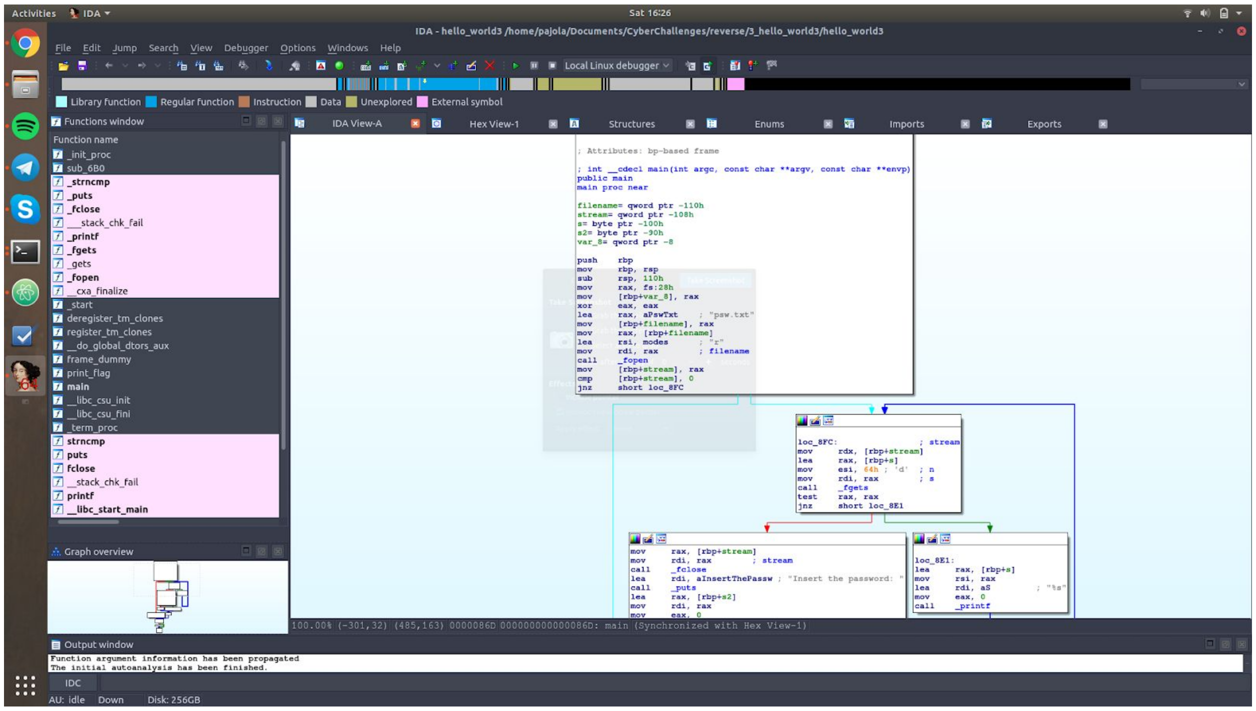
Reverse 3: Hello World 3

In this third exercise we need to find the password. This time the password is stored in a secure way, so let's assume that we cannot retrieve it easily as we did in the previous two exercises.

The purpose of this exercise is to practice with a useful and popular reversing tool called IDA: you can install it from this link.

Open *hello_world3* with IDA, and you should see something like this:



```
; Attributes: bp-based frame

; int __cdecl main(int argc, const char **argv, const char **envp)
public main
main proc near

filename= qword ptr -110h
stream= qword ptr -108h
s= byte ptr -100h
s2= byte ptr -90h
var_8= qword ptr -8

push    rbp
mov     rbp, rsp
sub     rsp, 110h
mov     rax, fs:28h
mov     [rbp+var_8], rax
xor     eax, eax
lea     rax, aPswTxt    ; "psw.txt"
mov     [rbp+filename], rax
mov     rax, [rbp+filename]
lea     rsi, modes      ; "r"
mov     rdi, rax        ; filename
call    _fopen
mov     [rbp+stream], rax
cmp     [rbp+stream], 0
jnz     short loc_8FC
```

```
loc_8FC:                    ; stream
mov     rdx, [rbp+stream]
lea     rax, [rbp+s]
mov     esi, 64h ; 'd'  ; n
mov     rdi, rax        ; s
call    _fgets
test    rax, rax
jnz     short loc_8E1
```

```
mov     rax, [rbp+stream]
mov     rdi, rax        ; stream
call    _fclose
lea     rdi, aInsertThePassw ; "Insert the password: "
call    _puts
lea     rax, [rbp+s2]
mov     rdi, rax
mov     eax, 0
```

```
loc_8E1:
lea     rax, [rbp+s]
mov     rsi, rax
lea     rdi, aS         ; "%s"
mov     eax, 0
call    _printf
```

```
100.00% (-301,32) (485,163) 0000086D 000000000000086D: main (Synchronized with Hex View-1)
```

```
Function argument information has been propagated
The initial autoanalysis has been finished.
```

