COMP.SEC.300

Joni Koskinen

Secure Programming exercise work report

General description

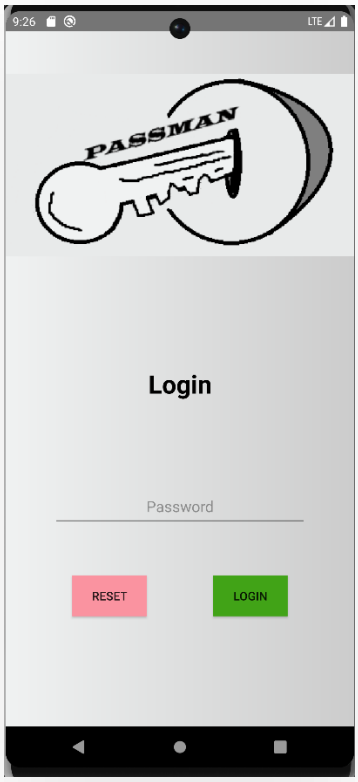
The program is a secure password manager app for Android devices. It works offline only on the device. It has a login function with a master password and support for biometric login via fingerprint authentication. The app currently lists added passwords in a single list with the passwords name showing but the actual password is hidden away. You can add a password and delete them. You can reset the master password, or you can just reset the whole program and wipe away all the data if you want. Every added password is encrypted when they are added and decrypted when they are fetched from memory.

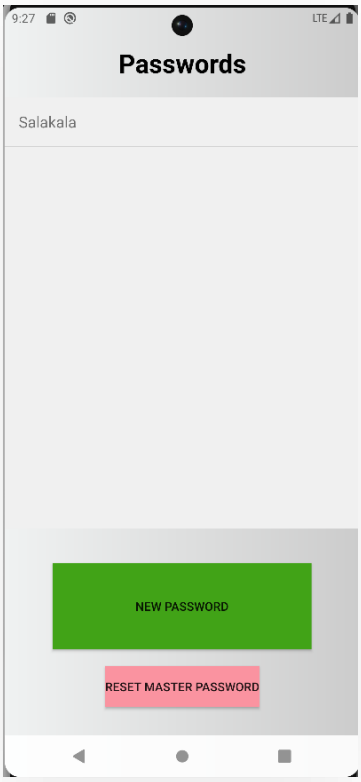
I used Kotlin as the programming language as it is common to use it with Android programs. My IDE was Android Studio because it is recommended to use when building Android apps. It also provides an Android emulator for quick testing. Gradle is used in the project to manage dependencies, to configure build process in build.gradle file, compile the app and package it into an APK file. Android Studio handled most of the Gradle work. The libraries that the program uses are Android Jetpack libraries, javax libraries, jbcrypt and gson. First two are for general usage, jbcrypt is for the hashing and gson for json file handling.

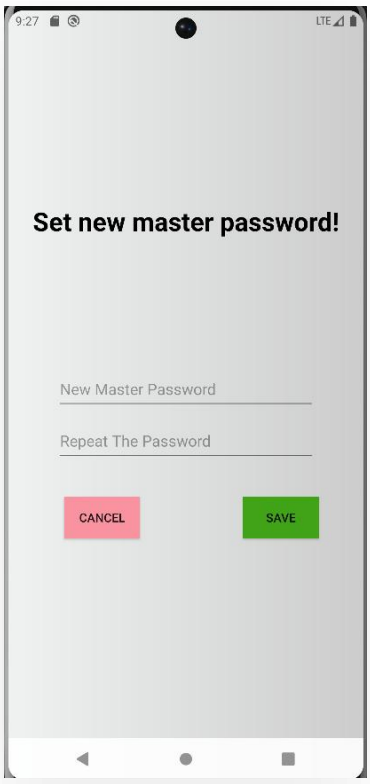
To run the program, it can be built from the source files using Android Studio or you can use the APK-file provided alongside the source files. Either way the APK needs to be transferred onto the device and then installed there. Android Studio emulator can also be used to test the program.

