146041

z =
4102787099014762875195010150634045265907185197662160905982885
5685245097406060957243281419025382723141685780888513262217794
7514445589218867852189497174444731331852537349498064043744685
2944120425757079934962419836547151908807528907907749131742050
1418789502781173508373890369075030372361337957754500050769707
1566929945837950628704824370126932547208410725575305551941349
0065220289106767624452240305113154971943161391186255888582188
4969745501184722408150324174621846179265034604669417745136570
3000270258044124651670306358263098313993670287801184647255713
0628369949880504858402606744375631655624145484352744364233972
```

```
4956985
c =
1163949686563357231154670903975415440127974530550694243068448
1285829133569835437174288743218822485278157672077469750425626
4102715427544312774188977718910626955275393839506953875503174
6273580986729168465290852426811276669505183040587626572460963
8173344386188877950233202027464614113110413913560992254226095
8552452985170837463764877486030189737061843030317805706135865
7341961861741848393126160255390995356623675240578054999500975
3286899788956118935444274341406862104040996672241764180688498
2061814194083829325289602226332263292322993376029125881073894
0762392864480491351696375305247031773396437972028623605795805
7273926
```

Chall.py :

```
from Crypto.Util.number import getPrime, bytes_to_long
from math import gcd
from secret import FLAG

def check(a, b):
    if b == 0:
        return a != 1
    else:
        return check(b, a % b)

def main():
    p = getPrime(1024)
    q = getPrime(1024)
    n = p * p * q
    e = 0x10001
    m = bytes_to_long(FLAG)
    c = pow(m, e, p*q)
    z = sum(check(i, n) for i in range(n))
    print(f'{n = }\n{z = }\n{c = }')

main()
```

Sepertinya pada challenge ini kita diminta untuk mendecrypt pesan yang terenkripsi menggunakan RSA, tetapi terdapat kesalahan dalam parameter yang digunakan untuk enkripsi.

Setelah memahami file yang diberikan, didapatkan beberapa poin penting

1. 'p' dan 'q' adalah bilangan prima dengan jumlah 1024 digit dan 'n' didefinisikan dengan  $p \cdot p \cdot q$
2.  $e = 65537$
3. c (Ciphertext) adalah pesan yang diencrypt dengan menggunakan metode RSA dengan rumus  $\text{pow}(m, e, p \cdot q)$ , yang mana ini seharusnya menggunakan  $n = p \cdot p \cdot q$
4. z adalah jumlah nilai dari i dimana i tidak terbagi dengan n yaitu, bukan coprime dengan n.

Setelah melakukan pencarian lebih mendalam, kita bisa menggunakan konsep fungsi totient Euler, " $\phi(n)$ ", yang merupakan jumlah bilangan yang coprime dengan n. Berdasarkan definisi  $n = p^2 \cdot q$ , fungsi totient dari n adalah  $\phi(n) = p \cdot (p - 1) \cdot (q - 1)$ .

$z = n - \phi(n)$  dapat kita gunakan karena kita dapat memanfaatkan "z", yang merupakan n dikurangi jumlah bilangan coprime dengan  $n(\phi(n))$ .

Maka, solusi yang dapat kita lakukan adalah:

1. Tentukan p dan q dengan menggunakan metode aproksimasi atau metode pendekatan dengan persamaan akar n
2. Gunakan nilai n dan z yang diberikan untuk memecahkan persamaan yang ada diatas dengan mencari nilai p dan q melalui faktorisasi.
3. Hitung kembali nilai bilangan yang coprime dengan  $n(\phi(n))$  dengan menggunakan p dan q yang telah ditemukan, yaitu  $\phi(n) = (p - 1) \cdot (q - 1)$ .
4. Hitung nilai private key RSA d sebagai invers dari e modulo  $\phi(n)$  menggunakan  $\text{inverse}(e, \phi(n))$ .
5. Decrypt c menggunakan  $\text{pow}(c, d, p \cdot q)$  untuk mendapatkan hasil akhir.

Berikut adalah script yang saya gunakan:

```
!pip install pycryptodome
from sympy import symbols, Eq, solve
from Crypto.Util.number import *

n =
270706682207843235782881776513797203898512467267291
132276399800140769272890479347758111656940979951605
757190010582063475973227944354502793846459514670851
808221800475202824163175619599632587341350308560347
352325885787226411702775465097563706224353539402649
615437566710254754909377976598210787093353802768872
785768384421028370558231931083961136432876601919351
650201353423598605363136523777860109891432788484408
863085524513366018843954156945815628701786470087820
552296342142363630512359679689833673336446284679240
732402159414606852605238599591964375838052126606537
055079029030847992553169390500502676041556579046756
858059143267139278821427886181955811200472297401470
461068492522532776630380541370420288558035193934214
060990347547513331151347367416022607766301071051183
565957306190520623294245258586061411330556844441876
474417071056733549973444203753259554669933287998296
292615420336491293731265582609385219385780060162893
8146041

z =
410278709901476287519501015063404526590718519766216
090598288556852450974060609572432814190253827231416
857808885132622177947514445589218867852189497174444
731331852537349498064043744685294412042575707993496
241983654715190880752890790774913174205014187895027
811735083738903690750303723613379577545000507697071
566929945837950628704824370126932547208410725575305
551941349006522028910676762445224030511315497194316
139118625588858218849697455011847224081503241746218
461792650346046694177451365703000270258044124651670
306358263098313993670287801184647255713062836994988
050485840260674437563165562414548435274436423397249
56985
```

