Politecnico di Torino

Cybersecurity for Embedded Systems
01UDNOV

Master's Degree in Computer Engineering

Browser Password Manager

Project Report

Candidates:
Matteo Battilana (281389)
Gabriele Salvatore La Greca (281589)
Giovanni Pollo (s290136)

Referents:
Prof. Paolo Prinetto
Dr. Matteo Fornero
Dr. Vahid Eftekhari

Contents

List of Figures

List of Tables

Abstract

It's widely known that password are becoming more, since every day people are using them to protect their data. There are several ways to store your password, such as writing them on a piece of paper or in a file (both in the smartphone or in the laptop). Another method is to remeber by hearth the password, but this is not feasible since there would be too many pieces of information to remember. In this context, password manager, such as Bitwarden, LastPass, or Keepass, are a good solution. However, if not self hosted, all these password managers store the passwords in a cloud, which means that they are not completely undes the user control.

These were the main reasons behind the creation of this project, whose goal is to help people to remember their passwords, and to make it easier for them to access their data. In addition to that, data are stored on a board, which means that they are secure by definition, since without the physical board they would not be able to access the data.

The password manager has been designed from the ground up to be secure, but it also has a user friendly chrome extension that simplifies the insertion of the passwords in various websites.

CHAPTER 1

Generic Chapter

This is a generic chapter of your thesis. Remember to put ANY chapter in a different source file (including introduction and all the others).

For the purpose of this guide, the main LaTeXconstructs and how to use them will be explained here. Other thematic chapters will follow, i.e., which will trace the chapters that should be present in your thesis. Delete this generic chapter once you have learned this contents.

You can write in italic *like this*, you can write in bold **like this**, or you can write using colors like this.

This is an *itemize*, where you can put a list of items, like this:

- item number 1
- item number 2

This is an *enumerate*, where you can put a list of items with numbers, like this:

1. item number 1
2. item number 2

You can cite references like this: [?] [?], by using the \cite directive. You have to copy within \cite brackets the label of the entry that you have in the BibTeX file (`.bib`). The `.bib` file of this thesis is `mybib.bib`. he command \addbibresource at the top of this main file indicates what BibTeX file you are referring to.

As an example, this is a BibTeX entry:

```
@inproceedings{urias2018cyber,
  title={Cyber Range Infrastructure Limitations and Needs of Tomorrow: A Position Paper},
  author={Urias, Vincent E and Stout, William MS and Van Leeuwen, Brian and Lin, Han},
  booktitle={2018 International Carnahan Conference on Security Technology (ICCST)},
  pages={1--5},
  year={2018},
  organization={IEEE}
}
```

For every online paper that you may read on online libraries, you can download its BibTeX entry. For example:

1. For IEEE Xplore, click on the paper name, then click on “Cite This”, “BibTeX”, and you can find the entry;

2. For Google Scholar, click on the “Cite” voice under the paper name, then click “BibTeX”, and you can find the entry.

Just copy and paste such an entry in the .bib file. If you find a paper on Scholar that is nevertheless published by IEEE, by convention you should take the entry from the IEEE website and not from Scholar. To do this, just click on the title of the paper. This will redirect you to the resource page on IEEE Xplore. Once here, follow instructions at point 1.

When you compile, a correct number will automatically be assigned to the citation in the text, and the complete entry will appear at the bottom of the document, in the “Bibliography” chapter.

If you need to cite a generic online resource, which does not necessarily correspond to a scientific paper, use the @misc entry in the `.bib` file. A @misc entry looks like this:

```
@misc{nist2018,
    author = "{NIST}",
    title = "Cyber Ranges",
    year = "2018",
    howpublished = "\url{https://www.nist.gov/system/files/documents/2018/02/13/cyber_ranges.pdf}",
    note = "[Online; Accessed 2019, 28 November]"
  }
```

You have to manually create this entry from scratch and manually type these fields. Remember not to forget any of these fields. You can choose the label with which to refer to the resource. The title of the website (which you can see at the top of the tab of your browser showing the page) can be used as the title of the resource.

In general, enter a citation of this type for sites only when there are data, phrases, or images that you intend to report. Instead, if you want to cite names of software or hardware devices, prefer the use of the \footnote, in which you will only have to specify the URL of the item.