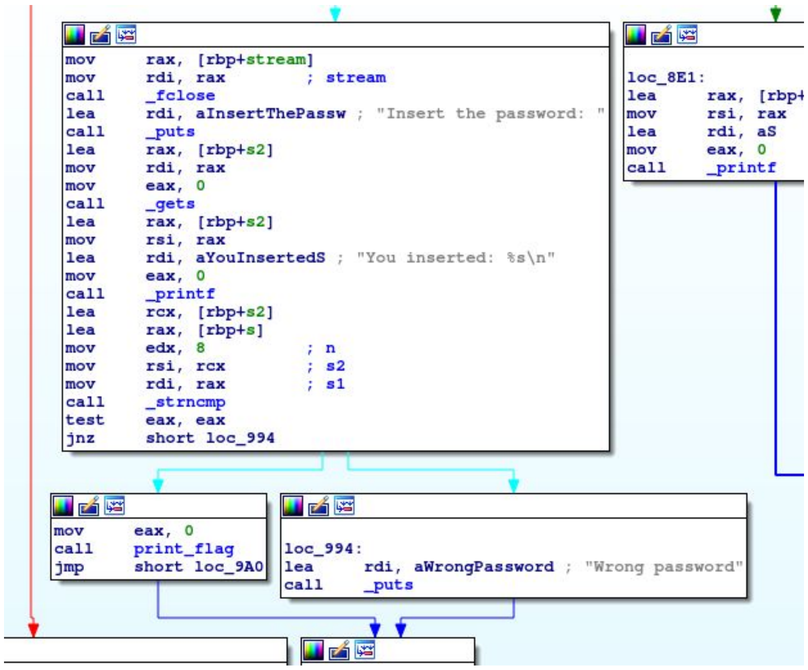
IDA shows us the execution flow; this is not bad, but IDA can do a lot more. For example … we can modify the program.

Let us explain it better: we know that hello_world3 is doing a password checking, and that we insert the correct password, we will gain privileges. We can imagine that the program follows the following pseudocode:

```
Inserted_psw <- input()
If check(real_psw, inserted_psw){
        give_privileges() //true branch
}
else{
        exit();  //false branch
}
```

What we can do, for example, is to call the function *give_privileges* also in the *else* branch, right? We can try to do this thing in IDA.

We are interested in the following part of the code, where there is a test between two registers (as in the previous exercise), and if True (correct password), we call *print_flag*.



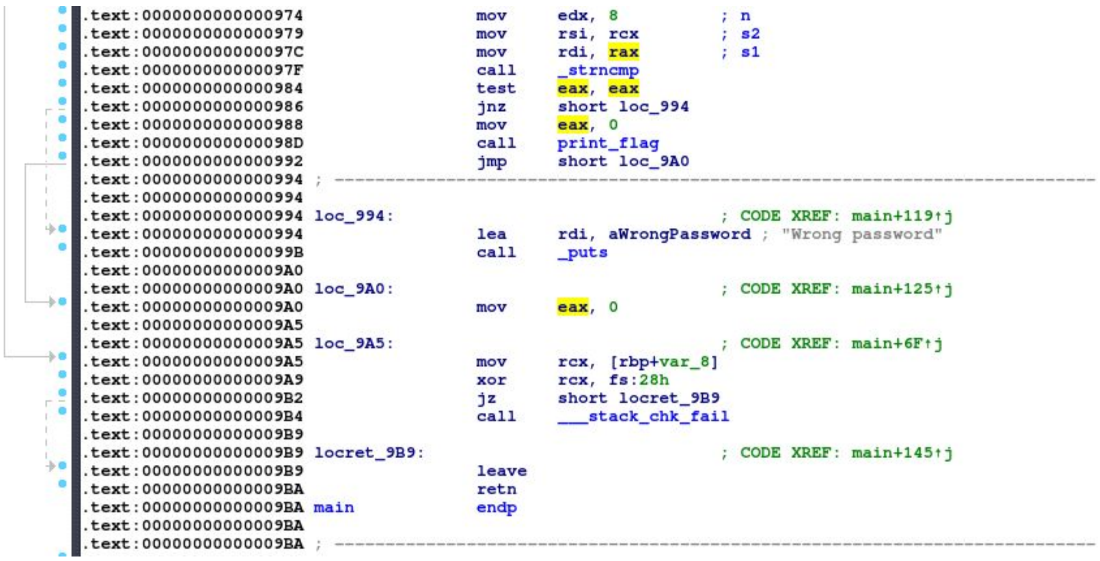
```
mov     rax, [rbp+stream]
mov     rdi, rax        ; stream
call    _fclose
lea     rdi, aInsertThePassw ; "Insert the password: "
call    _puts
lea     rax, [rbp+s2]
mov     rdi, rax
mov     eax, 0
call    _gets
lea     rax, [rbp+s2]
mov     rsi, rax
lea     rdi, aYouInsertedS ; "You inserted: %s\n"
mov     eax, 0
call    _printf
lea     rcx, [rbp+s2]
lea     rax, [rbp+s]
mov     edx, 8          ; n
mov     rsi, rcx        ; s2
mov     rdi, rax        ; s1
call    _strncmp
test    eax, eax
jnz     short loc_994
```

```
loc_8E1:
lea     rax, [rbp+
mov     rsi, rax
lea     rdi, aS
mov     eax, 0
call    _printf
```

```
mov     eax, 0
call    print_flag
jmp     short loc_9A0
```

```
loc_994:
lea     rdi, aWrongPassword ; "Wrong password"
call    _puts
```

