Politecnico di Torino

Cybersecurity for Embedded Systems
01UDNOV

Master's Degree in Computer Engineering

Browser Password Manager

Project Report

Candidates:
Matteo Battilana (281389)
Salvatore Gabriele 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

Introduction

CHAPTER 2

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 3

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 4

Implementation Details

4.1 SECube Firmware

SEcube is the smallest reconfigurable silicon combining three main cores in a single-chip design. Low-power ARM Cortex-M4 processor, a flexible and fast Field-Programmable-Gate-Array (FPGA), and an EAL5+ certified Security Controller (SmartCard) are embedded in an extremely compact package. This makes it a unique security environment where each function can be optimised, executed, and verified on its proper hardware device [3]. The SECube device has been the selected platform that allowed to build the entire Secure Password Manager application on top of it. This is due to, as introduced before, to all the security feature that are intrinsically implemented in the chip itself, both from the point of view of the hardware and software.

In fact, one of the most important security enabling technology is the 3D SiP (System-in-package); the device consists of a number of integrated circuit, each one built on top of the others and enclosed into a single package.

Not only the system is secure from the hardware point of view, that is a necessary condition in order to develop a secure software, but the already present firmware includes some high level functions to generate a secure channel only after the authentication.

Authentication is performed from both parties, in which not only the firmware checks the authenticity of the Host but, the also the Host can do the same. This is possible thanks to a pre-shared key that is stored in the device at the very first initialization of the device itself. A custom C program, after the firmware has been uploaded, allows to setup the master password for the Secure Password Manager. This implies that the password used to authenticate the Host application is configurable only once, allowing to simplify the entire authentication process for the new feature without reducing the security. Further consideration about the user usability and the security aspects are available at Section 4.1.3.

The authentication is performed using a challenge-based authentication from both sides using a MAC (Message Authentication Code) implementation called Hash-based Message Authentication Code (HMAC), that uses a secret and an hash algorithm to verify the solution of the challenge. The challenge is created thanks to the internal TRNG (True Random Noise Generator), that is an hardware included in the chip that generates a true random sequence of bits. Once the challenge has been sent to the party to be authenticated and solved, the solution is read back and checked. If it corresponds to the expected one, the party is authenticated and the channel create. A small possible

weakness study is available at 4.1.3.

4.1.1 Flash memory management

Data are saved in the internal Flash memory, by using a C structure define as follow:

```
typedef struct se3_flash_pass_ {
  uint32_t id;
  uint16_t host_size;
  uint16_t user_size;
  uint16_t pass_size;
  uint8_t* host;
  uint8_t* user;
  uint8_t* pass;
} se3_flash_pass;
```

The Image 4.1 shows a simplified representation of all the used field.

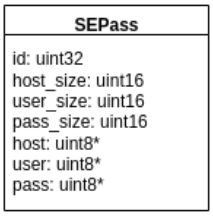

Figure 4.1: SECube firmware password record represented using ER database schema

Internally, some checks has been performed in order to inhibit the user to enter twice the same password record. This check is based on evaluating if the couple hostname and username is already present, and it is performed when the user modifies or creates a new password record. This has been made on purpose, since it has been assumed that a user can have two accounts for the same site and, even if wrong, he or she can use the same password for both of them. This has been implemented using a support method available in `se3_pass.c` file, refer to Code 4.2.

More precisely, data are stored in the flash memory using as few bits as possible to make the implementation able to store a large number of passwords, accordingly to the following table:

Field name	Number of bytes	Description
`id`	4	Id to univocally identify a password, using 32 bits
`host_size`	2	Number of character is the `host`
`user_size`	2	Number of character is the `user`
`pass_size`	2	Number of character is the `pass`
`host`	`host_size`	Hostname of the password record
`user`	`user_size`	Username of the password record
`pass`	`pass_size`	Plain text of the password record

Table 4.1: Flash memory for storing a Password record

The adopted solution allows to use 4 bytes for the id, and 2 bytes for the hostname, username and password length. The length of the other parameters are dependent on the previous three. This implies that at maximum $2^{16} = 65536$ characters can be used, but the system is able to reduce at minimum the occupied size, since the stored bits are not fixed.

Data into the Flash memory are stored as defined by the C structure shown at Table 4.1. Since the internal embedded Flash memory is limited, a possible attack could rely on the fact that creating few record with the hostname, username and password length set to the maximum could fill up the space. This corresponds a DoS (Denial of Service) attack, making the system not more fully usable, by saturating the internal memory.

A solution to this problem could be to limit the number of characters of the hostname, username and password itself to a maximum value. The problem is that even in this case, if the user has an access to the device and it is logged in, he or she will be able to saturate the memory, independently from the dimension of each field. For this reason and for possible future improvements based on using the unused bits in the fields, 2 bytes have been reserved for each field.

4.1.2 Code implementation

The entire solution has been developed using the C language and everything has been built on top of the already present firmware.

Besides all the adaptations to the code that has been necessary to the Secure Password Manager to work correctly, everything is based on two C files under the "*SEcube USBStick Firmware/Project/Src/Device*" directory:

- `se3_sepass.c`
- `se3_pass.c`

A coarse grain classification can be done on the level of data manipulation that the functions inside each one of the two files contains. In the case of `se3_sepass.c`, functions are much more command oriented, allowing to perform rather complex operation by calling directly a function from the just received command. On the other hand, `se3_pass.c` includes the functions for directly managing the Flash memory and abstract over some redundant operations, like the fetch from the storage.

Command dispatcher

The `se3_dispatcher_core.c` file contains the code implementation for managing the custom commands that are necessary to provide to the Host application, in order to manage the passwords.

