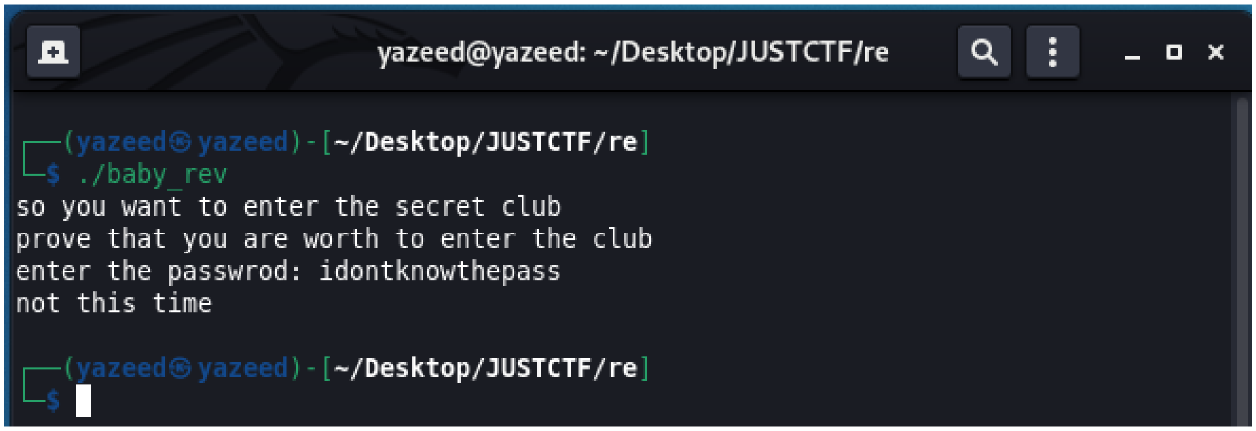
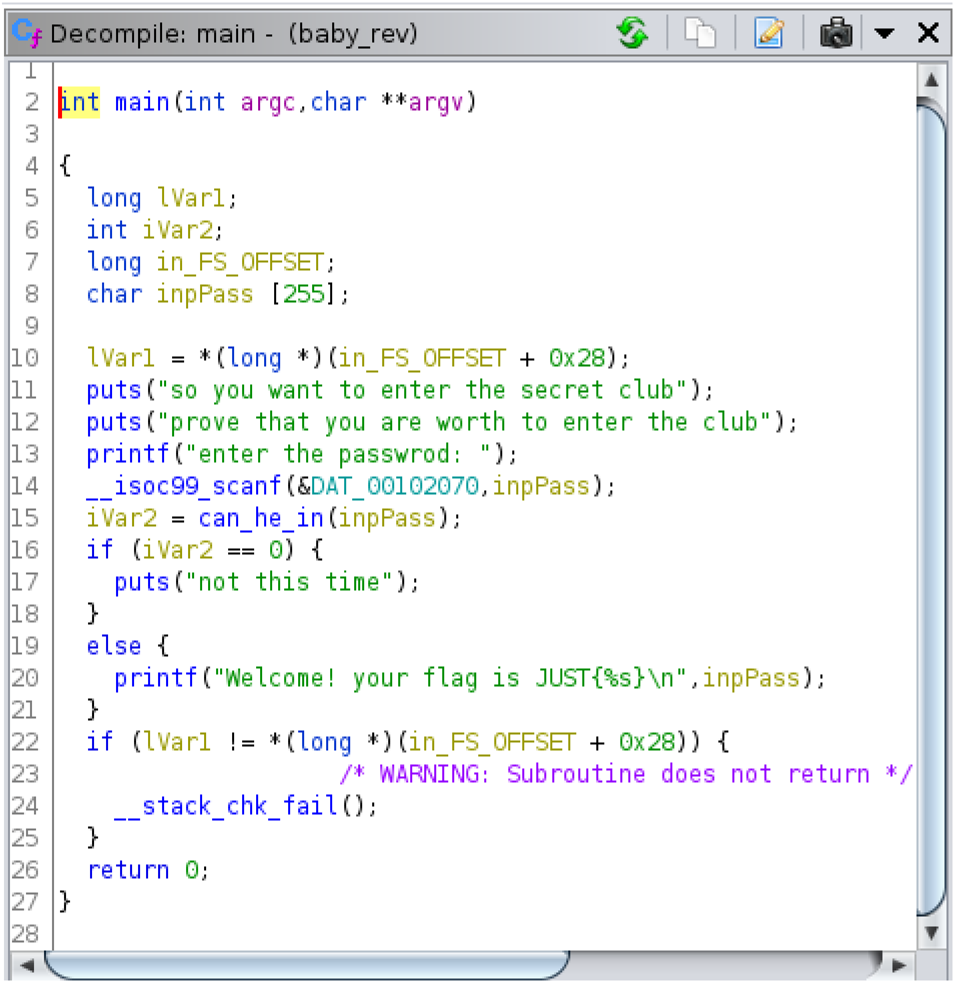
The second challenge was baby_rev, we go ahead and run the elf file and it asks us for a password we enter any value and it gives us the following:



```
┌──(yazeed㉿yazeed)-[~/Desktop/JUSTCTF/re]
└─$ ./baby_rev
so you want to enter the secret club
prove that you are worth to enter the club
enter the passwrod: idontknowthepass
not this time

┌──(yazeed㉿yazeed)-[~/Desktop/JUSTCTF/re]
└─$ 
```