Remember that citations, both in the text and in the image captions, usually go to the end of a sentence, before the fullstop, as in this case [?]. In case of long periods, they can also be placed before other detachment signs, such as commas or semicolons, or colons if they precede a list, itemized or enumerated. An exemption is allowed in the event that the name of research projects, described in some scientific resource, is being introduced, as in this case:

Cybertropolis [?] is described in a very good paper by Gary Deckard.

Remember to put citations very often to justify your claims, especially when you report data or results. Just consider them as a justification of what you, in an original way, are writing. Citations are not needed to have permission to copy and paste sentences from online resources, which should NEVER be done - always try to rephrase the concept with your words.

This is an image example. Images must ALWAYS be understandable: never introduce images that have text smaller than the text in your document. If you create the images yourself, try not to make them clash too much with the style of your document, and use the same font as this thesis. If they are not images of your own creation, you MUST reference them. In the caption of the image, you need to insert a citation to the resource from which you took the image, at the end of the caption sentence, before the fullstop. Each image you enter MUST be referenced in the text, using a formula similar to this:

Figure 1.1 describes the architecture of the system.

You can refer to the image using \ref followed by the image label, that you put in the \label entry of the figure. Remember to use the word Figure with a capital F.

Remember that the more your text is adorned with figures, the more understandable, appreciable and readable it becomes.

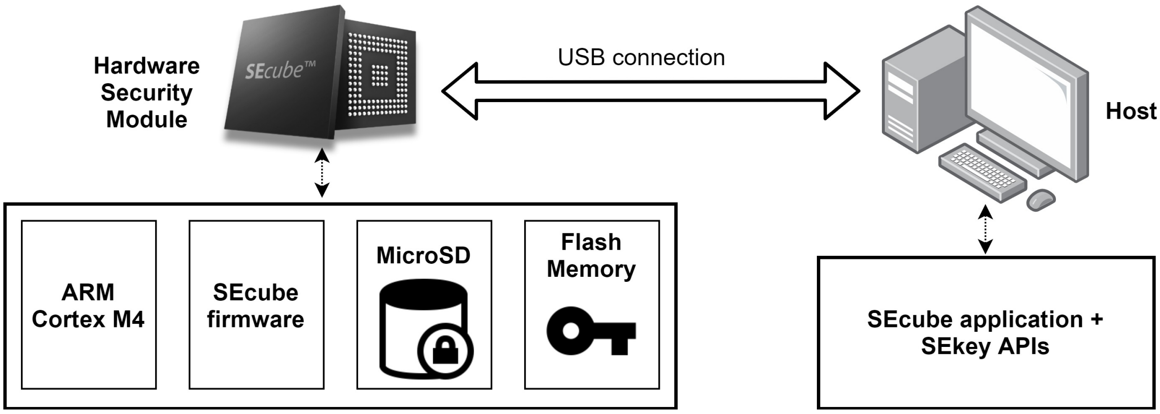

Figure 1.1: This is the image *caption*.

Table 1.1: Preliminary Experimental Results

Benchmark	**Inputs**	**Processing time**
SHA	Message of 100 KB	368449 s
RIJNDAEL	Message of 100 KB	1083568 s
DIJKSTRA	Matrix of 100x100 32-bit integers	324782 s
STRING	1331 50-char strings	178616 s
BITCOUNT	12800 32-bit integers	419545 s

1.1 Section title

This is a section under a chapter. The number of sections also contributes to greater readability of your text, and to a better display of the content in the index. In fact, sections are automatically shown in the Table of Contents. However, try not to make sections shorter than two pages. For smaller portions of your text, use subsections.

You can refer to a section using its label, using the \ref directive as for images, like this:

This concept has been explained in Section 1.1.

Remember to use the word Section with a capital S. This is also valid for chapters.

1.1.1 Subsection title

This is a subsection under the section.

The following is a table.

If you want to write a formula, you can do like this:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-ik\frac{2\pi}{N}n} \qquad k = 0, \ldots, N-1 \tag{1.1}$$

Tables and formulas are extensively documented online, and any doubts about their syntax can be easily resolved with a simple search. As for figures and sections, the same rules also apply to tables and formulas: mandatory reference in the text, possibility to use \label to label them, and naming with capital letter (e.g., “as in Table 1.1, as in Formula 1.1).