In order to add the five different commands needed to manage all the Secure Password Manager features (add, delete, modify, search and password generation), a custom command, with id 13 has been added. The five methods have been added by exploiting the sub-command management; part of the command payload is used to identify the id of the method to call. The implementation and management is available at Code 4.1.

```
1  uint16_t sepassword_manager_utilities(uint16_t req_size, const uint8_t* req,
     ↪ uint16_t* resp_size, uint8_t* resp)
2  {
3    uint16_t operation; // the type of operation to be executed
4    memcpy((void*)&(operation), (void*)req, 2);
5    se3_flash_it it = { .addr = NULL};
6    if(!login_struct.y)
7    {
8      return SE3_ERR_ACCESS;
9    }
10
11   se3_flash_it_init(&it);
12   it.addr = NULL;
13   switch (operation)
14   {
```

```
15        case SE3_SEPASS_OP_ADD:
16        return add_new_password(req_size, req+2, resp_size, resp);
17        break;
18        case SE3_SEPASS_OP_MODIFY:
19        return modify_password(req_size, req+2, resp_size, resp);
20        break;
21        case SE3_SEPASS_OP_DELETE:
22        return delete_password(req_size, req+2, resp_size, resp);
23        break;
24        case SE3_SEPASS_OP_GET_BY_ID:
25        return get_password_by_id(req_size, req+2, resp_size, resp);
26        break;
27        case SE3_SEPASS_OP_GETALL:
28        return get_all_passwords(req_size, req+2, resp_size, resp);
29        break;
30        case SE3_SEPASS_OP_GENERATE_RANDOM:
31        return generate_random_password(req_size, req+2, resp_size, resp);
32        break;
33        default:
34        SE3_TRACE(("[sepassword_utilities] invalid operation\n"));
35        return SE3_ERR_PARAMS;
36      }
37      return SE3_OK;
38    }
```

Listing 4.1: "Code for searching if password record is already present"

se3_pass.c

As already introduced before, the `se3_pass.c` contains all the low level operations with the Flash memory. The most important available methods are the following:

- `se3_pass_find`: given a password id, returns true if that id used
- `se3_pass_new`: given a password record, create a new password record in the Flash memory
- `se3_pass_read`: read from the Flash memory the password information of a single password
- `se3_pass_equal`: return true if there is a password with the same hostname and same username. The implementation is available at 4.2

```
1   bool se3_pass_equal(se3_flash_pass* password, se3_flash_it* it)
2   {
3     bool areEquals = false;
4     se3_flash_pass tmp;
5     se3_flash_it_init(it);
6
7     while (se3_flash_it_next(it) && !areEquals)
8     {
9       if (it->type == SE3_TYPE_PASS)
10      {
11        se3_pass_read(it, &tmp);
12
13        if (tmp.id == password->id || (is_str_eq(tmp.host, tmp.host_size,
      ↪ password->host, password->host_size) && is_str_eq(tmp.user, tmp.user_size
      ↪ , password->user, password->user_size)))
14        {
15          areEquals = true;
16        }
17
```

```
18          if(tmp.host != NULL) {free(tmp.host);}
19          if(tmp.user != NULL) {free(tmp.user);}
20          if(tmp.pass != NULL) {free(tmp.pass);}
21        }
22      }
23
24      return areEquals;
25    }
```

Listing 4.2: "Code for searching if password record is already present"

se3_sepass.c

Differently form the `se3_pass.c` file, the `se3_sepass.c` contains high level oriented functions that are directly called by the sub-command command dispatcher for the password manager.

The available methods are the following:

- `add_new_password`: used to generate a new password. The parameters such as the hostname, the username and the password are extracted from the command parameters checked against the current state of the Flash. More precisely, besides the consistency checks are performed before parsing the parameters, the pair hostname and username is searched into the memory, if not present the new password record is created. One important aspect is that the id of the new password is generated outside the device, and it is duty of the Host application to provide a valid value. This has been done in order to increase the flexibility of the solution and to allow the Host to use any kind on enumeration.
- `modify_password`: similarly form the `add_new_password` function, the parameters are read and checked. Only if the id is present, the previous record is deleted and replaced by the new one with all the correct information.
- `delete_password`: this simply deletes the password record by a given id
- `get_all_passwords`: return a list of all the passwords. It is also possible to filter by username or hostname
- `get_password_by_id`: given an id, the password record is returned
- `generate_random_password`: given the length and the set of characters that must be used, a random password is generated using the internal TRNG.

The password can be generated by using a combination of four different character set:

- Lowercase: abcdefghijklmnopqrstuvwxyz
- Uppercase: ABCDEFGHIJKLMNOPQRSTUVWXYZ
- Number: 1234567890
- Symbol: -_.:;,?&%$!#

The `generate_random_password` has been implemented by exploiting the TRNG that generates the number of characters in bytes. This means that if the password to be generated is 100 characters long, the TRNG will be exploited to gather 100 bytes.

This has been done for a specific reason, each byte is used to select a character from the set that has been generated from the union of all enabled sets.

This solution allows to have that each character in the cumulative set has the same probability to be used.

This solution has been choose to avoid the problem of a non-uniform distribution of each character probability. If all sets are enabled and merged together to form a single set, as in this implementation, the probability that taking a random character from the newly generated password is exactly the '`A`' one is:

$$\Pr(\mathrm{X} = \mathrm{A}) = 100 \cdot \frac{1}{26+26+10+13} = 1.3\%$$

The firmware function used to generate a random password is based on randomly select N characters from the enabled sets (alphabet lowercase and uppercase, numbers and symbols). The current implementation, is totally random but is able to ensure that at least one character from all the enabled sets is selected. This has been implemented by performing a loop over the just generated password, if it does not contains at least one character for each enabled set, the computation is repeated. In order to avoid an infinite loop, the upper limit of the number of iterations has been set to 100.

4.1.3 Possible weakness

At the current state-of-the art of the implementation, there is not a way to retrieve or change the password if the used does not know the current one. This is an intrinsic weakness of the solution, since having only one factor authentication that is not supported by a second authentication method, the only way to gain access to the stored password would have been creating a support method that would have reduced the security. This generates a problem: giving the possibility to the user to retrieve the master password in some way, would have reduced the overall security. For this reason, even if the user usability could be reduced, the main focus was to not disclose the password to who is not able to be authenticated.

From the point of view of the HMAC used to perform the Host and Device mutual authentication the security is intrinsically provided by the fact that the secret is stored in a secure hardware. The fact that also the Host application has the ability to verify the Device using the same challenge-based authentication explained before, implies that the solution generated by the firmware has been generated by using the same shared key that is used to authenticate the Host application. The following sentences explain how an bruteforce attack could be used against this type of authentication, with focus on a particular use case.

From the Chrome Extension usability definition, the user has to enter the password in order to unlock the extension itself and be allowed to manage the password. If an attacker is doing *piggyback*, he or she will see only the number of inserted character (since characters will be replaces with black dots) or some pressed keys. The problem with this is that, if the firmware challenge has been generate and available to the attacker in some way (like stealing the SECube for some minutes), he or she can perform a bruteforce attack. The time will be drastically reduced if the number of character is known. One solution to solve this would be to not allow the Host application to authenticate the SECube itself, but this is not secure for obvious reasons. During the early development of this project, it was proposed to implement a delayed authentication; if the user performs X wrong login attempts, the next one must be done after Y seconds, otherwise an error is always generated. The fact that challenge is independently generated from the login API, makes also this solution unfeasible. The feasibility of the attack strictly depends on the importance of the stored passwords and it is a matter of effort/cost, since the high security level of the challenge used.

