Nama : Dzaky Fattan Rizqullah

NIM : 13520003

2. Return-Oriented Programming – Solution

Perintah dari soal ini adalah untuk mengerjakan soal ROP pada *website* pwn.college. Soal yang dikerjakan adalah `babyrop_level1.0` dan `babyrop_level2.0`. ROP sendiri memanfaatkan *buffer overflow* yang dapat terjadi pada *stack* untuk mengubah *instruction pointers* setelah suatu program selesai menjalankan salah satu fungsi yang dijalankan. Cara mengubah nilai *instruction pointers* ini bermacam-macam, salah satunya yang paling umum adalah mengeksploitasi fungsi untuk membaca masukkan kita, dalam hal ini masukan / *payload* berupa *string byte* yang panjang, terdiri dari *padding* dan *return address* fungsi yang ingin dituju. Dengan mengubah nilai tersebut, program dapat beralih ke fungsi yang tidak semestinya, bahkan menuju *address* yang merupakan bagian di luar program tersebut.

Salah satu teknik yang memanfaatkan ROP adalah ROP-*chaining*, yaitu mengubah *instruction pointers* sedemikian rupa agar dapat menjalankan beberapa potongan kode program (*assembly*) secara berantai untuk menjalankan perintah yang diinginkan.

a. babyrop_level1.0

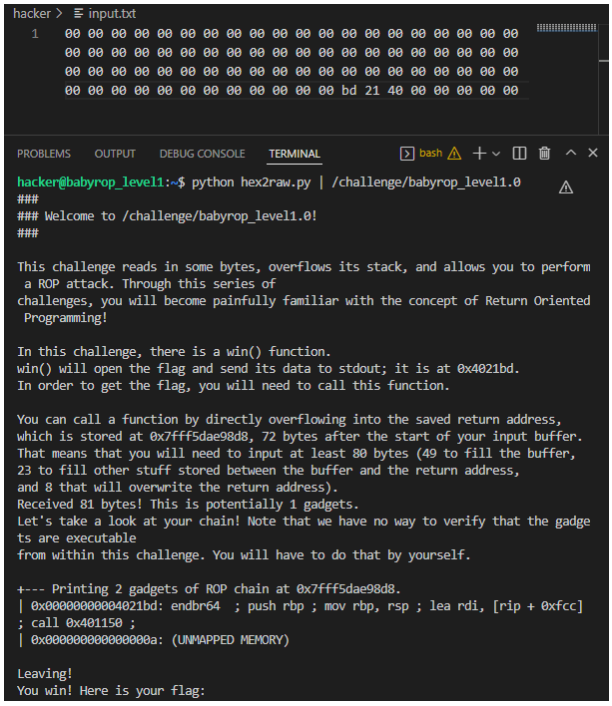
```
hacker@babyrop_level1:~$ cd /challenge
hacker@babyrop_level1:/challenge$ ./babyrop_level1.0
###
### Welcome to ./babyrop_level1.0!
###

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 0x4021bd.
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 0x7ffcc64a5118, 72 bytes after the start of your input buffer.
That means that you will need to input at least 80 bytes (49 to fill the buffer,
23 to fill other stuff stored between the buffer and the return address,
and 8 that will overwrite the return address).
Mayano Top Gun
Received 15 bytes! This is potentially -8 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 4294967289 gadgets of ROP chain at 0x7ffcc64a5118.

Leaving!
### Goodbye!
hacker@babyrop_level1:/challenge$
```

Gambar 1. `babyrop_level1.0`.

Pada soal ini, diberikan *prompt* seperti pada gambar di atas. Diminta untuk mengambil *flag* yang terletak pada fungsi `win()` pada *address* `0x4021bd`. Karena *stack* bersifat *randomized*, maka teknik untuk mendapatkan *flag* dapat menggunakan ROP. Dalam hal ini, diminta untuk memberikan masukan sebesar 80 *bytes* agar dapat meng-*overflow stack*. Karena kita tidak dapat langsung memasukkan *bytes input*, salah satu cara untuk mengakalinya dengan menggunakan teknik *pipe*, yaitu menggunakan stdout pada program lain sebagai stdin pada babyrop_level1.0. Program lain ini dapat berupa *utility* kecil yang menerima *input* .txt yang berisi *string* yang dapat diubah menjadi

bytes (~~misalnya dalam hal ini, menggunakan hex2raw.py yang diberikan pada praktikum orkom sebelumnya~~).