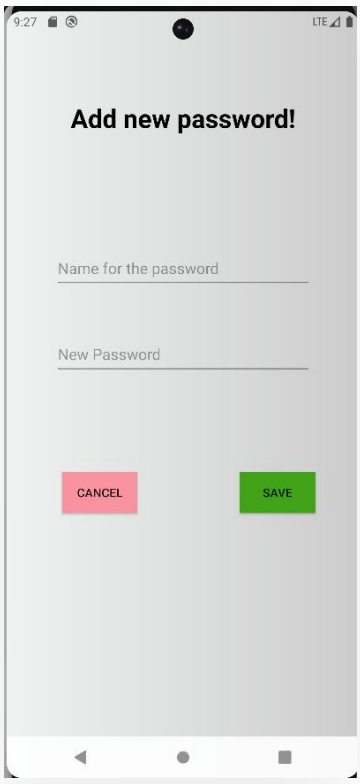
AI was used throughout the development process to help with the coding of an Android app using Kotlin. Most of the use cases were stuff like “How do I use this function” or “How to save data to Android internal memory?”. So, while some more difficult snippets can be from AI, I avoided copying whole functions.

On the next page there are the different Activities of the program. First there is login activity then main activity with the password list and the two activities where you can either set a new master password or add a new password to the list. The flow of the program starts with the login activity where it moves to the main activity. From main activity the user can move to the rest of the activities, and they loop back to main.


9:26
LTE
PASSMAN
Login
Password
RESET
LOGIN


9:27
LTE
Passwords
Salakala
NEW PASSWORD
RESET MASTER PASSWORD


9:27
LTE
Set new master password!
New Master Password
Repeat The Password
CANCEL
SAVE


9:27
LTE
Add new password!
Name for the password
New Password
CANCEL
SAVE

Structure of the program

The Android program consists of four activities: Login Activity, Main Activity, Add Activity and Reset Master Activity. Login Activity handles the login parts of the program such as setting up a master password, handling different kinds of login attempts, resetting the app and generating a secure key for the encryption.
Main Activity handles the list of passwords, fetching and deleting them and also encryption and decryption.
Add Activity gets a new password as user input and returns it to Main Activity to be decrypted.
Reset Master Activity gets a new master password as user input, hashes it and saves it to the internal memory.

There is also an item.kt file to define what a password entity contains and an Adapter file to handle the list object that holds the password list.

The data is saved in two files in internal memory of the program. Data.json holds the password data encrypted. The password name is cleartext, but the password itself is secret. Pass.json holds the master password hash so that when logging in, the hashes are compared with each other.

Secure Programming Solutions

I implemented most of the solutions related to security by memory of what is important such as the hashing of the master password and encryption of the saved passwords. I also used afterwards the OWASP cheat sheet series as a reference.

Hashing of the master password is implemented with bcrypt using the built-in salt generator. This is one of the hashes that OWASP recommends so it should be good. Bcrypt is designed to be slow so that brute-force attacks would be weaker.

The encryption of the saved passwords is implemented with AES-256 GCM mode. The encryption key is saved in Android Keystore which is a safe container for cryptographic keys that can only be used from the app that uses the keys.

There is a very basic length requirement for the master password because according to OWASP cheat sheet series, that is all that needs to be required.

All the passwords that are written to the app be it the master password or the passwords that are added to the list, are all hidden from the user so that anyone looking at the screen can’t see what is being written there.

Manual security testing?

I ran static security scanner MobSF on my code and it returned with two security issues:

The Activities aren’t protected which means that they are accessible from other apps on the device.

Activities are vulnerable to StrandHogg 2.0 Task hijacking meaning that the activities can be hijacked by other apps. They can display fake interface on top of the real one and steal information.

Missing implementations:

Change encryption key at the same time when you change master password. This requires decrypting every item in the list and encrypting them with the new key.

Current security issues

The activities are insecure, and they need better configurations so that they can't be accessed by anyone. I believe both issues presented by the static test could be fixed by configuring the app.

What could be implemented next?

Next, I should add the encryption key change and fix the security issues related to Activities. I also could improve the visuals of the app so it would look like a proper app instead of barebones proof of concept.