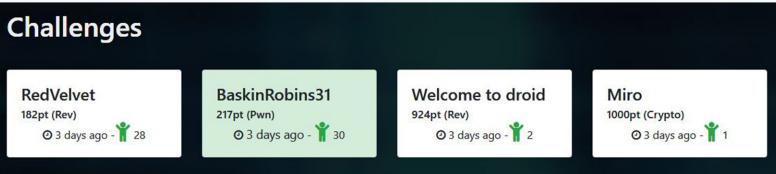


## [\*] Team Name: FetchDEX (32/39)



## [\*] WRITEUP



## BaskinRobins31: 30solves, 217p, PWN

- >file: ELF 64bit
- >as the file name says, the challenge was basically a popular koreean game caled Baskin Robins 31
- >in the main function there could be spotted a hint

```
loc_400B40: ; "Wow! You win!"
mov edi, offset aWowYouWin
call _puts
mov edi, offset aHintIsRop ; "Hint is : ROP"
call _puts
```

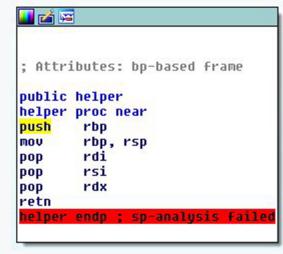
>after a short analyze of the code it can be spotted an interesting function: HELPER

function call

return address.

>next, looking for the vulnerability. in YOUR\_TURN function we can see a huge anount characters which are going to be read.

>this leads us to a stack overflow, which later will allow us to overwrite the



- >next the steps are pretty simple, leak libc and call system.
  - 1. Using the above vulnerability I first called write with the GOT entry of puts() in order to leak LIBC's base address.
  - 2. Next, I used a tool for detecting one\_gadgets in libc and called the one from offset 0x.
- >by running the following script:

https://github.com/Fineas/Me-CTF/blob/master/solve-BaskinRobins31.py

I managed to get the flag.

\*\*PRODECT-/CTE/COODGRATES\* Python SOLVE\_BLOST Py -- PRODUCE STATE OF THE PARTY OF