```

c =
116394968656335723115467090397541544012797453055069
424306844812858291335698354371742887432188224852781
576720774697504256264102715427544312774188977718910
626955275393839506953875503174627358098672916846529
085242681127666950518304058762657246096381733443861
888779502332020274646141131104139135609922542260958
552452985170837463764877486030189737061843030317805
706135865734196186174184839312616025539099535662367
524057805499950097532868997889561189354442743414068
621040409966722417641806884982061814194083829325289
602226332263292322993376029125881073894076239286448
049135169637530524703177339643797202862360579580572
73926
print('Using values: n =', n)
print('Using values: z =', z)

#ubah ke simbol
p, q = symbols('p q', integer=True)
eq = Eq(z, n - p * (p - 1) * (q - 1))

sol = solve(eq.subs(q, n / p**2), p)
print('Potential solutions found:', sol)
for s in sol:
    if s.is_prime:
        p_val = s
        q_val = n // (p_val**2)
        print('Potential p:', p_val)
        print('Potential q:', q_val)
        break
    else:
        print('Discarding non-prime solution:', s)
e = 0x10001
phi_n = (int(p_val) - 1) * (int(q_val) - 1)

#biasa
d = inverse(e, phi_n)
p_val_int = int(p_val)
q_val_int = int(q_val)
decrypted_m = pow(c, d, p_val_int * q_val_int)
flag =

```



```
decrypted_m.to_bytes((decrypted_m.bit_length() + 7)
// 8, byteorder='big')
flag1 = flag.decode()
print('Flag:', flag1)
```

dan outputnya adalah:

```
Requirement already satisfied: pycryptodome in /usr/local/lib/python3.10/dist-packages (3.20.0)
Using values: n = 27070668220784323578288177651379720389851246726729113227639980014076927289047934775811165
Using values: z = 41027870990147628751950101506340452659071851976621609059828855685245097406060957243281419
Potential solutions found: [1526078622191735229242128574530086886188122757212164266686814158187810514934403
Potential p: 1526078622191735229242128574530086886188122757212164266686814158187810514934403914513242719380
Potential q: 1162372050895075214973998898398835580795140799449192961018611746275043214322508459012525923155
Flag: freepass{i_th1nk_n0W_u_und3rst4nd_h0w_r54_w0rk5}
```

Flag: freepass{i\_th1nk\_n0W\_u\_und3rst4nd\_h0w\_r54\_w0rk5}

# CRYPTOGRAPHY

Acaruto

Deskripsi Soal :

Challenge

1 Solve


✕

acaruto

100

connection using nc 10.34.4.172 2023

habis lapor sepuh langsung bisa netcat nya :< lanjut ngerjain ges

 chall.py

Flag

Submit

Pada chall ini, diberikan sebuah file code python dan netcat, yang mana ketika saya menghubungkan server menggunakan netcat :

```
(steven@steven)-[~]  
$ nc 10.34.4.172 2023  
Select one:  
1) Encrypt  
2) Check  
>> 
```

lalu ketika membuka source code nya:

Chall.py

```
from Crypto.Util.number import *  
  
FLAG = open('flag.txt').read()  
ACARUTO = 'Hey it\'s me Acaruto! I am the pillar of life. You  
don\'t know me, but I know everyone'.encode()  
  
def encrypt(e, n):  
    try:
```

```

        msg = input('Enter the message: ')
        assert msg.isascii()
    except AssertionError:
        print('Invalid input!')
    else:
        if msg.encode() == ACARUTO:
            print('Not allowed!')
            return

        enc = pow(bytes_to_long(msg.encode()), e, n)
        print('Encrypted message in hex:', hex(enc)[2:])

def check(d, n):
    try:
        enc = int(input('Enter the message in hex: '), 16)
    except ValueError:
        print('Invalid input!')
    else:
        if enc > n:
            print('Not allowed!')
            return

        msg = pow(enc, d, n)
        if long_to_bytes(msg) == ACARUTO:
            print('You got it! Here\'s the flag:', FLAG)
        else:
            print('Not yet :(')

def main():
    while True:
        try:
            p = getPrime(1024)
            q = getPrime(1024)
            n = p * q
            e = 0x10001
            d = pow(e, -1, (p-1)*(q-1))
        except:
            continue
        else:
            break

    while True:
        try:
            choice = int(input('Select one:\n1) Encrypt\n2)
Check\n>> '))
            assert 1 <= choice <= 2
        except:
            print('Invalid choice!')

```

```

        break
    else:
        if choice == 1:
            encrypt(e, n)
        elif choice == 2:
            check(d, n)

if __name__ == '__main__':
    main()

```

Pada chall ini, kita diminta untuk mendapatkan hasil enkripsi dari ACARUTO agar bisa membuka file “flag.txt” tetapi dengan suatu masalah yaitu kita tidak dapat langsung mengenkripsinya karena terdapat code :

