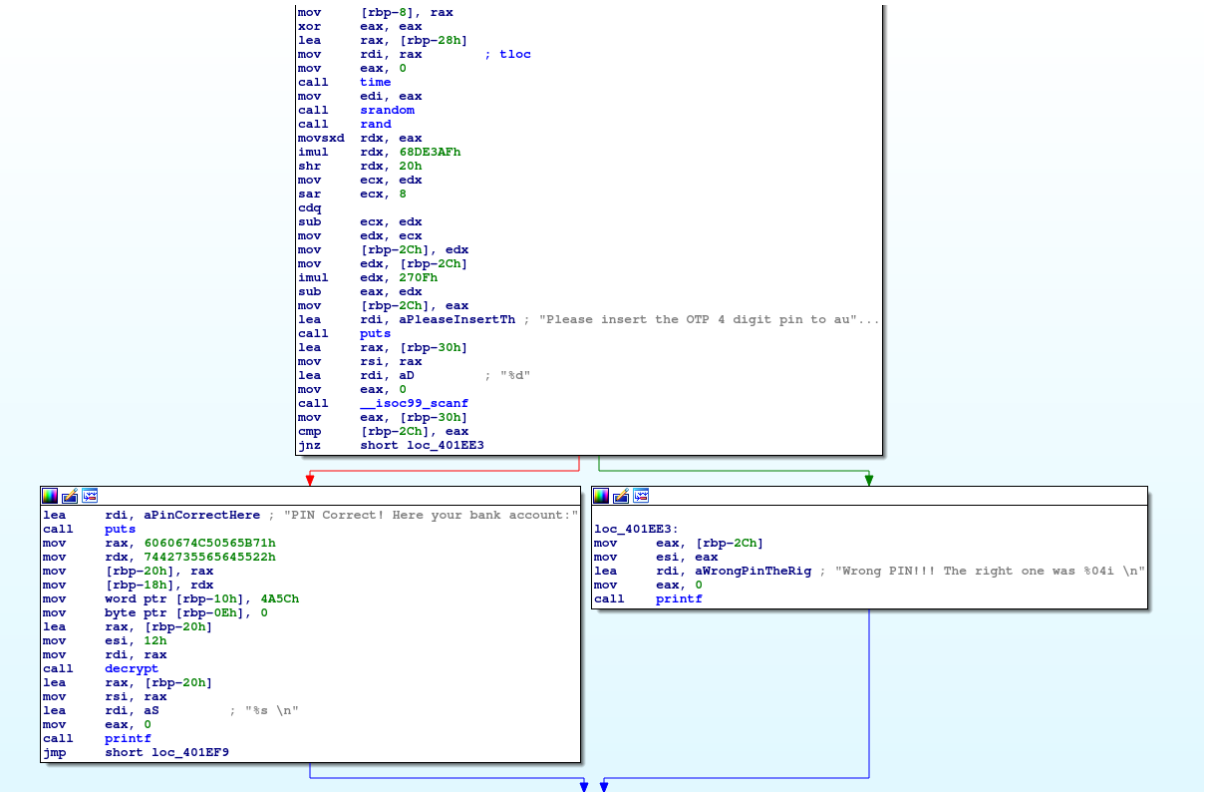
BANK ACCOUNT

We are asked to insert a pin to enter in our bank account. We can see that everytime the code changes, so we cannot guess.

Let's then open the binary with IDA.



We can see the workflow is very simple. The program uses random and srand functions to generate the code, then it asks to insert the 4 digit pin, uses the scanf to get the user input, and checks if they match (cmp [rbp-2Ch], eax). If they don't match it says the ping it wrongs, otherwise it prints that the PIN is correct, and then we see there is a decrypt function and a printf, so probably it's going to reveal the flag.

The patch here is very basic, what we need to do is to change the jump condition: If the PINS are equal we go to the wrong branch, otherwise we go to the correct branch. Instead of the jnz (jump if not zero, which means if they are different) we want jz (jump if zero, i.e., if they are equal).

