POLITECNICO
MILANO 1863

Password Manager

Project for the courses Advanced Operating Systems and Embedded Systems

Authors:

Rosevinay Yelluri

Arlind Rufi

Jan Fischer

POLITECNICO
MILANO 1863

Contents

POLITECNICO
MILANO 1863

1 Specification

The goal of the project is to implement a password manager as firmware for the STM32F407 board while using miosix as operating system.

The program should provide a command line interface through the serial port using appropriate library functions (e.g. printf and scanf) and allow inputting and retrieving passwords that will need to be stored encrypted on the internal flash. Functions for retrieving passwords should offer a possibility to show all stored passwords with their respective user/website name and in addition a search for a website / user name should be possible.

The driver for the internal flash has been written in a previous project by other students, and is available on the following git repository: https://github.com/goofy91/FileSystem

2 User Manual

3 Documentation of the implementation

3.1 The user interface

3.2 Writing to the flash

The data by the Password Manager is saved on the flash. It must be considered that the executable code (namely the operating system) is stored there as well. So one must be careful not to interfere with the data of the operating system. The Password Manager uses two functions to write (storeData) and read (loadData) the data from the flash. Both functions don't use parameters but the attributes of the PasswordManager class. For accessing the flash the memory layout described in table 1 is used. Here the standard start address is used as an example but it is valid for any other address with the corresponding offsets.

Address	Content	Comment
0x080F8000	'P'	Identification string
0x080F8001	'W'	"
0x080F8002	'M'	"
0x080F8003	0x00	"
0x080F8004	numOfPass	Short, which means that it occupies two bytes in the memory
0x080F8006	encryptedData	From here the flash contains WPTuples and the checksum of the password manager, which means all the user's stored password data
0x080F8007	"	Note: all this data is encrypted
...	...	...
EndOfData(= EOD)	0xFF	EOD = 0x080F8006+numOfPass*64+16 Because sizeof(WPTuple) = 64 and sizeof(checksum) = 16
Afterwards	0xFF	Area reserved for new passwords
...	0xFF	"
0x080FFFFF	0xFF	"

Table 1: Flash memory layout

For identification a string with content "PWM" is written at the beginning. In this way we can be sure that there is valid data in the flash and that there is no coincidentally valid data which doesn't make sense.

The loadData function

The storeData function

Address calculation

3.3 Encryption

4 Meeting Protocols

1. Meeting on the 28. January 2016

Main topics:

- flash functionality
- implementation details for data structure
- use of available libraries

As we had never done hardware programming with a flash before, we needed a briefing on the basic functionality of the flash. Before we had tried to implement a test program to write on the flash, but didn't know about the necessity to erase it beforehand. Briefly to use flash memory two functions are necessary: write and erase. However the write function can only put values to 0 not to 1. Whereas the erase function does not clear to 0 but to 1. So after erasing every byte of the erased sector contains 0xFF (all ones). The flash driver available only offers an erase of a whole flash sector.
Further it is important to know that erasing the flash destroys it over time; so erase should only be used when necessary. For our project we can reduce the number of used erase calls by using a commit function (not erasing the whole sector for every single password but erasing once in the end of a user session).

With the short time available for implementation in mind we agreed on a simple implementation of the necessary data structure. Instead of using a dynamically growing structure a static structure can be used. The basic idea given by our adviser was to use a struct to store websites and passwords as well as other relevant data on the flash. A password could consist of 32 bytes. The data structure may for instance use 32 KiB.

Most C++ library functions are usable in miosix (the corresponding system calls are implemented). This allows a much faster implementation of various parts of the project: The read function (which is not implemented in the flash driver) can make use of the memcpy function. Encrypting does not have to be implemented from scratch but available functions may be used. Further printf and scanf are available for the user interface. Alternatively the string class with cin and cout could be used.

2. Meeting on the 12. February 2016

Main topics:

- Flash address
- Encryption

0x08000000

Linker script (prevent OS and password data conflict)

Numbers of erase (proposed optimisation not possible because of checksum)

PolarSSL

CBC

Checksum

masterPassword

5 Glossary

Standard start address – used to write/read from flash. It is the last address in the flash with 32 KiB of space behind. This way it is as far away from the operating system as possible. The address is 0x080F8000. It is defined in Passwordmanager.h as STANDARD_ADDRESS.

WPTuple – (or WebsitePasswordTuple) is a struct defined in PasswordManager.h. It is used to store a website and the corresponding password.