```

if msg.encode() == ACARUTO:
    print('Not allowed!')
    return

```

dan setiap kita menghubungkan server dengan netcat, kita akan mendapatkan hasil enkripsi yang berbeda pula karena terdapat perintah *getPrime(1024)*, sehingga kita tidak dapat membuat kode baru dengan memanipulasi kode diatas karena hasilnya tentu akan berbeda.

Setelah dipahami lebih lanjut source code ini, kita dapat mencari variabel ‘n’ dengan rumus awal :

$$C = M^e \pmod{N}$$

Lalu, kita bisa mendapatkan cipher text sesuai yang kita inginkan dengan cara menginput plaintext sembarang/bebas.

Pada kasus ini, saya mencoba memasukkan plaintext “test” dan mendapatkan hasil sebagai berikut

```

(steven@steven)-[~]
$ nc 10.34.4.172 2023
Select one:
1) Encrypt
2) Check
>> 1
Enter the message: test
Encrypted message in hex: 2dac960139e1e3ddd48452777c3cae37604faae3eba8424480fc9cc942179
4f8902735d709b68594ff361a5e7262848e94e5603da5a06b122c94b174116f9c041b856b9e2469348c513d
9aead04c55ebcb0e18234ee0911acb7885c864d39a6f2dc2af191bf894bb1513181c

```

setelah itu, saya ubah nilai hex tersebut menjadi desimal dan mendapatkan angka berikut:

```
5765829030397448105391067684363475797834487047055125696200309
2040572930418543689318379628094167330546819610598164784315834
1528617239475771408146429803604709551980912554826786964430173
5929198608314913827173007009045370688870515219901830271979259
8843868148436979623192792555363950677212468372869229268442753
9245812993962968831708303606516341846213905760759459449891102
7521524581814711592127405429195614010771867144863015409225313
2527447104611038948696259179435080549668124183494809449206977
3107921541869885983360435390941921874212086759572562237386083
6350390460729741420194311902764373055419639097612788715747800
193052
```

Setelah itu, kita hanya juga perlu mengubah kata 'test' menjadi desimal dengan code:

```
dec = bytes_to_long('test'.encode())
print(dec)

1952805748
```

lalu kita hanya perlu memodifikasi rumus ini agar bisa mendapatkan n dengan beberapa variabel yang telah kita ketahui yang mana;

```
C =
5765829030397448105391067684363475797834487047055125696200309
2040572930418543689318379628094167330546819610598164784315834
1528617239475771408146429803604709551980912554826786964430173
5929198608314913827173007009045370688870515219901830271979259
8843868148436979623192792555363950677212468372869229268442753
9245812993962968831708303606516341846213905760759459449891102
7521524581814711592127405429195614010771867144863015409225313
2527447104611038948696259179435080549668124183494809449206977
3107921541869885983360435390941921874212086759572562237386083
6350390460729741420194311902764373055419639097612788715747800
193052
M = 1952805748
E = 65537
```

Terlihat mudah bukan? ternyata tidak...

Setelah berfikir selama 45 menit, akhirnya saya menyerah dan mencari beberapa referensi yang ada hingga menemukan :

```
import gmpy2

"""
@param pairings
    list: [(pt1, ct1), (pt2, ct2), ..., (ptk, ctk)]
@param e
    int : encryption exponent
@return
    int : recovered N
"""
def recover_n(pairings, e):
    pt1, ct1 = pairings[0]
    N = ct1 - pow(pt1, e)

    # loop through and find common divisors
    for pt, ct in pairings:
        val = gmpy2.mpz(ct - pow(pt, e))
        N = gmpy2.gcd(val, N)

    return N
```

- In reality, you're likely to only need two or three (plaintext, ciphertext) pairings (in the context of ctf challenges and exercises), and as such computations can be manual if needed, but shouldn't be too complex
- As it's likely you'll be dealing with large numbers, overflows and precision errors may arise in code - using libraries like `gmpy` provide support for integers of (theoretically) infinite size, and some nice accompanying features too (like in-built gcd and efficient modular exponentiation)

Setelah dipahami, ternyata kita perlu 2 “Pair” atau lebih ( $M, C$ ) untuk mendapatkan hasil ‘n’ serta menggunakan library ‘gmpy2’ karena angka yang terlalu besar sehingga library ‘math’ tidak bisa mengeksekusinya

lalu kita hanya perlu membuat persamaan kedua

```
Select one:
1) Encrypt
2) Check
>> 1
Enter the message: test1
Encrypted message in hex:
4ecf13aaaedd6a19f415eaf9ad81017716985e35d9c7fd9d3bcbf45d3ec49
42578c10270cc8caa389504e73d2b74834da088537bf5b4095369edd0fc42
e92b43e9ecc88d6766dd8a8d9d81f62bc30a00942b1792d977945a5cd62c2
d3b3ffe07f301c38a384402bba1ca122e27ace8af2148474e5fdf96f81ef8
f353f0df5a742638baa8f7b4b979ddca8a4c85049baabe995650ec376ba7f
8bd4af06afc89bf691b226332bf2e387ad2a1ae7a49f1d49f4ee2ee20e49e
ec29ae5cab42980a59f0176b999e54494bca334b489c4ef31b01c038f8b38
0edbed5c0aa69ec45740aa62f8069e9c671dd9597b92b4a4697acf43b5900
8d67eef3d660edff3f8c0cec
```

Setelah itu kita ubah menjadi desimal