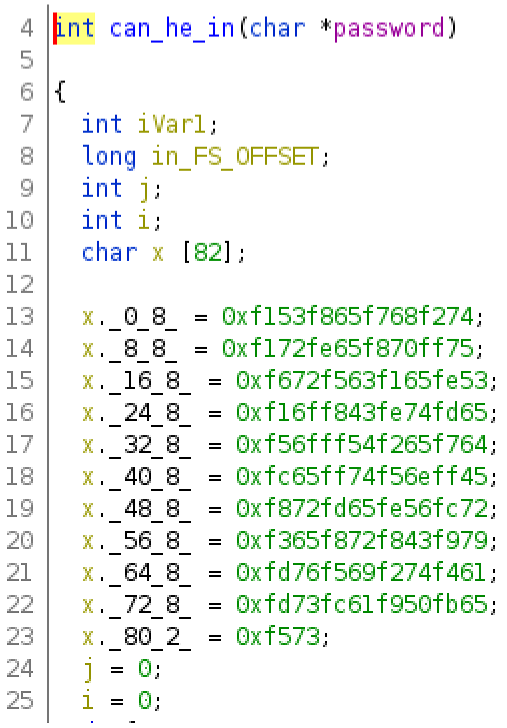
Let’s open the file with ghidra this time and look for the main function, here is the pseudo C code of main function:



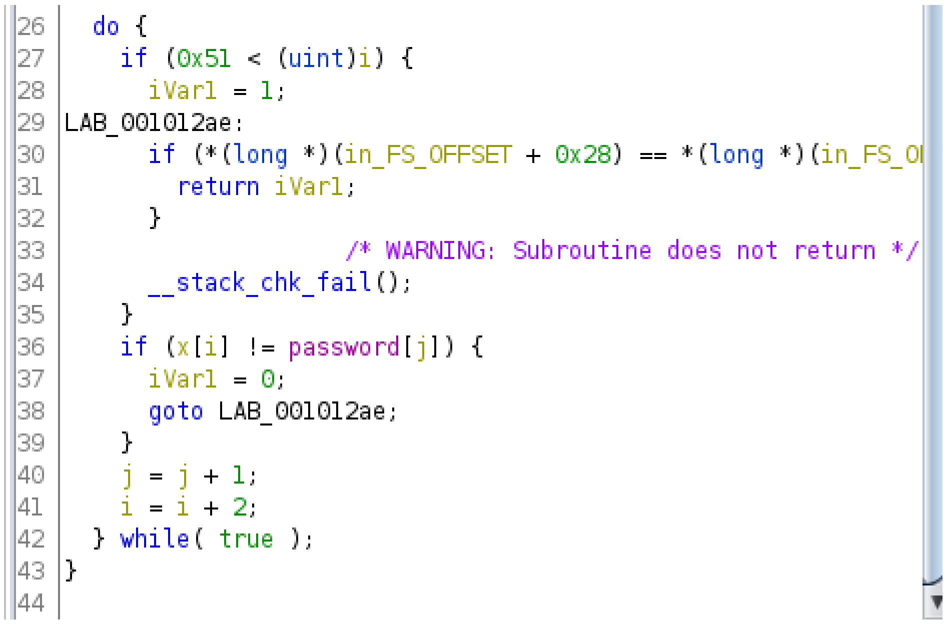
```
 1
 2 int main(int argc,char **argv)
 3
 4 {
 5   long lVar1;
 6   int iVar2;
 7   long in_FS_OFFSET;
 8   char inpPass [255];
 9
10   lVar1 = *(long *)(in_FS_OFFSET + 0x28);
11   puts("so you want to enter the secret club");
12   puts("prove that you are worth to enter the club");
13   printf("enter the passwrod: ");
14   __isoc99_scanf(&DAT_00102070,inpPass);
15   iVar2 = can_he_in(inpPass);
16   if (iVar2 == 0) {
17     puts("not this time");
18   }
19   else {
20     printf("Welcome! your flag is JUST{%s}\n",inpPass);
21   }
22   if (lVar1 != *(long *)(in_FS_OFFSET + 0x28)) {
23                     /* WARNING: Subroutine does not return */
24     __stack_chk_fail();
25   }
26   return 0;
27 }
28
```

