CAA 2019

Lab #2

08-04-2019

1 Introduction

The goal of this lab is to implement a **password manager** that respects some security requirements.

- Please provide a **report** describing how you modelled your security. In particular explain how you secured your program to meet the required security properties shown below.
- Please provide also your **code** as well as a **makefile** or other compilation script.
- Provide a small **guideline** on how to use your program.
- You do not need to do a GUI for your program. A command-line version is sufficient.
- You do not need to use any complicated database system. Simple text files are enough for this lab.
- The C/C++ language is recommended but not mandatory. Keep in mind that you will have to erase data from memory which is a task that is easier to do in C/C++ than in Java.

2 Password Managers

Password managers are software used to manage the passwords of different websites/programs. They are unlocked using a **master password** which is the only password the user has to remember. Password managers can be in three different **states**:

1. **Not running**: the state in which the password manager is before being launched.
2. **Unlocked**: once the user launched the password manager and entered his master password, the password manager is in the unlocked state. To recover passwords, the user does not have to type his master password anymore.
3. **Locked**: The user can voluntarily locks the password manager which is then in the locked state. It can be unlocked by entering the master password.

In each state, we have different security requirements.

3 Not Running State

In the not running state, the password database is stored on the **disk**.

- One should **not** be able to recover any password (including the master password) without knowing the master password in this state.
- **Bruteforcing** the master password should be difficult for a user of the database as long as it's not trivial (123456 is trivial. HouseWithHorse is not).

4 Unlocked State

In this state, the user can freely ask for passwords corresponding to websites.

- It should **not** be possible to extract the **master password** from the memory.
- **Unaccessed passwords** should not be in clear in the memory.

5 Locked State

The password manager in the locked state should have the **same security** as the password manager in the **"not running" state**. In particular, an adversary reading the memory should not obtain any material allowing him to recover any of the passwords.

6 Your Implementation

Your implementation should include the following functionalities:

- A way to **lock** the password manager. A simple LOCK command in the command line is fine.
- A way to **recover** the password of a website. Displaying the password in the terminal is fine. Putting it directly in the clipboard is a plus.
- A way to **add** a new password in the database.
- A way to **change** the master password.