```
/*****************************************************************************
        COMPUTER SECURITY - LAB 1: Identification and Authentication

        Pedro Gonçalo Mendes - pedrogo@student.chalmers.se
        Mattias Olofsson - gusolomay@student.gu.se

In this solution, when a account is blocked all the system is blocked and has
to wait to try again the login

Pros: Easy to implement and doesn't take many resources to block the system
Cons: When a account is blocked, the system is all blocked (instead of block
only one account).

*****************************************************************************/

#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdio_ext.h>
#include <string.h>
#include <signal.h>
#include <pwd.h>
#include <sys/types.h>
#include <crypt.h>
#include "pwent.h"

#define TRUE 1
#define LENGTH 16

#define BLOCK_FOREVER -1
#define LAST_ATTEMPS -4


void sighandler() {
        //sighandler to ignore keyboard interruptions
        signal(SIGINT, SIG_IGN);                //ignore crtl+C
        signal(SIGTSTP, SIG_IGN);               //ignore crtl+Z
}


int main(int argc, char *argv[]) {

        mypwent *passwddata;

        char important[LENGTH] = "***IMPORTANT***";
        int veri;
        char user[LENGTH];
        char user_check[LENGTH];
        char prompt[] = "password: ";
        char *user_pass;
        char *crypt_pass;
        char salt[2];

        sighandler();

        while (TRUE) {
                /* check what important variable contains - part of buffer
overflow test*/
                printf("Value of variable 'important' before input of login name: %
s\n",
                                important);

                //empty strings
                strcpy(user, "");
                strcpy(user_check, "");
```

```
                printf("login: ");

                /*with the function fgets, we prevent buffer overflows of the
string user
                this functions only stores the last LENGTH-1 charecters read from
stdin
                (keyboard) in the string*/
                if (fgets(user, LENGTH, stdin) == NULL)
                        exit(0);
                sscanf(user, "%s", user_check);


                /*check to see if important variable is intact after input of
login name*/
                printf("Value of variable 'important' after input of login name: %
*.*s\n",
                                LENGTH - 1, LENGTH - 1, important);

                //verify if the user exist and in case afirmative returns the data
                passwddata = mygetpwnam(user_check);

                if (passwddata == NULL) {
                        //username doesn't exist
                        printf("Login Incorrect - Username does not exist \n");
                }else{
                        //passwddata != NULL -> username exists

                        if (passwddata->pwfailed == BLOCK_FOREVER){
                                //account is blocked forever
                                printf("Your account is blocked forever. \n");

                        }else{

                                if (passwddata->pwfailed == LAST_ATTEMPS){
                                        printf("Your account probabily was been
compromised. You have 3 more attempts until it blocks\n" );
                                }

                                /*wait for the user writes the password
                                text is not “echoed” on the terminal*/
                                user_pass = getpass(prompt);

                                /*encrypt user_pass -  salt is a two-character
string chosen
                                from the set [a-zA-Z0-9./].  This string is used
to perturb the
                                algorithm in  one of 4096 different ways.*/

                                /*we use strncpy only to copy 2 charecters are
copied and prevent
                                buffer overflows*/
                                strncpy(salt, passwddata->passwd_salt, 2);
                                crypt_pass = crypt(user_pass, salt);


                                if (!strcmp(crypt_pass, passwddata->passwd)) {

                                        //print the failed attempts
                                        if (passwddata->pwfailed < BLOCK_FOREVER){
                                                //you fail the first 5 tries
                                                printf("Number of failed attempts=
%d\n",(passwddata->pwfailed+10));
                                        }else{
                                                printf("Number of failed attempts
= %d\n", passwddata->pwfailed);
                                        }
```

```
                                        passwddata->pwage = passwddata->pwage + 1;

                                        if((passwddata->pwage > 10)||(passwddata-
>pwfailed < BLOCK_FOREVER)){
                                                printf("ALERT - You have to change
the password\n");
                                                printf("Entry the new password:
\n");

                                                user_pass = getpass(prompt); //new
password
                                                strncpy(salt, passwddata-
>passwd_salt, 2);
                                                crypt_pass = crypt(user_pass,
salt);  //encryption
                                                passwddata->passwd = crypt_pass;
                                                passwddata->pwage = 1;
                                        }

                                        passwddata->pwfailed = 0;

                                        veri = mysetpwent(user_check, passwddata);
                                        if (veri == -1){
                                                printf("Error in Myserpwent");
                                                exit(0);
                                        }

                                        /*  check UID, see setuid(2) */
                                        if(setuid(passwddata->uid) != 0 ){
                                                perror("Error on UID: ");
                                        }else{
                                                //if the password is correct
                                                printf(" Welcome to your system! :)
\n");

                                                /*  start a shell, use execve(2) */
                                                char *newargv[] = {NULL};
                                                newargv[0] = "/bin/sh";
                                                newargv[1] = NULL;

                                                if( execve("/bin/sh", newargv,
NULL) == -1){
                                                        perror("Error on execve:
");
                                                }
                                        }
                                }else{
                                        //if the password is incorrect

                                        //increment the number of failed attemps
                                        passwddata->pwfailed = passwddata-
>pwfailed + 1;

                                        if (passwddata->pwfailed <= BLOCK_FOREVER){

                                                if(passwddata->pwfailed ==
BLOCK_FOREVER){
                                                        //account  is blocked
                                                        printf("Your account is
blocked forever. \n");
                                                        passwddata->pwfailed =
BLOCK_FOREVER;
                                                }else{
                                                        //you have 3 more chances
to write the correct password
```

```
- Last attemps \n");


passwddata);


Myserpwent");



(passwddata->pwfailed) > 0){

attemps)

again - you have %d more chances\n",

>pwfailed));


passwddata);


Myserpwent");



many times

blocked


password to many times\n");

Blocked\n");


demonstration we only blocked the

to don't have to wait a lot of

system should be blocked during more

minutes*/


LAST_ATTEMPS;


passwddata);


Myserpwent");


                                }
                        }
                }
        }
```

```
                printf("Password Incorrect

        }
        veri = mysetpwent(user_check,

        if (veri == -1){
                printf("Error in

                exit(0);
        }


} else if( (passwddata->pwfailed) < 5 &&

        //password incorrect (first 5

        printf("Password Incorrect - try

                        (5-passwddata-


        veri = mysetpwent(user_check,

        if (veri == -1){
                printf("Error in

                exit(0);
        }

}else if((passwddata->pwfailed) >= 5){
        /*you wrote the wrong password too

        now the account is temporarily

        the user has to wait*/
        printf("You enter the wrong

        printf("You have to wait: System


        /*system is blocked - for the

        system during 20 seconds (in order

        time) but in real systems, the

        time, for example, 5 or 10

        sleep(20);

        passwddata->pwfailed =


        veri = mysetpwent(user_check,

        if (veri == -1){
                printf("Error in

                exit(0);
        }
}
```

```
        return 0;
}
```