4.2 Host Middleware

The Host Middleware is a software intended to run in the user's PC (for example as a daemon on Linux or as a service on Windows) and to provide a secure connection between the user's PC and the SECube board. This means that it acts as a bridge between the Chromium Extension (the frontend for the user) and the board, thus the Middleware is developed with security in mind.

It's main job is to serve some HTTPs requests. In fact, it provides REST APIs to allow the Chromium Extension to interact with the features exposed by the SECube's firmware. This means that it acts as a web server with HTTPs support in order to provide a secure connections.

4.2.1 The web server

HTTPs is a secure protocol that uses a TLS connection to provide a secure connection between two endpoints, and it's a replacement for the HTTP protocol. This means that HTTPs provides the following benefits:

- Authentication
- Privacy: the connection is encrypted and the data is encrypted
- Integrity: the data is signed and the signature is verified

Thus, HTTPs helps to avoid the risk of eavesdropping, which is a risk that can be exploited by an attacker to intercept the data and modify it. More in general, it avoids *Man In The Middle* attacks.

The middleware is developed mainly in Python. To implement the web server, Flask is used as module. It natively supports HTTPs and a self-signed certificate is used, generated with *openssl*.

4.2.2 The REST APIs

The exposed APIs are totally complaint to the REST principles. It uses cookies to authenticate the user, and it uses JSON as exchange data format.

The main API is the one the allows to create a session. Once a session is created (via a successful authentication), a cookie is generated and sent to the browser. The browser will then store securely the cookie and the extension will automatically attach it to each request. In the end, the cookie is strictly needed to interact with all other APIs.

The endpoint to create a session is `POST /api/v0/device/0/session?pin=<pin>&enditme=<endt>`. The `pin` parameter is the PIN code of the board, used to unlock it, while the `endtime` parameter is the time in seconds until the session expires. When the session expires, the middleware will automatically invalidate it and each subseguent request will result in `403 Unauthorized`. The `endtime` parameter is a timestamp in seconds, and it's relative to the middleware's one. The middleware is capable of generating it internally and the current timestamp can be obtained via `GET /api/v0/time`. For a more complete description of the API, see the appendix B.

4.2.3 Session Management

When a cookie is created, it containts only the session ID. This ID is used to identify the session, and it's used to identify the user. The session ID is generated by the middleware and it's unique for each session. This means that on the user's side, only the ID is stored instead of any other sensitive

information. The user is protected by *client identity steal* attacks thanks to the security given by the browser in storing it locally.

On the middleware side, a session corresponds to a file stored in the file system, in the same directory as the middleware's executable. The file name is the session ID. The file contains the following information:

- The board's PIN given by the user when the session was created
- The endtime of the session (timestamp in seconds)

In order to avoid possible attacks because of the files stored in clear in the file system, the session is encrypted with a key that is generated by the middleware. At each startup of the middleware, a 2048 bit RSA key is generated and stored in the RAM, and each previously created session is invalidated and destroyed. The public/private keys are used to encrypt/decrypt on the fly the requested session file. The encryption/decryption is done on-the-fly and then the session is stored in the file system, so a non encrypted version of the file will never appear in the file system. Both the PIN and the endtime are encrypted, along with other side informations.

4.2.4 Timestamp and Timeout Management

In order to avoid the risk of *time-leap* or *time travel* attacks, the middleware uses an internal timestamp instead of the PC's one. So, even if a malicious user changes the PC's time, the middleware's timestamp will continue to update itself correctly and sessions will expire correctly.

In order to generate the timestamp, the middleware uses a dedicated thread that periodically (each seconds) updates the timestamp. The same information can be accessed via `GET /api/v0/time`. The timestamp is updated each second, so it's not possible to get a timestamp that is in the past.

At each request, the middleware:

1. finds the session file corresponding to the session ID in the request (associated to the cookie sent by the Extension)
2. decrypts the file
3. gets the stored endtime timestamp
4. get the current timestamp (so the middleware's timestamp)
5. compares the endtime with the current timestamp and if the endtime is reached (greater or equal), the session is invalidated and the request is replied with `403 Unauthorized`
6. gets the stored PIN
7. tries to authenticate the user with the PIN
8. if the *login* is unsuccessful, the request is replied with `403 Unauthorized`, otherwise the request continues with the operation requested by the user

4.2.5 How to interface Python with C++?

As mentioned above, the middleware is developed in Python. However, the HOST libraries used to communicate with the board are written in C++. This means that the Python code needs to be able to interface with the C++ libraries. In order to do that, the middleware ues the *Ctypes* module. It's

a builtin module that allows to interface with C libraries.

The actual implementation sees some wrappers (both for the *L0* and *L1* libraries). Those wrappers are C functions that uses only C's primitive types as arguments and as return values, because of the way ctypes works. An example of function wrapper is the following one, used to initialize the *L0* library:

```
#ifdef _WIN32
#include <Windows.h>
#define __MODIFIER __declspec(dllexport)
#else
#define __MODIFIER
#endif

#define EXPORT_FUNC(_type, _name) extern "C" __MODIFIER _type _name

EXPORT_FUNC(void *, createL0Instance)() {
    return new(std::nothrow) L0;
}
```

Once C wrappers are all defined and implemented, all the code needs to be compiled. In this case there is no *main* function, so there is no an executable file to run. Because those functions are meant to be called from the external (i.e. the Python code), the code is compiled as a shared library. On linux, they are compiled as *.so* files. On Windows, they are compiled as *.dll* files. The shared library is then loaded by the Python code.

With ctypes, the function wrapper is easily invoked. The first thing that is done is to tell to ctypes what are the arguments and return types of the function, then the function can be called. Because this procedure must be done for each function, a python class has been created to encapsulate the functions. There is a class both for the *L0* and *L1* libraries, and this is an example:

```
class L0:
    def __init__(self, path_lib: str):
        self._libname = path_lib
        self._c_lib = ctypes.CDLL(self._libname)

        # Create L0 instance
        self._c_lib.createL0Instance.argtypes = []
        self._c_lib.createL0Instance.restype = ctypes.c_void_p

        self._l0inst = self._c_lib.createL0Instance()
```

4.2.6 How to secure Python code?

The python code is not secure. While the C code is compiled into executable code and it's a very difficult job to reverse engineer, the Python code is directly interpreted on the fly, so there is no compilation and the code is in clear. Moreover, it can suffer from *code injection* attacks, so the code must be protected against those attacks.

