TD 2 : Secure Password Handling and Compilation Options

M1 CSSE / CyberUs – Secure Programming

Camille Monière (course manager, A. Pr.) Maria Mendez-Real (teacher, A. Pr.)

November 2024

Foreword

The goal of the exercises is to present you different functions in C, as well as how password handling works on Unix.

You may continue to work on the exercises outside *if needed*, but have to focus on them in class, not on the labs.

They are not graded, but they prepare you for graded labs, and to the final exam, as well as giving you proper insights on still relevant issues.

GCC Compilation Options

Remember, if you need information on a GCC compilation option, use the manual. Run `man gcc` and write `/--the-option` to find the dedicated section in the terminal.

You can also search for it on the Internet, but be cautious with Large Language Models (LLMs, e.g., ChatGPT). They cannot know what you actually mean, only what you ask them. On such topic as compiler options, you may quickly get erroneous answer...

Instructions

You will work on the sources of the programs from TD1. You will have to implement a safe way to handle a password. There is no single answer, so be curious and adventurous!

Exercise 1: Storing Passwords

> *For the whole exercise, be as exhaustive as possible in your analysis. E.g., indicate the manual sections you used, how well your solutions would integrate in existing workflows depending on the use case (web server, distributed authentication system, local identities...).*

Q1. Context

Both program were vulnerable. What is the conceptual vulnerability in both ?

Q2. The Right Way

What is the safest place to store a password then ? In what form ? (several answers possible)

Q3. Implementation for a Unique Password

Write a program that uses your method and just verify a unique password.

Q4. Implementation for Several Passwords

First, search for `/etc/passwd` and `/etc/shadow` in the manual and on the Internet.

What do you have to enforce, concerning your password-related files ?

Exercise 2: Compiling Options

For this exercise, you may use Ghidra, GDB, `objdump` and `xdd`.

> *For the whole exercise, be as exhaustive as possible in your analysis. E.g., detail code snippet effect and objective, detail impact on execution time, and so on.*

Q1. Set Up

Copy your code from the previous exercise. Initialize a git repository, and use a Makefile.

Q2. Stack Protector

Compile first with `-fno-stack-protector` and then with `-fstack-protector`. Observe the decompiled code, the hex code, and so on. Explain the differences.

Q3. Exec Stack

What does the `-z execstack` flag do ? From what does this protect the program ? Can it be always enabled ?

Q4. -D_FORTIFY_SOURCE=X

In the same spirit, try using this flag for X set to 0, 1, and 2. Observe the generated object. What happens ?

Q5. Optimization

Globally using the compiler option Check the resulting binary code with `-O0`, and compare it to the binary code using `-O2`.

Locally using `#pragma` Look for the `#pragma GCC optimize("O0")` directive. What does it do ? How can it be used ?