# Pwn 6: Enc PWN 0

As usual, we need to answer to the following questions:

1. What is the goal of the exercise?
2. What is the entry point that allows us to reach our goal?

The **goal** is to call the function *print_flag*, defined on line 5:

```
void print_flag(){
        system("cat flag.txt");
}
```

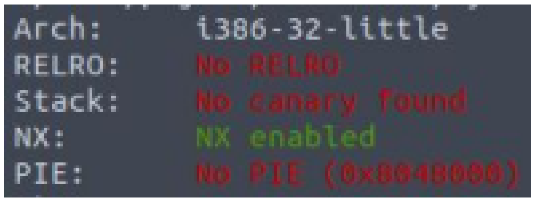
The **entry point** of the program is given on line 14, a *gets* function over the variable *buffer*. How can we reach our function? What we can do is to consider that the function is called during the execution of *main*, and, in more detail, during the *if-else* statement (lines from 15 to 20). The problem right now is that the function is called only when the *if* condition is TRUE, which happens when the variable *josh* is set to “H!gh”. However, this cannot happen during the normal execution flow.

What we can do is to observe that the variable *josh* should be located over the *buffer* on the stack frame and, since the *gets* function allows us to overwrite the stack, we can think of inserting “H!gh” during a buffer overflow attack.

Let’s draw the stack:

- Return address (4 bytes);
- Base pointer (4 bytes);
- Josh (4 bytes);
- Buffer (64 bytes).

N.B.: the program has been compiled for a 32-bit architecture (i.e., the registers are 4 bytes). We can see it by opening a terminal and typing “checksec pwn0”, which produces the following output:



```
Arch:     i386-32-little
RELRO:    No RELRO
Stack:    No canary found
NX:       NX enabled
PIE:      No PIE (0x8048000)
```

We have all the ingredients, and we can solve the problem easily with Python:

```
from pwn import *

garbage = 'a' * 64
msg = 'H!gh'
```

```
msgin = garbage + msg

p = process('./pwn0')
p.sendline(msgin)
msgout = p.recvall()

print('output:\t', msgout)
```

If you re-compile the program, it may happen that the solution described does not fit. This happens because the compilation mechanism is complex, and additional bytes can be inserted between the variables. If this is the case, you should try to debug a bit, for example by printing the value of *josh* after the *gets* to see its value at *run-time* and determine whether additional characters should be inserted between *buffer* and *josh*.