In order to achieve this, a python library is used, called *pyarmor*. Its first job is to obfuscate the code, so write the code in such a way it can't be understood easily by an human. This protects against *code injection* and reverse engineering. The resulting obfuscated script is something like this:

```
from pytransform import pyarmor_runtime
pyarmor_runtime()
__pyarmor__(__name__, __file__, b'\x06\x0f...')
```

Moreover, the original python code (that now is obfuscated) is wrapped with functions that insert timers to prevent debugging and to prevent old stack traces dump. This is done by letting the python code run normally and at the end of each function call (where each function call creates a new stack frame), the frame is cleared.

Finally, the last precaution is to wrap everything into a single executable. The middleware is mainly composed of different python files, the shared library and the private and public keys. Using a library called *pyinstaller*, the middleware is packed into a single executable. What *pyinstaller* does is to create an executable that when is executed it creates a temporary folder and extract from itself all the files together with a standalone version of the python interpreter. This is done so that the middleware can be executed without the need to install the python interpreter and without the library installed. Moreover, it protects by possible *code injection* attacks done at the interpreter level. The final job of pyinstaller is to encrypt the content of the executable. Beucase the executable needs to know the key to decrypt it when it's requested by the user, the key is hardcoded into the executable itself, so it's relatively easy to obtain the key but a malicious user needs to do some reverse engineering of the executable itself so it's an increase of time spent on trying to attack the middleware.

4.3 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.

4.3.1 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.

4.3.2 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.

4.3.3 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.

4.3.4 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.

4.3.5 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.

4.3.6 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.

4.3.7 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

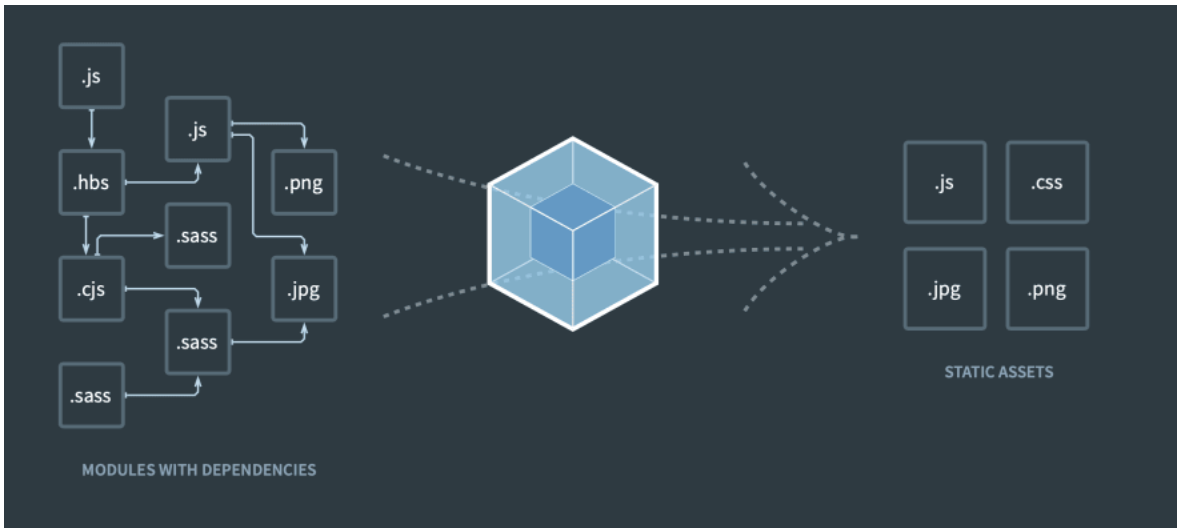

Figure 4.2: webpack workflow

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

4.4 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.

4.4.1 Popup and Tab Navigation

The two key elements of the extension are the popup and the tab navigation. The popup is the main window of the extension and the tab navigation is the bottom part of the popup, in which there are the tabs that are used to navigate between the different sections of the extension. The four tabs are:

1. **Tab**: it shows passwords, if there are, for the current website.
2. **My Vault**: it shows the passwords that are stored in the vault.
3. **Generate**: it allows the user to generate a new password.
4. **Add**: it allows the user to add a new password.

Another detail in the popup is the lock button. This button, when clicked, blocks the extension. This allows the user to force lock the extension, since the next time it will be opened it will ask for the master password. The whole popup with the component just described is shown in image 4.3

4.4.2 Tab

The tab section of the extension is shown in image 4.3. It's visible that when the user clicks on the extension, the popup opens and the Tab is the default section. In fact, in this section, the user can see the passwords that are stored for the current website.

This is done by retrieving the hostname of the website, and then searching for the website in the vault. If the website is found, the passwords are shown.

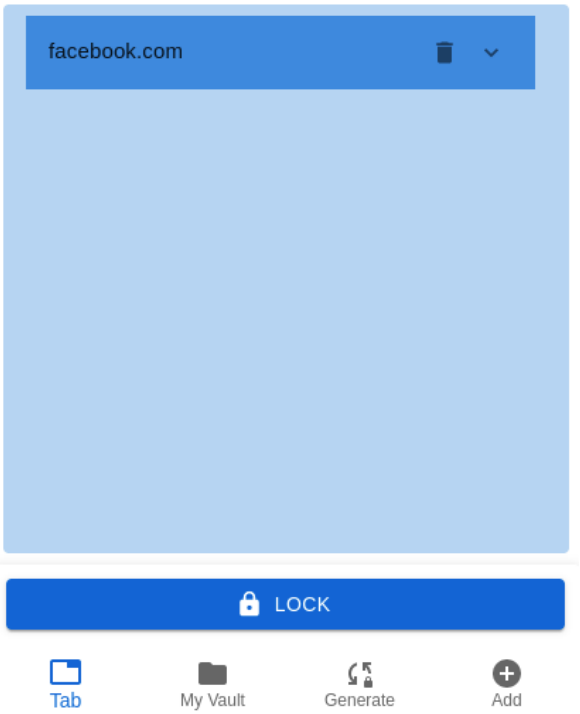

Figure 4.3: Popup, lock button and tabs

4.4.3 My Vault

The My Vault section is shown in image 4.4. It's visible that when the user clicks on the extension, the popup opens and the My Vault is the default section. In fact, in this section, the user can see the passwords that are stored in the vault.