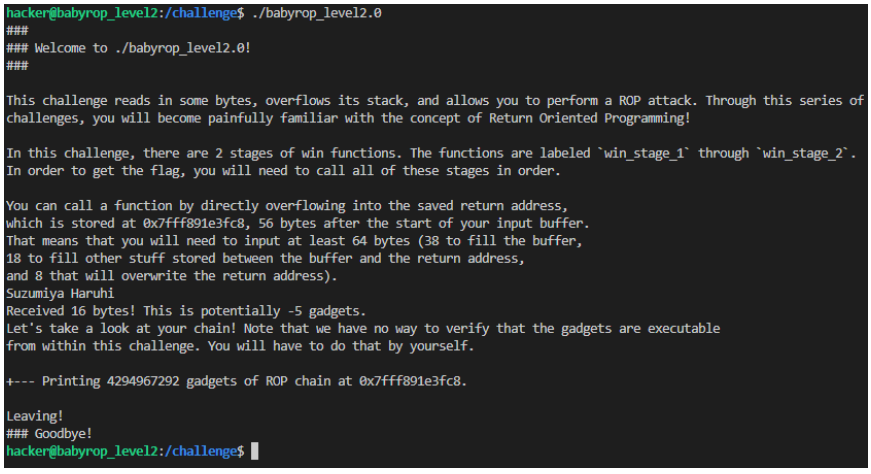
Mengikuti petunjuk pada *prompt*, yang dapat dilakukan adalah memasukkan 49 *bytes padding* untuk mengisi *buffer*, 23 *bytes padding* juga untuk mengisi bagian antara *buffer* dan *return address*, dan 8 *bytes return address* itu sendiri. Untunglah soal ini juga menunjukkan *return address* sebelum menuju ke fungsi yang ditunjuk oleh *return address* tersebut, sehingga dapat dicek apakah *buffer overflow* berhasil.



Gambar 2. `babyrop_level1.0` setelah dilakukan Teknik ROP.

Teknik ROP berhasil dan soal menampilkan flag yang benar.

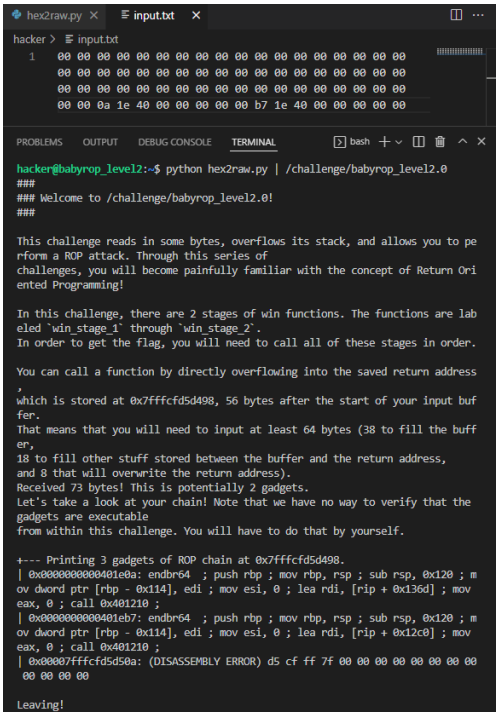
b. babyrop_level2.0



Gambar 3. `babyrop_level2.0`.

Soal ini serupa dengan soal sebelumnya, perbedaannya adalah fungsi yang dipanggil ada dua dengan urutan tertentu, yaitu `win_stage_1()` dan `win_stage_2()`. Teknik yang dilakukan akan sama dengan yang sebelumnya. Perbedaannya hanya terletak pada masukkan *bytes* untuk memanipulasi *return address*-nya.

Kembali mengikuti *prompt*, 38 ditambah 18 *bytes* sebagai *padding* dan 8 *bytes* berisi *address* menuju `win_stage_1()` terlebih dahulu. Memanfaatkan konsep *stack*, maka untuk memastikan *return address* dari fungsi tersebut menuju `win_stage_2()`, hanya perlu menambahkan masukan 8 bytes setelah *address* `win_stage_1()` agar *return address* fungsi tersebut akan menunjuk `win_stage_2()`. Karena *address* kedua fungsi tersebut tidak diberitahukan secara eksplisit, maka dapat diketahui dengan memanfaatkan *gdb*. Didapat fungsi `win_stage_1()` terletak pada `0x401e0a`, sementara `win_stage_2()` terletak pada `0x401eb7`. Keseluruhan *string* masukan akan menjadi seperti gambar di bawah.



Gambar 4. `babyrop_level2.0` setelah dilakukan Teknik ROP**.**

Teknik ROP juga berhasil dan soal menampilkan *flag* yang benar.