Hacker Terminal Simulation Game

Ahmed sharif
H00458377
B37VB
13/04/2025

Abstract

This report will go into detail about the implementation of the game, how it works, and an explanation of how the code is written and why.

Introduction

The game, Hacker Simulator, has the objective of breaking through some simple firewalls to get to what the system is "hiding" with the help of an anonymous person giving the player hints and some dialogue to make the game more immersive.

Development

The game was developed purely with the language C and compiled with MiniGW, a Windows compiler for OS 10/11. Some functions, like windows.h will only work on Windows and MiniGW, windows.h helped with the sleep() function, which delays the text giving the game that hacker feel to make the player more immersed in the world of hacking and make the game feel as close to what is generally known as hacking.

Game Design and Flow

The game starts with a boot-up screen; the player then presses any key and enters to start the game. The first part of the game is a password-cracking-themed part. "Anonymous" sends a message before with a hint towards the password. There are 4 options, and one will be chosen at random: "root," "hack," "code," or "crack." The player will have 5 attempts to guess the password; if guessed incorrectly 5 times, the game will automatically shut down and the player will have to restart. If guessed correctly, another message from "anonymous" will show, congratulating the player and showing several files. One will automatically open, the player will then be met with another blockade that needs to be cleared, guess the right number in 30 seconds to access the file. Once the right number is cleared, the player will then be met with 3 options: steal the data, destroy the system, or take control of the system. Once the player chooses an option, "anonymous" will send a message, and the game will end.

Code Structure

The code is C and follows a top-down approach. It starts with the header inclusions, followed by the function declarations, into the main game functions, and then the main function in the end. The crack_password() function is called into play when the main() function section starts. Followed by the welcome_screen() function, which shows a boot up effect using the type_effect() function. type_effect(const char* text, int delay_ms) displays a typing effect giving the delay in the flow of the letters into the screen. Generate_random_password() function uses the rand() function to randomly select a password from the 4 options given. Then crack_password(const char* password) function comes into play, it handles the main password guessing phase of the game by taking the guesses from the user, compares and provides feedback within the time limit. number_guessing_game() function is then triggered after passing the cracking phase, the function provides the guessing game within a 30 second time limit. And lastly the final twist to the game, where the player is met with the three options that end the game.

Conclusion

the hacking simulator game in C is quite simple but is engaging to play with how the game is designed and how it plays out. Provides the player with an opportunity to experience what the normal world presumes hackers are like and what their world feels like.