Generate

This section is very important, since it is used to generate a new password. The user can choose the length of the password, and the type of characters that will be used. The supported type of characters are:

- **Uppercase**: uppercase letters
- **Numbers**: numbers
- **Special**: special characters

If no type of characters is selected, the password will be generated with lowercase letters only. The generated password is shown in image 4.5

Add

The last section of the extension is the Add, that is useful to add a new password to the vault. The logic behind the autocompletion of the hostname is the same as the one from the Tab section. The Add section is shown in image 4.6

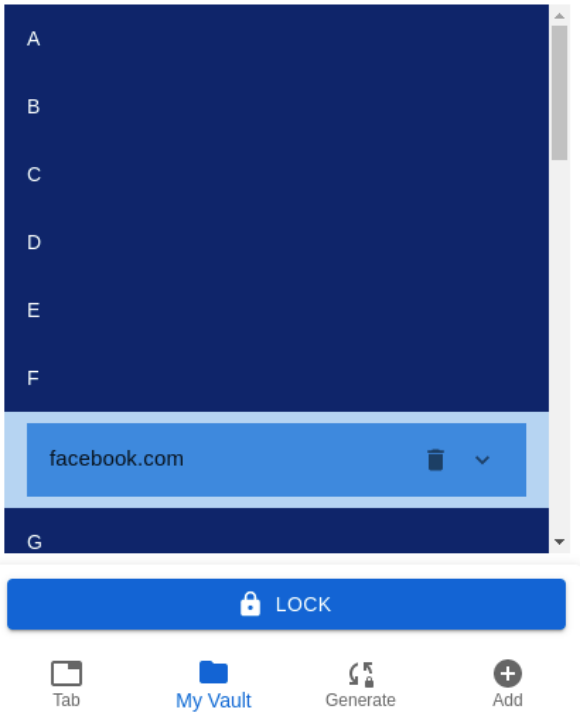

Figure 4.4: My Vault section of the extension

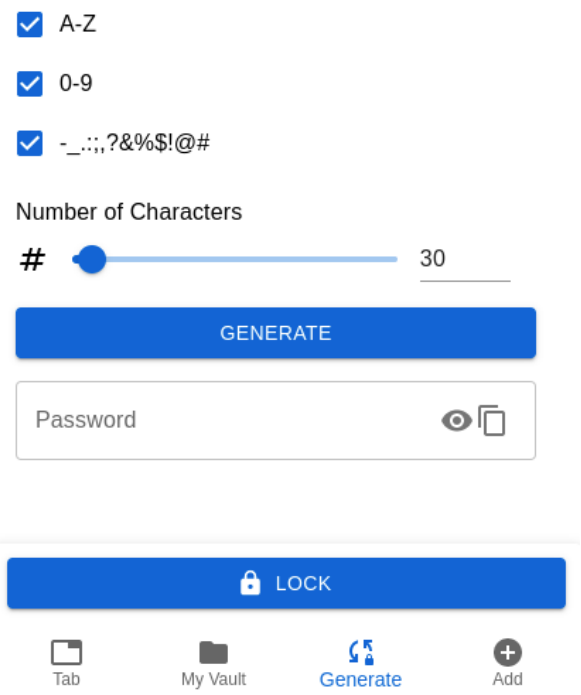

Figure 4.5: Generate section of the extension

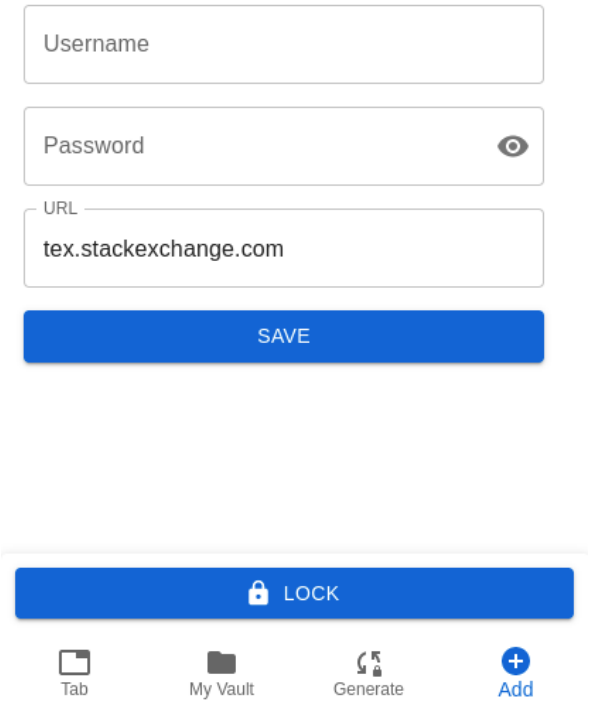

Figure 4.6: Add section of the extension

4.4.4 Options

The options are kept as simple as possible. In fact there are only two of them:

1. **Autocomplete**:it allows to autocomplete the login form when the page is uploaded
2. **Lock After**: it allows to specify the minutes after which the extension will be locked and ask again the password.

In order to retrieve the options when the users reopens the extension, the local storage is used. Since the options are not sensible data, it is not important that the data are encrypted, and in fact the local storage was the correct choice. In a future implementation, since Chome support by default the sync storage, the options can be stored in the sync storage so that each Chrome browser connected to the same Google account will have the same options. This uses the Google sync API and it can be taken as it is, supposing it is secure out of the box.

4.4.5 Expandable Menu and Buttons

In the section of the extension where we have the entries with the password, there always is a expandable menu. When expanded, the entry shows the username, the password and the URL. The expanded menu is shown in

To improve security, the password is hidden by default. This rule applies to all field where there is a password. However, if the user wants to see the password, there is an eye button at the end of the password field, that allows to show the password.

When the menu is expanded, near each field there are two buttons:

1. **Edit**: it allows to edit the field

2. **Copy**: it copies the content of the field to the clipboard.

In addition to the expandable menu there is also a button that allows to delete the password. When the button is clicked, the user is prompted with a confirmation dialog. The button of the confirmation dialog are colored in a way such that the delete button is red, and the keep button is green.

4.5 Autocomplete

This feature is a key part of the extension, since it improves drastically the user experience and usability. The extension is able to autocomplete the login form when the page is loaded. The idea behind the autocomplete is the following: when the page is loaded, the extension retrieves the username and password text fields. Then, the extension, knowing the current website, makes a request to the Middleware to retrieve the username and password of the website. If the username and password are found, the extension automatically fills the fields. Otherwise, nothing happens.

The autocomplete feature works in conjunction with the lock after. In fact, if the extension is locked, the autocomplete feature is disabled. Indeed, if the extension is unlocked, the autocomplete feature follows what the user set in the options.