The following is a piece of code:

```
int func(int N, int M) {
  float (*p)[N][M] = malloc(sizeof *p);
  if (!p)
    return -1;
  for (int i = 0; i < N; i++)
    for (int j = 0; j < M; j++)
      (*p)[i][j] = i + j;
  print_array(N, M, p);
  free(p);
  return 1;
}
```

You can customize the style of your code, changing the language, the colors of keywords, of comments or the background by changing the settings inside the \ lstset directive found in the main file. Usually, the listings are not referenced within the text as happens for figures, tables, formulas and sections. Do not overdo the code within your text: use it only for short passages (e.g., function prototypes, or 2 to 5 lines of code within a function to help the reader in better understanding the meaning of the text).

You can also write in-text code using the \ lstinline directive,
like this: `int main(int argc, char** argv);`.

CHAPTER 2

Introduction

CHAPTER 3

Background

Password Manager

A password manager is a computer program that allows users to store, generate, and manage their passwords for local applications and online services. A password manager assists in generating and retrieving complex passwords, storing such passwords in an encrypted database, or calculating them on demand.

Types of password managers include:

- locally installed software applications
- online services accessed through website portals
- locally accessed hardware devices that serve as keys

Depending on the type of password manager used and on the functionality offered by its developers, the encrypted database is either stored locally on the user's device or stored remotely through an online file-hosting service. Password managers typically require a user to generate and remember one "master" password to unlock and access any information stored in their databases. Many password manager applications offer additional capabilities that enhance both convenience and security such as storage of credit card and frequent flyer information and autofill functionality.

Locally Installed Software Applications

Password managers commonly reside on the user's personal computer or mobile device, in the form of a locally installed software application. These applications can be offline, wherein the password database is stored independently and locally on the same device as the password manager software. Alternatively, password managers may offer or require a cloud-based approach, wherein the password database is dependent on an online file hosting service and stored remotely, but handled by password management software installed on the user's device.

Some offline password managers do not require Internet permission, so there is no leakage of data due to the network. To some extent, a fully offline password manager is more secure, but may be much weaker in convenience and functionality than an online one.

Online Services Accessed Through Website Portals

An online password manager is a website that securely stores login details. They are a web-based version of more conventional desktop-based password manager.

The advantages of online password managers over desktop-based versions are portability (they can generally be used on any computer with a web browser and a network connection, without having to install software), and a reduced risk of losing passwords through theft from or damage to a single PC although the same risk is present for the server that is used to store the users passwords on. In both cases this risk can be prevented by ensuring secure backups are taken.

The major disadvantages of online password managers are the requirements that the user trusts the hosting site and that there is no keylogger on the computer they are using. With servers and the cloud being a focus of cyber attacks, how one authenticates into the online service and whether the passwords stored there are encrypted with a user defined key are just as important. Another important factor is whether one- or two-way encryption is used.

Some online password management systems, such as Bitwarden, are open source, where the source code can be independently audited, or hosted on a user's own machine, rather than relying on the service's cloud.

The use of a web-based password manager is an alternative to single sign-on techniques, such as OpenID or Microsoft's Microsoft account (previously Microsoft Wallet, Microsoft Passport, .NET Passport, Microsoft Passport Network, and Windows Live ID) scheme, or may serve as a stop-gap measure pending adoption of a better method.

Locally Accessed Hardware Devices That Serve as Keys

Token-based password managers need to have a security token mechanism, wherein a locally-accessible hardware device, such as smart cards or secure USB flash devices, is used to authenticate a user in lieu of or in addition to a traditional text-based password or other two-factor authentication system. The data stored in the token is usually encrypted to prevent probing and unauthorized reading of the data. Some token systems still require software loaded on the PC along with hardware (smart card reader) and drivers to properly read and decode the data.