```
9948701174674235925136603982586147887157168676798252705649550
9435976447892894719927447647491579757881678337335788603433432
7848327935898846680292398832829985660172270403344739842397698
5978849138997779959733525688686603893372669659878231059927846
5541549243817240617243882114031192449496036518125095839858164
7142919321455196529827792726418436487833795235351570605843945
9160342937250268431265621378290611533634908918961735470063982
9423605079557064514233278343556167010171138520768275147324834
6289905410035350111776610016241044908182862699955260956788636
2594306774547901539428094455268874073105620280697996531109632
675052
```

dan 'test1' kita ubah menjadi desimal juga

```
dec = bytes_to_long('test1'.encode())
print(dec)

499918271537
```

lalu kita dapatkan persamaan kedua!

Kemudian kita cari value dari 'n' menggunakan script seperti diatas

```
e = 65537
pt1 = 1952805748
ct1 =
5765829030397448105391067684363475797834487047055125696200309
2040572930418543689318379628094167330546819610598164784315834
1528617239475771408146429803604709551980912554826786964430173
5929198608314913827173007009045370688870515219901830271979259
8843868148436979623192792555363950677212468372869229268442753
9245812993962968831708303606516341846213905760759459449891102
7521524581814711592127405429195614010771867144863015409225313
2527447104611038948696259179435080549668124183494809449206977
3107921541869885983360435390941921874212086759572562237386083
6350390460729741420194311902764373055419639097612788715747800
193052

pt2 = 499918271537
ct2 =
9948701174674235925136603982586147887157168676798252705649550
9435976447892894719927447647491579757881678337335788603433432
7848327935898846680292398832829985660172270403344739842397698
5978849138997779959733525688686603893372669659878231059927846
5541549243817240617243882114031192449496036518125095839858164
7142919321455196529827792726418436487833795235351570605843945
9160342937250268431265621378290611533634908918961735470063982
9423605079557064514233278343556167010171138520768275147324834
6289905410035350111776610016241044908182862699955260956788636
2594306774547901539428094455268874073105620280697996531109632
675052

N = ct1 - pow(pt1, e)
val = gmpy2.mpz(ct1 - pow(pt1, e))
N = gmpy2.gcd(val, N)
val = gmpy2.mpz(ct2 - pow(pt2, e))
N = gmpy2.gcd(val, N)

print('N :', N)
```

dan hasilnya adalah

```
N :
1792855827425644062321248714691025722577378247014616608744703
5041231414108154190403828387185712200199955562313393951763671
0574994571612196415836444593770204869726030047060738253635710
```



```
5083462183596827471654053803205664467215013710647512393586932
8360816118310142839648121485554519707474420256811483201456710
6769067792818853863306495720452053062158814071185684028745005
6474610760103857539413416911803942411608897645566458877694993
7609450906053536588761603904578241633217405348575381085974643
5518071841711149030563261308149390590129056031795836238767674
9435724061718752240680172537700698621795057184025962307760119
2893297
```

Setelah itu, saya mengenkripsi pesan “Hey it's me Acaruto! I am the pillar of life. You don't know me, but I know everyone” dengan memasukkan ‘n’ yang tadi saya dapatkan

Berikut adalah script yang saya buat:

```
!pip install pycryptodome
!pip install gmpy2
import gmpy2
from Crypto.Util.number import *
from gmpy2 import *
e = 65537
pt1 = 1952805748
ct1 =
5765829030397448105391067684363475797834487047055125696200309
2040572930418543689318379628094167330546819610598164784315834
1528617239475771408146429803604709551980912554826786964430173
5929198608314913827173007009045370688870515219901830271979259
8843868148436979623192792555363950677212468372869229268442753
9245812993962968831708303606516341846213905760759459449891102
7521524581814711592127405429195614010771867144863015409225313
2527447104611038948696259179435080549668124183494809449206977
3107921541869885983360435390941921874212086759572562237386083
6350390460729741420194311902764373055419639097612788715747800
193052
pt2 = 499918271537
ct2 =
9948701174674235925136603982586147887157168676798252705649550
9435976447892894719927447647491579757881678337335788603433432
7848327935898846680292398832829985660172270403344739842397698
5978849138997779959733525688686603893372669659878231059927846
5541549243817240617243882114031192449496036518125095839858164
7142919321455196529827792726418436487833795235351570605843945
9160342937250268431265621378290611533634908918961735470063982
9423605079557064514233278343556167010171138520768275147324834
6289905410035350111776610016241044908182862699955260956788636
2594306774547901539428094455268874073105620280697996531109632
675052
```

```

N = ct1 - pow(pt1, e)
val = gmpy2.mpz(ct1 - pow(pt1, e))
N = gmpy2.gcd(val, N)
val = gmpy2.mpz(ct2 - pow(pt2, e))
N = gmpy2.gcd(val, N)

print('N :', N)

msg = 'Hey it\'s me Acaruto! I am the pillar of life. You
don\'t know me, but I know everyone'
enc = pow(bytes_to_long(msg.encode()), e, N)
print('Encrypted message in hex:', hex(enc)[2:])

```

dan kita dapatkan hasil enkripsi dari “ACARUTO”