4.6 Lock After

This feature is very important from two points of view: first, it allows the user to keep the extension unlocked for a certain period of time avoiding the need to enter the password every time the extension is opened. Second, it allows the user to lock the extension after a certain period of time, and ask again the password. So it touches both usability and security, finding a balance between the two. The is balance stands is the customizability; in fact the time, in minutes, after which the extension will be locked is configurable in the options.

The implementation is pretty straightforward. When the user unlocks the extension, a timestamp is sent to the Middleware, that saves it. After that, at every request, the Middleware checks if the current timestamp is greater than the saved timestamp. If it is, the Middleware returns 403, and so the extension is locked. Otherwise, the Middleware returns the asked data.

CHAPTER 5

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).

5.1 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.

5.2 Future Work

CHAPTER 6

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.

[3] What is SEcube, `https://www.secube.blu5group.com/`.

APPENDIX A

User Manual

A.0.1 Firmware

This tutorial needs a working version of Eclipse for C/C++ and the AC6 Tools are properly installed in order to build the firmware and flash it to the SECube device. The configuration describes uses the following configuration:

- ST-Link/v2 programmer
- Eclipse IDE for C/C++ Developers (includes Incubating components) Version: 2022-03 (4.23.0) Build id: 20220310-1457
- Ubuntu 18.04 5.4.0-117-generic

Hardware and Compiler setup

In this Section are reported the instructions you need to follow to properly connect the SEcube DevKit to the Host PC and to Programmer/debugger, refer to Figure A.1. A more detailed configuration tutorial is available via the official SECube Wiki.

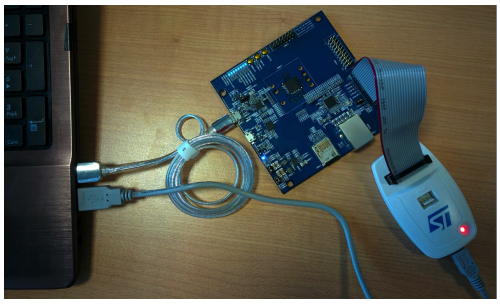

Figure A.1: Connection between the STLink/v2 programmer and the SEcube DevKit

Assembling is composed of the following two steps in order to obtain the situation that is available at Figure A.2:

1. Connect the SEcube DevKit with the programmer by means of the JTAG/SWD cable: the cable should be inserted on the JTAG docks on both the programmer (in this case the orientation of the plug is forced from the dock) and the DevKit (in this case you must pay attention in inserting the plug on top of both lines of connectors and with its protrusion oriented towards the inner side of the DevKit).

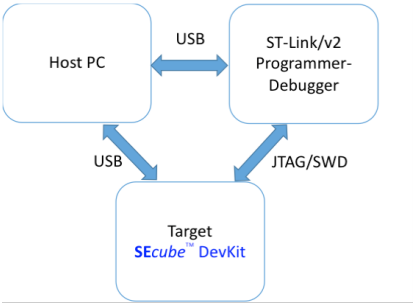

Figure A.2: System Architecture

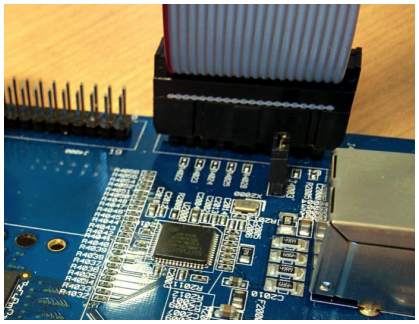

Figure A.3: Connection between the STLink/v2 programmer and the SEcubeTM DevKit, close-up (highlighted in red) on the JTAG connector orientation

2. Connect the ST-Link/v2 with the PC by means of the USB cable.

The system assembled is shown in Figure A.1, while a close-up on the JTAG connection is in Figure A.3.

In order to be able to build and flash the firmware, the AC6 Tools must be installed via Eclipse. The AC6 Tool will install the GNU Embedded Toolchain for ARM, which is a ready-to-use, open source suite of tools for C, C++ and Assembly programming targeting ARM Cortex-M and Cortex-R family of processors. It includes the GNU Compiler (GCC) and is available free of charge directly from ARM for embedded so ware development on both Windows and Linux operating systems. The reference platform for this document is the System Workbench for STM32 (SW4STM32) Eclipse plugin.

SW4STM32 is an integrated environment that includes:

- Building tools (GCC-based ARM cross compiler, assembler and linker);
- OpenOCD and GDB debugging tools;
- Flash programming tools

To install SW4STM32 as an Eclipse plugin:

1. launch Eclipse IDE
2. on the toolbar, click Help Install New Software...
3. in the Available Software window, click Add
4. in the Add Repository window, set Name and Location fields as follows, and then click OK:
    - Name: System Workbench for STM32 - Bare Machine edition

- Locaton: http://www.ac6-tools.com/Eclipse-updates/org.openstm32.system-workbench.site

5. select OpenSTM32 Tools and click Next
6. accept the license agreement and click Finish to start the plugin installation, continue the installation also if a warning for incompatible or unsigned components is prompted
7. restart Eclipse

Firmware flashing

Once the SECube has been connected to the Host computer via both the ST-Link/v2 programmer and the USB connection, the firmware can be imported, compiled and flashed.

The latest firmware version has been already compiled and available at "*SEcube USBStick Firmware/Project/Eclipse/USBStick/Release/USBStick.elf*" but it is possible to import the project into Eclipse and recompile it. If you want to use the pre-compiled version, you can skip the following sections and go to Section A.0.1.

In order to import the firmware and the software for performing the init, you need to click File and then Import..., as in Figure A.4. At this point after having selected Existing Projects into Workspace (Figure A.5), the first two projects must be imported (Figure A.6).