Credentials are protected using a security token, thus typically offering multi-factor authentication by combining something the user has such as a mobile application that generates rolling a Token similar to virtual smart card, smart card and USB stick, something the user knows (PIN or password), and/or something the user is like biometrics such as a fingerprint, hand, retina, or face scanner. There are a few companies that make specific third-party authentication devices, with one of the most popular being YubiKey. But only a few third-party password managers can integrate with these hardware devices. While this may seem like a problem, most password managers have other acceptable two-step verification options, integrating with apps like Google Authenticator and in-built TOTP generators. While third-party token devices are useful in heightening security, they are only considered extra measures for security and convenience, and they are not considered to be essential nor are they critical to the proper functioning of a password manager.

Chrome Extension

A browser extension is a small software module for customizing a web browser. Browsers typically allow a variety of extensions, including user interface modifications, cookie management, ad blocking, and the custom scripting and styling of web pages. The extension to allows the user to interact with the board has been developed using Chrome's WebExtension API, since Google Chrome is the most popular browser in the world. However, since the extension is not embedded in the browser, it can be installed on any browser that is Chromium-based.

CHAPTER 4

Implementation Overview

In this chapter you should provide a general overview of the project, explaining what you have implemented staying at a high-level of abstraction, without going too much into the details. Leave details for the implementation chapter. This chapter can be organized in sections, such as goal of the project, issues to be solved, solution overview, etc.
It is very important to add images, schemes, graphs to explain the original problem and your solution. Pictures are extremely useful to understand complex ideas that might need an entire page to be explained.
Use multiple sections to explain the starting point of your project, the last section is going to be the high-level view of your solution...so take the reader in a short 'journey' to showcase your work.

CHAPTER 5

Implementation Details

5.1 Tools used for the Chrome extension

Extensions are made of different, but cohesive, components. Components can include background scripts, content scripts, an options page, UI elements and various logic files. Extension components are created with web development technologies: HTML, CSS, and JavaScript. An extension's components will depend on its functionality and may not require every option.

In this project the extension was build with the following tools:

- HTML
- CSS
- Typescript
- Node
- React
- Webpack
- Material-UI

Let's see in the details each of these tools.

HTML

As said in the previous section, the extension is made with the web development technologies. HTML, HyperText Markup Language, is one of them. In this project, HTML does not have a central role, since it is managed automatically by webpack. However, when the extension is build, the popup and the options main page are in HTML.

CSS

CSS, Cascading Style Sheets, is another web development technology. In this project, CSS is lightly used to style the extension's UI. The majority of the styling part is done through React and Material-UI. However, it's important to mention CSS, since it heavily used and is a key part of the web development workflow.

Typescript

Typescript is a web development technology that is used to write code in a more readable and easy to understand way. In this project, Typescript is used to write the extension's logic. Typescript is a syntactic superset of Javascript and adds optional static typing to the language. Another difference compared to Javascript is that Typescript is compiled and not interpreted. This means that the extension will not run unless it is compiled. The reason behind the creation of Typescript was the maintenance of large-scale applications. So the decision of using Typescript was made to make the extension's code more maintainable. In addition, since Typescript has a static typing system, that enables static language analysis, which simplifies the devolopment process by providing meaningful errors.

Node

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes JavaScript code outside a web browser Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games). Node.js allows the creation of Web servers and networking tools using JavaScript and a collection of "modules" that handle various core functionalities.

npm is the pre-installed package manager for the Node.js server platform. It installs Node.js programs from the npm registry, organizing the installation and management of third-party Node.js programs. Packages in the npm registry can range from simple helper libraries to task runners.

React

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page, mobile, or server-rendered applications with frameworks like Next.js. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

Webpack

Webpack is a free and open-source module bundler for JavaScript. It is made primarily for JavaScript, but it can transform front-end assets such as HTML, CSS, and images if the corresponding loaders are included. Webpack takes modules with dependencies and generates static assets representing those modules.