We see that it asks us for input in line 14 and then sends our input as a parameter to the function “can_he_in” and the returning value is checked and depending on it will print us our flag in this case it’s our password.

Pseudo C code of the function can_he_in:



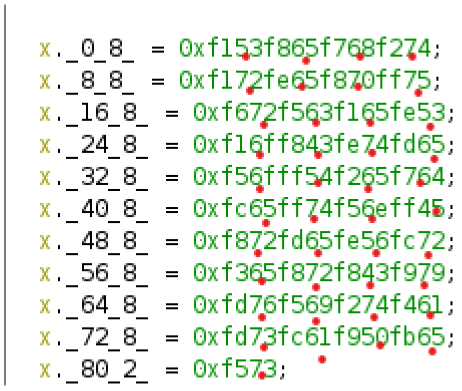
```c
int can_he_in(char *password)

{
  int iVar1;
  long in_FS_OFFSET;
  int j;
  int i;
  char x [82];

  x._0_8_ = 0xf153f865f768f274;
  x._8_8_ = 0xf172fe65f870ff75;
  x._16_8_ = 0xf672f563f165fe53;
  x._24_8_ = 0xf16ff843fe74fd65;
  x._32_8_ = 0xf56fff54f265f764;
  x._40_8_ = 0xfc65ff74f56eff45;
  x._48_8_ = 0xf872fd65fe56fc72;
  x._56_8_ = 0xf365f872f843f979;
  x._64_8_ = 0xfd76f569f274f461;
  x._72_8_ = 0xfd73fc61f950fb65;
  x._80_2_ = 0xf573;
  j = 0;
  i = 0;
```



```c
  do {
    if (0x51 < (uint)i) {
      iVar1 = 1;
LAB_001012ae:
      if (*(long *)(in_FS_OFFSET + 0x28) == *(long *)(in_FS_O
        return iVar1;
      }
                    /* WARNING: Subroutine does not return */
      __stack_chk_fail();
    }
    if (x[i] != password[j]) {
      iVar1 = 0;
      goto LAB_001012ae;
    }
    j = j + 1;
    i = i + 2;
  } while( true );
}
```

Basically what this function does is that it compares every character of our input with a byte from the array of characters(x) noting that it leaves a byte then it compares the next to our input byte

So our password “flag” can be figured by leaving a byte and decoding a byte from x, also be careful for endianness



```
x._0_8_  = 0xf153f865f768f274;
x._8_8_  = 0xf172fe65f870ff75;
x._16_8_  = 0xf672f563f165fe53;
x._24_8_  = 0xf16ff843fe74fd65;
x._32_8_  = 0xf56fff54f265f764;
x._40_8_  = 0xfc65ff74f56eff45;
x._48_8_  = 0xf872fd65fe56fc72;
x._56_8_  = 0xf365f872f843f979;
x._64_8_  = 0xfd76f569f274f461;
x._72_8_  = 0xfd73fc61f950fb65;
x._80_2_  = 0xf573;
```