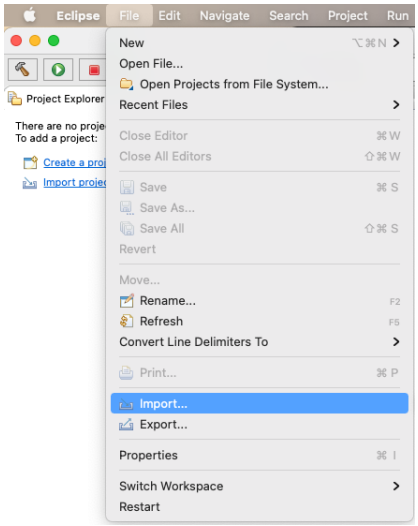

Figure A.4: Project import in Eclipse

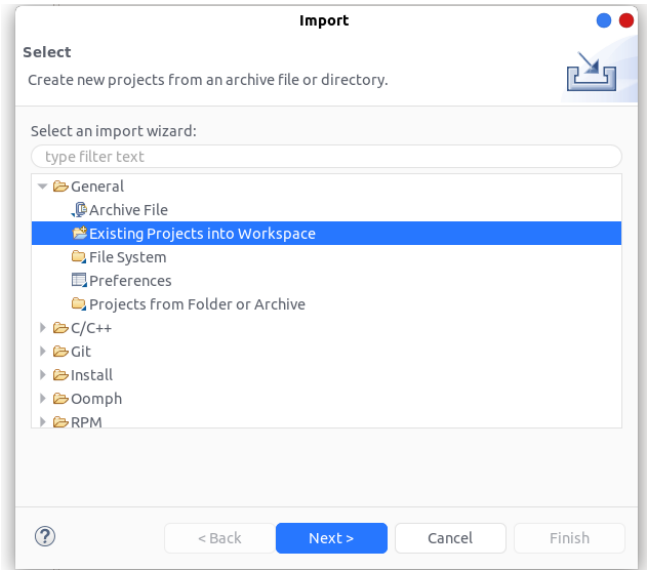

Figure A.5: Import of projects

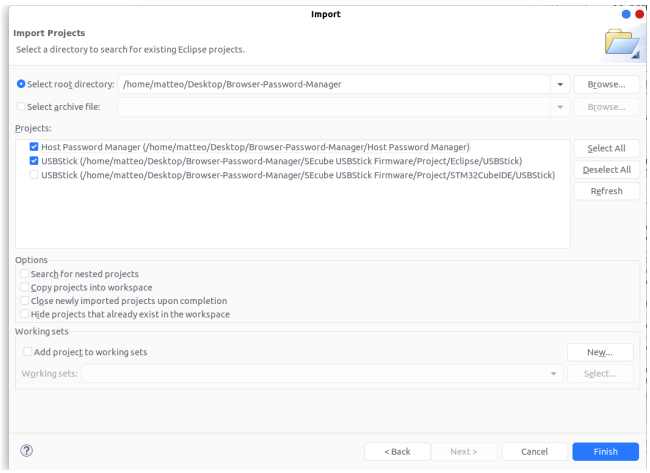

Figure A.6: USBStick firmware and Host Password Manager for Initilization

The first project will be used during configuration of the device while the second one is the firmware itself.

At this point, on the right you should have two projects, you have to Right click over the “USB-Stick” one and select “Build Project” (refer to Figure A.7).

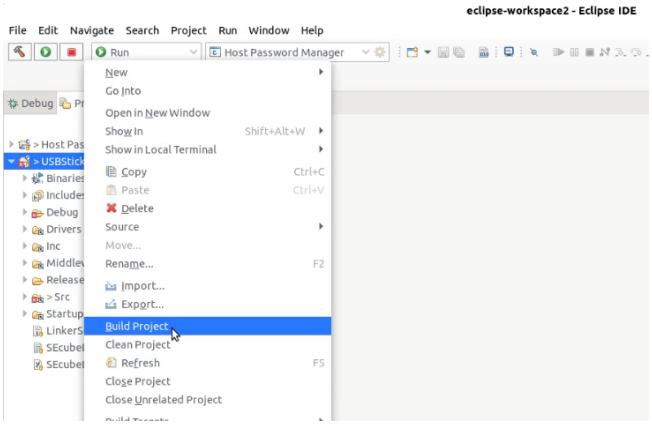

Figure A.7: Build the firmware

Configuring the device

Once the firmware, you have to first of all to erase the chip in order to remove the previous pin configuration, by doing a Right click on the project and then “Target” and then “Erase Chip...”. Once it has finished, you can flash the firmware into to the device by clicking “Target” and then “Program Chip...” (refer to Figure A.8). In the next window you have to select the “Release” version and flag the “Reset after program” before clicking “OK”.

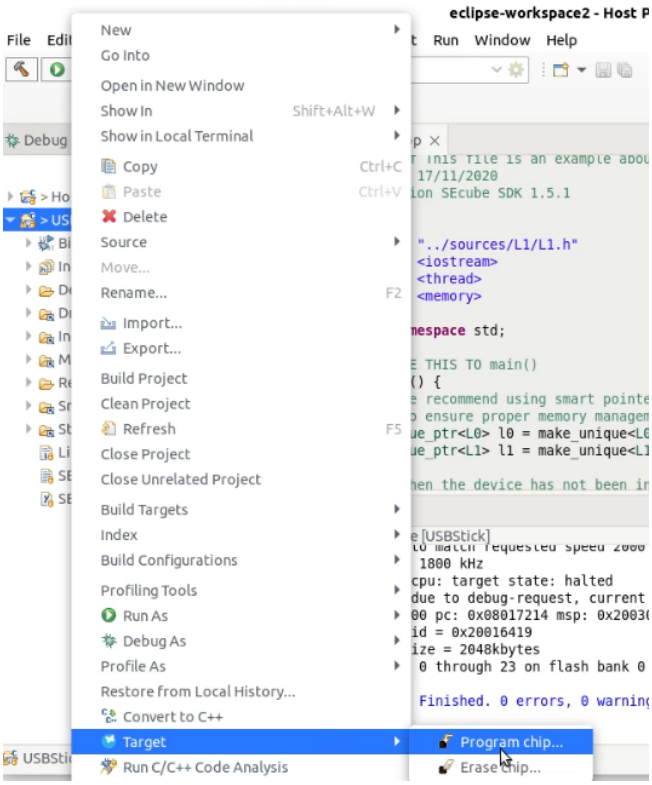

Figure A.8: Flash the firmware

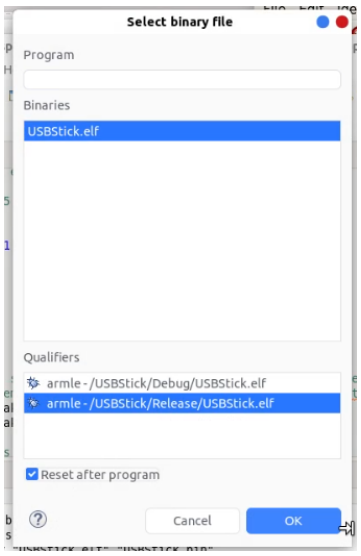

Figure A.9: Firmware Release version selection

At this point, once the firmware has been flash, you need to configure the device by setting a pin. Now it is the turn of the second project called "Host Password Manager".

