- THOSE ARE VERY NOOB LEVEL AND CAN BE IGNORED .

Ctf problem solved , Using radare2 for debugging https://tryhackme.com/room/25daysofchristmas?ref=blog.tryhackme.com

Task 26 :

- As always start by analyzing the file , using `aaa` command (can also be done faster using `aa`) [analyze all]
- Then let's go to the main function using `pdf @main`

- To find the value of var_ch after the mov instruction using radare 2 , we set a breakpoint at the address of the mov instruction using `db 0x00400b51` .This command is used to set breakpoints.
- `dc` (This command is used to continue execution until the next breakpoint is hit) to run the code , `ds` (This command is used to step through the program instruction by instruction. It's short for "step") , and then print the hexdump for var_ch by using `px @ rbp-0xc`

```
[0x00400b51]> px @ rbp-0xc
- offset -      4445 4647 4849 4A4B 4C4D 4E4F 5051 5253  456789ABCDEF0123
0x7ffc4050e544  0100 0000 1890 6b00 0000 0000 4018 4000  ......k.....@.@.
0x7ffc4050e554  0000 0000 e910 4000 0000 0000 0000 0000  ......@.........
0x7ffc4050e564  0000 0000 0000 0000 0100 0000 78e6 5040  ............x.P@
0x7ffc4050e574  fc7f 0000 4d0b 4000 0000 0000 0000 0000  ....M.@.........
0x7ffc4050e584  0000 0000 0600 0000 8e00 0000 8000 0000  ................
0x7ffc4050e594  0c00 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffc4050e5a4  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffc4050e5b4  0000 0000 0000 0000 0000 0000 0004 4000  ..............@.
0x7ffc4050e5c4  0000 0000 6c1e a496 1a45 992b e018 4000  ....l....E.+..@.
0x7ffc4050e5d4  0000 0000 0000 0000 0000 0000 1890 6b00  ..............k.
0x7ffc4050e5e4  0000 0000 0000 0000 0000 0000 6c1e e46c  ............l..l
0x7ffc4050e5f4  3bc5 61d4 6c1e 1087 1a45 992b 0000 0000  ;.a.l....E.+....
0x7ffc4050e604  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffc4050e614  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffc4050e624  0000 0000 0000 0000 0000 0000 0000 0000  ................
0x7ffc4050e634  0000 0000 0000 0000 0000 0000 0000 0000  ................
```

PicoCTF (Two-Sum) : [I just bypassed the condition checking problem by patching binaries]

The problem gives us a C code , that has a function to return either -1 or 0 based on overflow condition. However instead of giving a large input to cause overflow I decided to patch the binary file to return -1 for any conditions .

```
sojib@debian-sjb:~/Academics/CSE406/CTFproblem/PicoCTF/two-sum$ ./demo
n1 > n1 + n2 OR n2 > n1 + n2
What two positive numbers can make this possible:
2
4
You entered 2 and 4
No overflow
```

- Opened the file in radare2 and analyzed using aaa command. Then I seeked to the main function (by using V to enter hex mode and then s main to seek to main)
- Found that it is actually calling a function named addIntOvf. Seek to the function addIntOvf.

```
|   0x000012b0      e8e0feffff     call sym addIntOvf(int, int, int) ; sym.addIntOvf
```

- That function was returning 0 based on a condition. Entered the write mode in the desired line using ctrl+A command. And added : mov eax,-1
- And the file was patched.

```
[VA:5]> mov eax,-1 (b8ffffffff)
[0x000013ba]> quit
sojib@debian-sjb:~/Academics/CSE406/CTFproblem/PicoCTF/two-sum$ ./demo
n1 > n1 + n2 OR n2 > n1 + n2
What two positive numbers can make this possible:
2
3
You entered 2 and 3
You have an integer overflow
```