From the main block, we can see that the instruction that jumps is a *jnz*, which is “jump if not equal”: if we change the value “loc_994” that is the label of the wrong password with the address of the True branch, the exercise is solved.
We first need to have the address of the True branch; we can change the view from “graph view” to “text view”, and you should have something like:



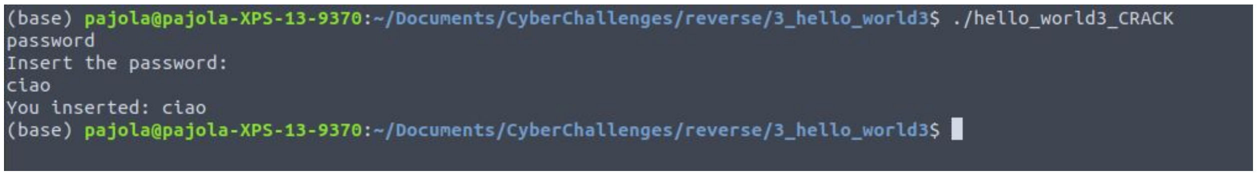
```
.text:0000000000000974                 mov     edx, 8          ; n
.text:0000000000000979                 mov     rsi, rcx        ; s2
.text:000000000000097C                 mov     rdi, rax        ; s1
.text:000000000000097F                 call    _strncmp
.text:0000000000000984                 test    eax, eax
.text:0000000000000986                 jnz     short loc_994
.text:0000000000000988                 mov     eax, 0
.text:000000000000098D                 call    print_flag
.text:0000000000000992                 jmp     short loc_9A0
.text:0000000000000994 ; ---------------------------------------------------------------------------
.text:0000000000000994
.text:0000000000000994 loc_994:                                ; CODE XREF: main+119↑j
.text:0000000000000994                 lea     rdi, aWrongPassword ; "Wrong password"
.text:000000000000099B                 call    _puts
.text:00000000000009A0
.text:00000000000009A0 loc_9A0:                                ; CODE XREF: main+125↑j
.text:00000000000009A0                 mov     eax, 0
.text:00000000000009A5
.text:00000000000009A5 loc_9A5:                                ; CODE XREF: main+6F↑j
.text:00000000000009A5                 mov     rcx, [rbp+var_8]
.text:00000000000009A9                 xor     rcx, fs:28h
.text:00000000000009B2                 jz      short locret_9B9
.text:00000000000009B4                 call    ___stack_chk_fail
.text:00000000000009B9
.text:00000000000009B9 locret_9B9:                             ; CODE XREF: main+145↑j
.text:00000000000009B9                 leave
.text:00000000000009BA                 retn
.text:00000000000009BA main            endp
.text:00000000000009BA
.text:00000000000009BA ; ---------------------------------------------------------------------------
```

Now we can try to modify the value of the *jnz* instruction (986) and substitute "loc_9A0" with "0x998". Our we can do also something smarter, like the substitute the *jnz* with its opposite *jz*; Do the following:

1. Click over 'jnz' : now it should be highlighted;
2. Go on: Edit > Patch Program > Assembly
3. Write "jz   short loc_994";
4. Press OK.

What we did is to manually modify the execution flow of our program. We just need to apply this patch to the original program … and voila'!
For patching the program do the following:

1. Edit > Patch Program > Apply patches to input file …
2. Confirm the operation



```
(base) pajola@pajola-XPS-13-9370:~/Documents/CyberChallenges/reverse/3_hello_world3$ ./hello_world3_CRACK
password
Insert the password:
ciao
You inserted: ciao
(base) pajola@pajola-XPS-13-9370:~/Documents/CyberChallenges/reverse/3_hello_world3$
```