```

Encrypted message in hex:
6ea34657bbb8354638f4a0e602150f3e57740af87c7ff061c27bb92f3635c
7aaala4ce57d2f90fc82edeb565766ad93a04791fd182fe61ce3e0af71782
613655f7a8f9822bc5739e5d71d24bb258706d244b37f6b598710fead6fb6
75ce3b8143c3b838adfb262479527e248bd0f45b7cc6d1ff6cbe2e1b09fa
8a80aced4fb0ab157046feb1dfe50918c54721e14c74c55f2e35d2ee56947
b34f77efca8892487c6116fe5cb2e4f2f82e1bf7b477221475b488a801c55
5f676711d01b1f573b4e59cccb73a720848d0767ec60ecdedd15861612e46
0466df712647378b9381185206e36ed7526c4cf875efa873e0ac194a21b4d
6e446f31bda29d90953ea729

```

lalu kita hanya perlu check dengan server netcat yang telah saya hubungkan sebelumnya

```

Select one:
1) Encrypt
2) Check
>> 2
Enter the message in hex: 6ea34657bbb8354638f4a0e602150f3e57740af87c7ff061c27bb92f3635c7aaala4ce57d2f90fc82edeb565766ad93a04791fd182fe61ce3e0af71782613655f7a8f9822bc5739e5d71d24bb258706d244b37f6b598710fead6fb675ce3b8143c3b838adfb262479527e248bd0f45b7cc6d1ff6cbe2e1b09fa8a80aced4fb0ab157046feb1dfe50918c54721e14c74c55f2e35d2ee56947b34f77efca8892487c6116fe5cb2e4f2f82e1bf7b477221475b488a801c555f676711d01b1f573b4e59cccb73a720848d0767ec60ecdedd15861612e460466df712647378b9381185206e36ed7526c4cf875efa873e0ac194a21b4d6e446f31bda29d90953ea729
You got it! Here's the flag: freepass{s1mpl3_3n0ugh_r1ght???m4yb3}

```

dan didapatkan flagnya!!

Flag: freepass{s1mpl3\_3n0ugh\_r1ght???m4yb3}


dont\_open it

Challenge

11 Solves

×

don\_t open it  
100

 chall

Flag

Submit

Ketika dilakukan exiftool:

```
(steven@steven)-[~/Downloads/Poros]
$ exiftool chall
ExifTool Version Number      : 12.67
File Name                    : chall
Directory                    : .
File Size                    : 2.6 MB
File Modification Date/Time   : 2024:02:14 22:23:51+07:00
File Access Date/Time        : 2024:02:14 22:24:36+07:00
File Inode Change Date/Time   : 2024:02:14 22:24:27+07:00
File Permissions              : -rw-r--r--
Error                        : Unknown file type

(steven@steven)-[~/Downloads/Poros]
$
```

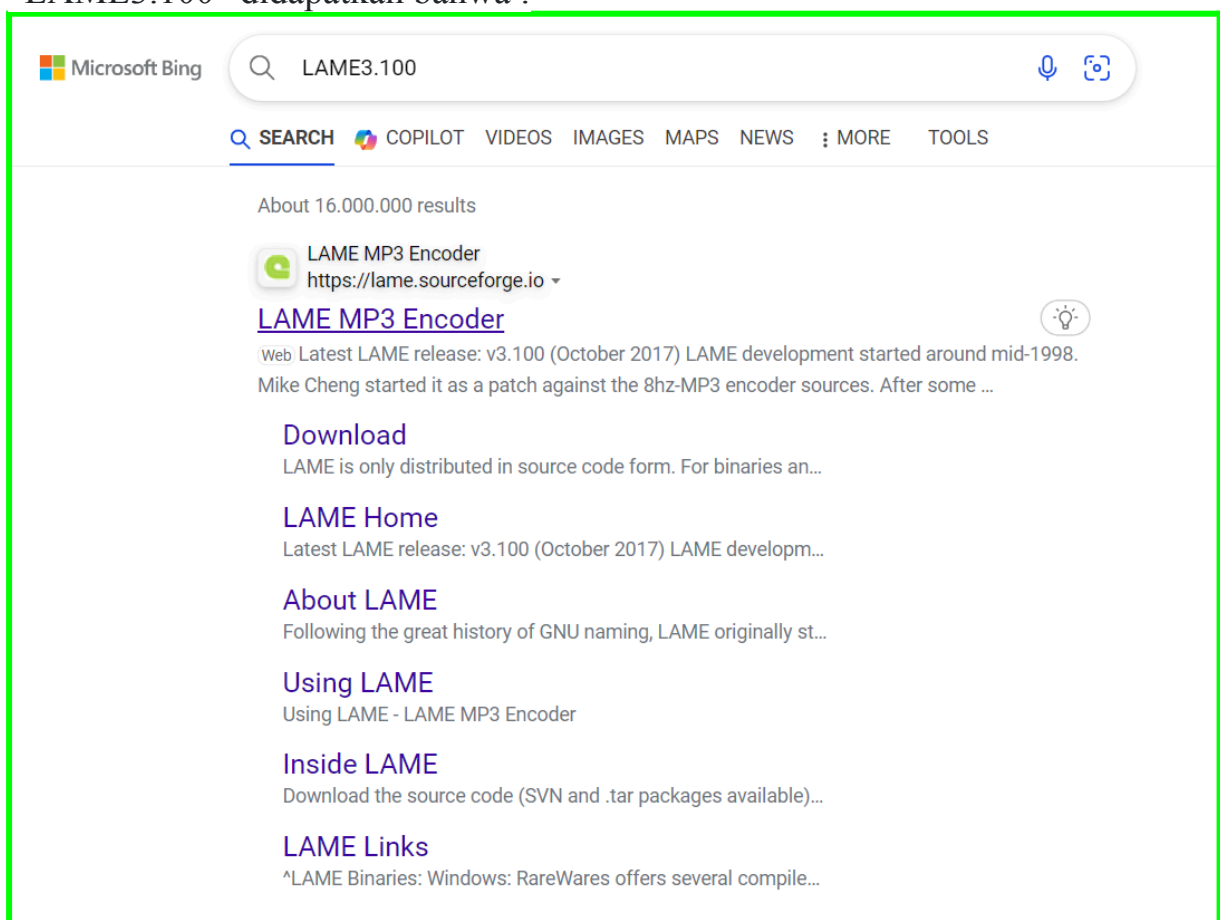
begitupun juga dengan *cat*