Opening the hex view we see the instruction is 75 CD, where 75 is the opcode for JNZ. At this link we can see other JUMP instructions. http://www.unixwiz.net/techtips/x86-jumps.html

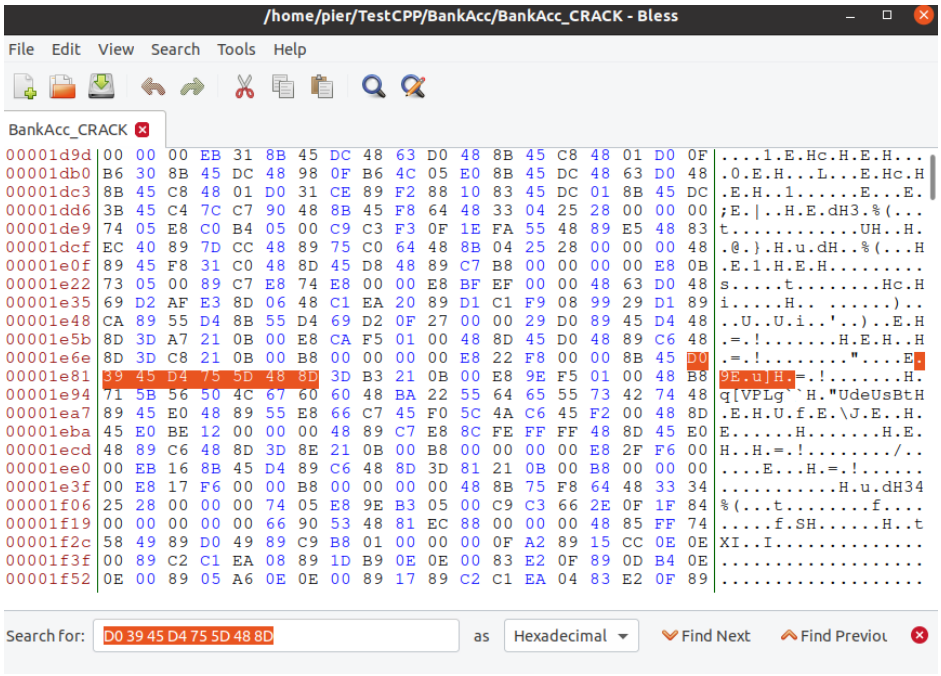
JZ is 74, so replacing 75 with 74, we would accomplish the goal.

Alternatively, looking at the addresses, we can see that the correct PIN Codeblock is right below the jnz instruction. This means that if we just NOP the jnz instruction, we would always go into the right PIN scenario.

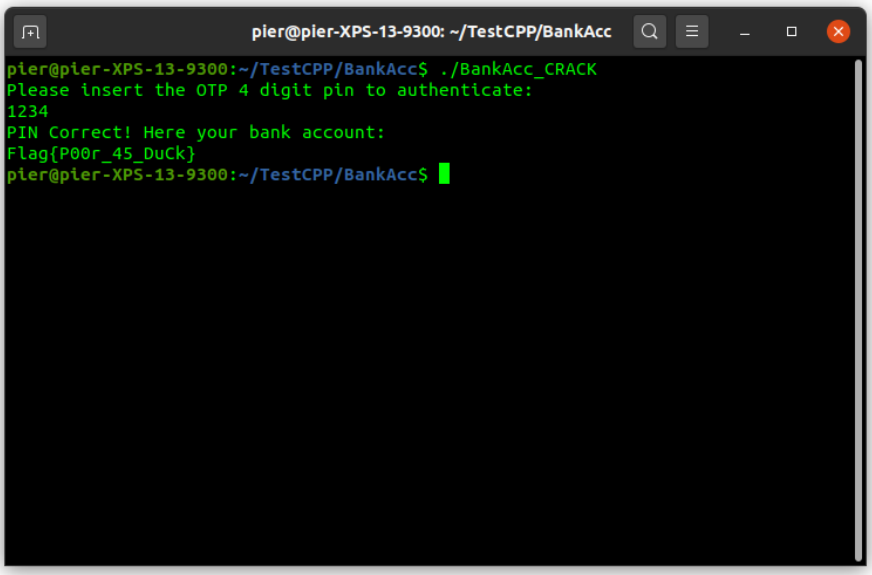
We patch the binary using an hex editor. First let's copy some bytes around the instruction to patch (0x75 0xCD)

```
00 E8 CA F5 01 00 48 8D  45 D0 48 89 C6 48 8D 3D
C8 21 0B 00 B8 00 00 00  00 E8 22 F8 00 00 8B 45
D0 39 45 D4 75 5D 48 8D  3D B3 21 0B 00 E8 9E F5
01 00 48 B8 71 5B 56 50  4C 67 60 60 48 BA 22 55
64 65 55 73 42 74 48 89  45 E0 48 89 55 E8 66 C7
```

For example let's copy D0 39 45 D4 75 5D 48 8D and search it into our hex editor



Change 75 with 74 as we said before, or NOP with 90 90 instead of 75 5D and here we go:



The flag is **Flag{P00r_45_DuCk}**