Material-UI

Material UI is an open-source React component library that implements Google's Material Design. In this project is used to create the extension's UI. It includes a comprehensive collection of prebuilt components that are ready for use in production right out of the box. Material UI is beautiful by design, and features a suite of customization options that make it easy to implement your own custom design system on top of our components. The main advantages of material UI are:

- Ready to use

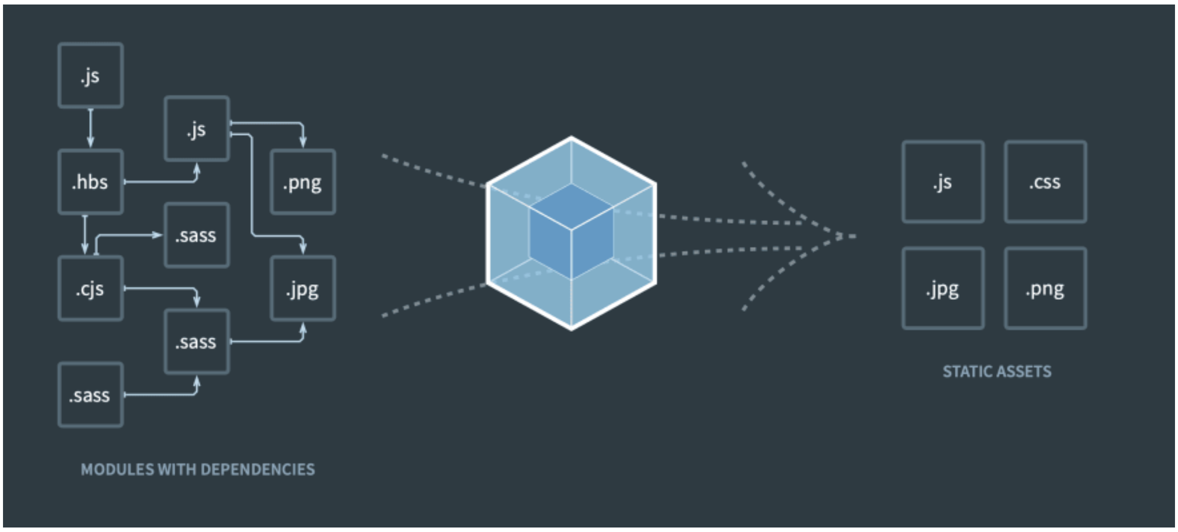

Figure 5.1: Webpack workflow

- Follow Google's Material Design so it is consistent with a big part of the web
- Realiable with a great community

Design Choices

In this section all design choices are explained. For each key part of the extension there is an image showing it and there is a complete explanation.

CHAPTER 6

Results

In this chapter we expect you to list and explain all the results that you have achieved. Pictures can be useful to explain the results. Think about this chapter as something similar to the demo of the oral presentation. You can also include pictures about use-cases (you can also decide to add use cases to the high level overview chapter).

6.1 Known Issues

If there is any known issue, limitation, error, problem, etc...explain it in this section. Use a specific subsection for each known issue. Issues can be related to many things, including design issues.

6.2 Future Work

Adding a section about how to improve the project is not mandatory but it is useful to show that you actually understood the topics of the project and have ideas for improvements.

CHAPTER 7

Conclusions

This final chapter is used to recap what you did in the project. No detail, just a high-level summary of your project (1 page or a bit less is usually enough, but it depends on the specific project).

Bibliography

[1] Donald E. Knuth (1986) *The TEX Book*, Addison-Wesley Professional.

[2] Leslie Lamport (1994) *LATEX: a document preparation system*, Addison Wesley, Massachusetts, 2nd ed.

APPENDIX A

User Manual

In the user manual you should explain, step-by-step, how to reproduce the demo that you showed in the oral presentation or the results you mentioned in the previous chapters.

If it is necessary to install some toolchain that is already well described in the original documentation (i.e., Espressif's toolchain for ESP32 boards or the SEcube toolchain) just insert a reference to the original documentation (and remember to clearly specify which version of the original documentation must be used). There is no need to copy and paste step-by-step guides that are already well-written and available.

The user manual must explain how to re-create what you did in the project, no matter if it is low-level code (i..e VHDL on SEcube's FPGA), high-level code (i.e., a GUI) or something more heterogeneous (i.e. a bunch of ESP32 or Raspberry Pi communicating among them and interacting with other devices).

APPENDIX B

API

If you developed some source code that is supposed to be used by other software in order to perform some action, it is very likely that you have implemented an API. Use this appendix to describe each function of the API (prototype, parameters, returned values, purpose of the function, etc).