```
(steven@steven)-[~/Downloads/Poros]
$ cat chall
POROSXingH!
  4%xC6sh7r~e"y+yJ+9pp+-+6t+d-qbTM. -b+Q=====
  4+a`r~A++4P++++O++++:H"WFL          +I+++B+d"+i
  4+@y#?+-----jgh+wutVCV+Q(+,4$E+E
                                0(I+dI
                                tP++++Df+
)++@4++++e/
      ~^++++H$+
                        lZ
FT++L9]+K` `[]X*+    +4+. [s+k=D+?+=++++w+++dQEi+Y{+RQ"+
+4%KAty+++ \+++++uZYe++3[cqp++++Vn+r++V+]::}.Ph6D+}>_+
+}+3+4+yUSK+++I+'(8pa+X++5+7+++++L+[vVa#+++++xH,+l++++X+\MF++D   I~o
f++3},-V W+V++++vHcd++,cxM+`~LV0)K0dtB+T+++++yū+nt+n++/A+"++U+TH@++!+P!           H++++*j++PU+
                               J+7HX5(4Y+v+0XTid+z+qIk<+a,,+U++++H+mM+C+++++
```

tetapi pada saat saya melakukan eksekusi dengan command “strings” didapatkan sesuatu yang janggal yaitu:

```
jLAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100  
LAME3.100
```

muncul suatu kata yaitu “LAME3.100”, dan setelah saya mencari apa itu “LAME3.100” didapatkan bahwa :



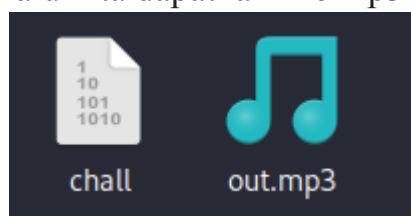
The screenshot shows a Microsoft Bing search results page for the query "LAME3.100". The search bar at the top contains the text "LAME3.100". Below the search bar, there are navigation links for "SEARCH", "COPILOT", "VIDEOS", "IMAGES", "MAPS", "NEWS", "MORE", and "TOOLS". The search results indicate "About 16.000.000 results". The first result is for "LAME MP3 Encoder" with the URL "https://lame.sourceforge.io". Below this, there is a section titled "LAME MP3 Encoder" with a lightbulb icon. The text below this section reads: "web Latest LAME release: v3.100 (October 2017) LAME development started around mid-1998. Mike Cheng started it as a patch against the 8hz-MP3 encoder sources. After some ...". Below this, there are several links: "Download", "LAME Home", "About LAME", "Using LAME", "Inside LAME", and "LAME Links". Each link has a brief description: "Download" (LAME is only distributed in source code form. For binaries an...), "LAME Home" (Latest LAME release: v3.100 (October 2017) LAME developm...), "About LAME" (Following the great history of GNU naming, LAME originally st...), "Using LAME" (Using LAME - LAME MP3 Encoder), "Inside LAME" (Download the source code (SVN and .tar packages available)...), and "LAME Links" (^LAME Binaries: Windows: RareWares offers several compile...).

ternyata LAME3.100 adalah suatu software yang meng-encode file mp3 menjadi audio coding format.

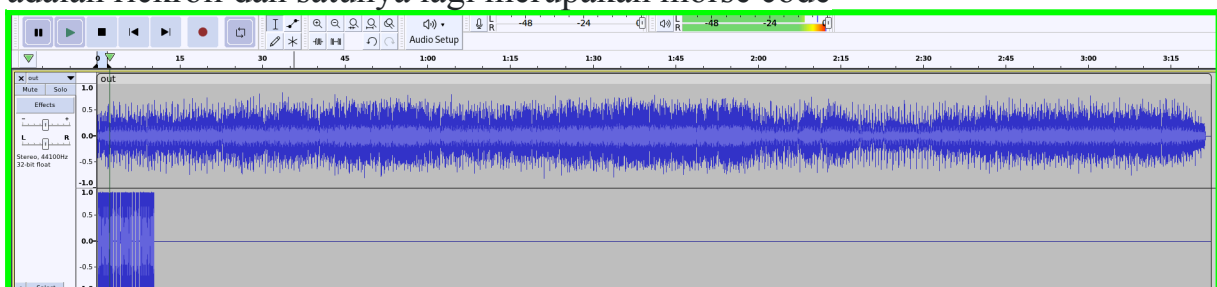
setelah mencari beberapa referensi, akhirnya saya mendapatkan tools untuk mengembalikan semula bentuk audio coding format menjadi mp3 dengan tools “ffmpeg”, lalu kita run saja

```
(steven@steven)-[~/Downloads/Poros]
$ ffmpeg -i chall out.mp3
ffmpeg version 6.1.1-1 Copyright (c) 2000-2023 the FFmpeg developers
  built with gcc 13 (Debian 13.2.0-9)
  configuration: --prefix=/usr --extra-version=1 --toolchain=hardened --libdir=/usr/lib/x86_64-linux-gnu --incdir=/u
om --enable-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libcodec2 --enable
le-libglslang --enable-libgme --enable-libgsm --enable-libjack --enable-libmp3lame --enable-libmysofa --enable-libop
and --enable-libshine --enable-lisnappy --enable-libsoxr --enable-lispeex --enable-lisrt --enable-libssh --enable
--enable-libxml2 --enable-libxvid --enable-libzimg --enable-libzmq --enable-libzvi --enable-lv2 --enable-omx --ena
-librsvg --enable-libvpl --disable-libmfx --enable-libdc1394 --enable-libdrm --enable-libiec61883 --enable-chromapri
libavutil      58. 29.100 / 58. 29.100
libavcodec     60. 31.102 / 60. 31.102
libavformat    60. 16.100 / 60. 16.100
libavdevice    60.  3.100 / 60.  3.100
libavfilter     9. 12.100 /  9. 12.100
libswscale     7.  5.100 /  7.  5.100
libswresample  4. 12.100 /  4. 12.100
libpostproc   57.  3.100 / 57.  3.100
[mp3 @ 0x56303cc2ab80] Skipping 417 bytes of junk at 0.
[mp3 @ 0x56303cc2ab80] Estimating duration from bitrate, this may be inaccurate
Input #0, mp3, from 'chall':
  Duration: 00:02:13.33, start: 0.000000, bitrate: 153 kb/s
  Stream #0:0: Audio: mp3, 44100 Hz, stereo, fltp, 153 kb/s
Stream mapping:
  Stream #0:0 → #0:0 (mp3 (mp3float) → mp3 (libmp3lame))
Press [q] to stop, [?] for help
Output #0, mp3, to 'out.mp3':
  Metadata:
    TSSE                : Lavf60.16.100
  Stream #0:0: Audio: mp3, 44100 Hz, stereo, fltp
  Metadata:
    encoder              : Lavc60.31.102 libmp3lame
[out#0/mp3 @ 0x56303cc2ce40] video:0kB audio:3164kB subtitle:0kB other streams:0kB global headers:0kB muxing overhea
size=      3165kB time=00:03:22.47 bitrate= 128.0kbits/s speed= 117x
```

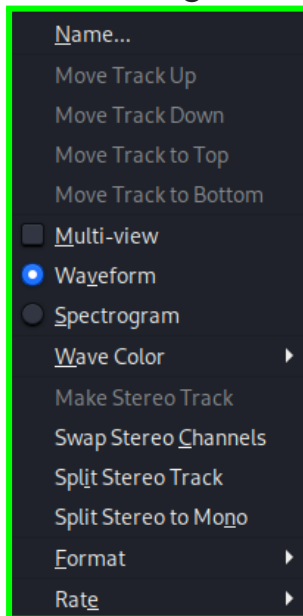
lalu kita dapatkan file mp3 tersebut



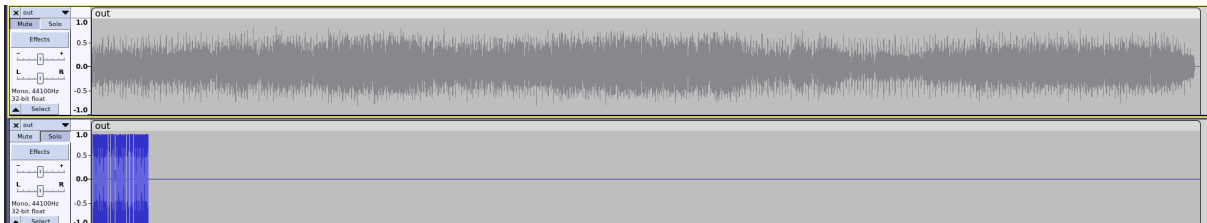
dan setelah dibuka file mp3 tersebut terdapat 2 suara yang mana salah satunya adalah rickroll dan satunya lagi merupakan morse code



setelah mencari berbagai cara, akhirnya saya dapat memisahkan kedua suara tersebut dengan cara klik “Split Stereo to Mono”



Lalu kita mute sound rickrollnya dan kita export agar bisa mendapatkan hasil dari morse code tersebut



Setelah di convert maka kita dapatkan flagnya!



Flag: freepass{BUK4NPR4MUKAYA4}

## WEB

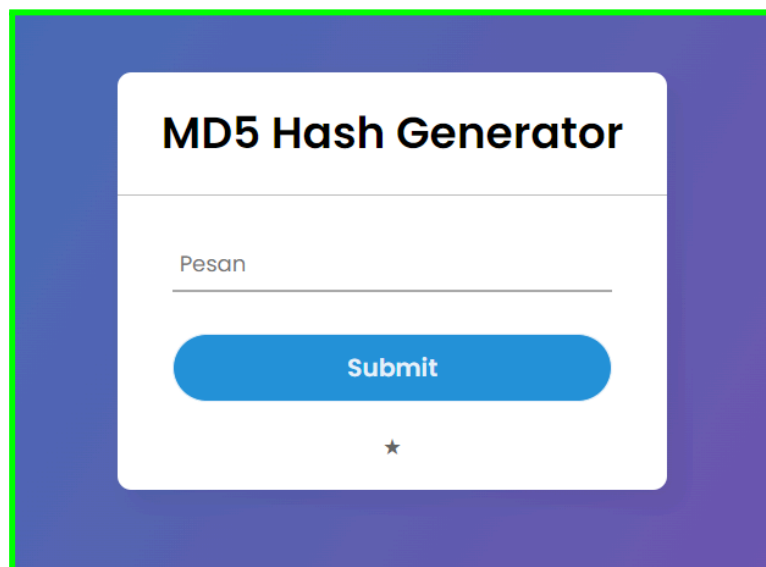
messi

Deskripsi Soal :

is messi the goat???

10.34.4.172:3029

Pada chall ini, diberikan suatu web dengan form input yang meng-generate input yang kita masukkan menjadi md5 hash

A screenshot of a web form titled "MD5 Hash Generator". The form has a white background with a blue border. It contains a text input field labeled "Pesan" with a light gray border. Below the input field is a blue rounded rectangular button with the word "Submit" in white. At the bottom center of the form is a small black star icon. The entire form is set against a dark blue background with a green border.

Form tersebut tampak normal, sehingga ketika saya mencoba meng-inspect elementnya terdapat komentar html yang berisi parameter “?source”. Parameter ini kemudian kita gunakan pada url, sehingga didapatkan source codenya:

```

<?php
include 'flag.php';
$secret = '0e972564156277235517192160089948';

if (isset($_GET['source'])) {
    show_source('index.php');
    return;
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>MD5 Hash Generator</title>
</head>

<body>
    <div class="card">
        <h1>MD5 Hash Generator</h1>
        <form>
            <div class="uinput">
                <input name="message" type="text" required placeholder="Pesan">
            </div>
            <input type="submit" value="Submit">
            <div class="result">
                <?php
                if (isset($_GET['message'])) {
                    if ($secret == md5($_GET['message'])) {
                        echo $flag;
                    } else {
                        echo 'Hasil MD5:<br>' . md5($_GET['message']);
                    }
                } else {
                    echo '★';
                }
                ?>
            </div>
        </form>
    </div>

    <!-- /?source -->
</body>

```

disini terdapat 'flag.php' yang mana ketika saya coba, saya terkena rickroll lagi...

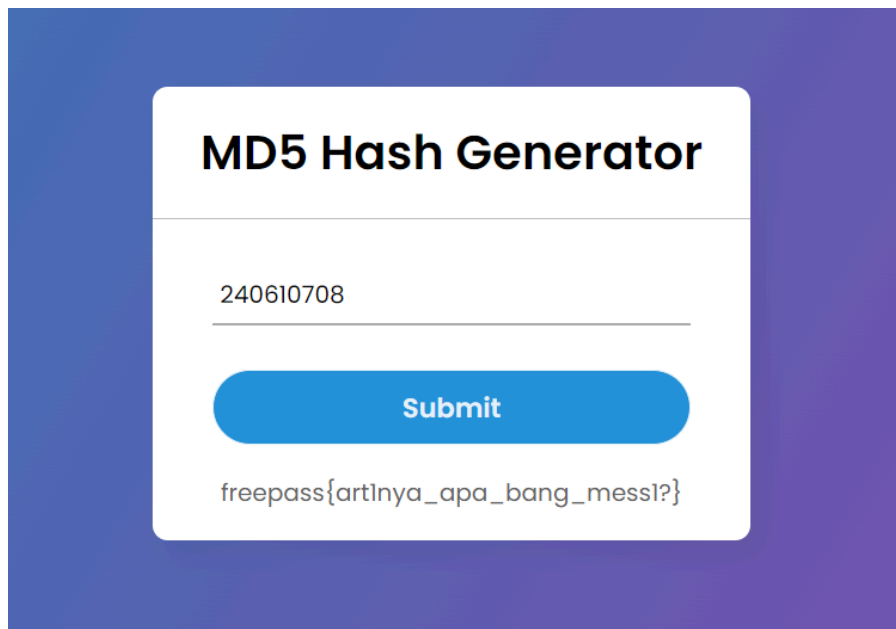


Setelah menganalisa kode tersebut, didapatkan bahwa kita perlu menginput suatu kata yang ketika di-hash kan akan memiliki hasil yang sama dengan “\$secret”

Setelah melakukan analisa lebih lanjut, kita tidak perlu mencari kode sama persis dengan “\$secret”, kita hanya perlu mencari kata yang ketika di MD5 hash akan menghasilkan “0e....”

Seperti kata ‘240610708’, yang ketika di hashed akan menghasilkan: ‘0e462097431906509019562988736854’.

Kemudian kita coba memasukkan kata ‘240610708’ kedalam form input tadi

A screenshot of a web application titled "MD5 Hash Generator". The interface is centered on a white rounded rectangle with a blue gradient background. At the top, the title "MD5 Hash Generator" is displayed in bold black text. Below the title is a text input field containing the value "240610708". Underneath the input field is a blue rounded button with the word "Submit" in white. At the bottom of the white area, the output hash is shown: "freepass{art1nya\_apa\_bang\_mess1?}".

dan didapatkan flagnya.

Flag: `freepass{art1nya_apa_bang_mess1?}`