You have to open the `device_init.cpp` file and check that the name of the first method is set to `main`. At this point, you can perform the compilation and run the program. This allows to initialize the device and set the master password for the Secure Password Manager (at line 49 it is possible to change the pin). From now, the device is ready to receive commands by the Host Middleware application in order to manage all passwords features.

A.1 Host Middleware

This section will describe how to build and run the host middleware, both on Linux and Windows. The build process is not necessary because a ready-to-run executable will be provided for the two opearting systems. However, if there are problems in executing them, the build process can be used as a workaround.

> ☞ Because of all the dependencies and operations to achieve a build, some problems may occurr. This section will try to indicate all the necessary eoftware that are required, but unfortunately the successful build of the host middleware is not guaranteed because of the heterogeneous nature of computers.

A.1.1 Linux

Luckily, on Linux the build process is pretty straightforward. The only thing that needs to be done is to install the necessary software. The following is a list of software that is required to build the host middleware.

- **Python 3.9.x** (tested with 3.9.7). Check that your PATH environment variable points to the Python executable *python3*.
- **pip 20.3.4** (tested with 20.3.4). Check that your PATH environment variable points to the pip executable *pip* and refers to the correct Python version.

- **gcc 11.x** tested with 11.2.0. Check that your PATH environment variable points to the gcc executable *gcc*.
- **g++ 11.x** tested with 11.2.0. Check that your PATH environment variable points to the g++ executable *g++*.
- **GNU Make** tested with GNU Make 4.3. Check that your PATH environment variable points to the GNU Make executable *make*.
- **git** Check that your PATH environment variable points to the git executable *git*.

Here a few steps to build the host middleware if all the required software is installed correctly. Note, it can change depending on the used linux distribution. It may requre further steps to install the dependencies.

```
$ git clone https://github.com/SEcube-Project/Browser-Password-Manager.git
$ cd Browser-Password-Manager/HostMiddleware

# To build the shared library
$ make clean
$ make -j4 lib.so

# To run the scrypt as-is (include build of lib.so)
$ make clean
$ make -j4 run

# To compile, obfuscate and pack into a single executable
$ make clean
$ make -j4 dist
$ ./BPMMiddleware
```

Listing A.1: bash version

A.1.2 Windows

Unfortunately, on Windows there is a lot more work to do.

Python

Python 3.9.x is needed. If you are not sure it's installed on your system, try to launch a Powershell console and type `python --version`. If you get `Python 3.9.0` or similar means that Python is installed. Otherwise, you need to install it.

☞ **ATTENTION**: if the Windows Store opens up, close it! You need to install it in the *classic way* otherwise strange things will happens later one.

☞ **ATTENTION**: if you installed Python from the Windows Store, unistall and download it from the official Python webpage.

☞ **ATTENTION**: if python is not available from the Powershell after manual installation, try to reboot. If it's still not available, you need to manually specify the Python's executable path. Start men, type Python, right-click and select "Open File location". Most likely it will head you to the Start Menu Shortcuts, so right-click again on the Python 3.9 folder and click on "Open file location". Select the path and copy in the clipboard.

In the start menu, search for "environment" and click "Edit the system environment variable". Click on "environment variables" button, select "Path", click "Edit", click "New" and paste the path you previously copied.

Confirm and close everthing, the Powershell too. Open it again and check if now python is available.

C++ Compiler - Buildtools

Head to the start men and look for `x64 Native Tools Command Prompt for VS 2022` (if you are on a 32 bit system, look for `x86 Native Tools Command Prompt for VS 2022`). Open it, and a terminal emulator will show up. Type `cl`. If you get *'cl' is not recognized as an internal or external program...* means that something is missing, otherwise you will get the following message and it means that the compiler is installed. Same reasoning must undergo with the `link` command.

```
C:\Program Files\Microsoft Visual Studio\2022\Community>cl
Microsoft (R) C/C++ Optimizing Compiler Version 19.32.31329 for x64
Copyright (C) Microsoft Corporation.  All rights reserved.

usage: cl [ option... ] filename... [ /link linkoption... ]

C:\Program Files\Microsoft Visual Studio\2022\Community>link
Microsoft (R) Incremental Linker Version 14.32.31329.0
Copyright (C) Microsoft Corporation.  All rights reserved.

 usage: LINK [options] [files] [@commandfile]

   options:

      /ALIGN:#
      /ALLOWBIND[:NO]
      /ALLOWISOLATION[:NO]
      /APPCONTAINER[:NO]
      /ASSEMBLYDEBUG[:DISABLE]
      /ASSEMBLYLINKRESOURCE:filename
      /ASSEMBLYMODULE:filename
      /ASSEMBLYRESOURCE:filename[,[name][,PRIVATE]]
      /BASE:{address[,size]|@filename,key}
      /CLRIMAGETYPE:{IJW|PURE|SAFE|SAFE32BITPREFERRED}
      /CLRLOADEROPTIMIZATION:{MD|MDH|NONE|SD}
      /CLRSUPPORTLASTERROR[:{NO|SYSTEMDLL}]
      /CLRTHREADATTRIBUTE:{MTA|NONE|STA}
      /CLRUNMANAGEDCODECHECK[:NO]
      /DEBUG[:{FASTLINK|FULL|NONE}]
      /DEF:filename
```

```
31        /DEFAULTLIB:library
32        /DELAY:{NOBIND|UNLOAD}
33        /DELAYLOAD:dll
34        /DELAYSIGN[:NO]
35        /DEPENDENTLOADFLAG:flag
36        /DLL
37        /DRIVER[:{UPONLY|WDM}]
38        /DYNAMICBASE[:NO]
39        /EMITVOLATILEMETADATA[:NO]
40  (press <return> to continue)
```

Listing A.2: bash version

If something is missing (or the Visual Studio's Command Prompt Tool is not available), Visual Studio must be installed. Go to https://visualstudio.microsoft.com/downloads/ and download Visual Studio Community edition. Once the installer is downloaded, launch it, select *Visual Studio Community 2022* (click on Modify if Visual Studio is already installed) and select *Desktop Development with C++*. The following parts must be installed:

- MSVC v143 - VS 2022 C++ x64/x86 build Tools
- Windows 10 SDK
- C++/CLI support for v143 build Tools
- C++ Modules for v143 build tools
- C++ Clang tools for Windows

Repeat from the beginning, be sure that the *Visual Studio's Command Prompt* is installed and the compiler is available.

How to build

Now everything should be installed. Open the Visual Studio's Command Prompt, head to the Host-Middleware folder (with the CD command) and type `compile_win.bat`. It will compile everything and pack into a single BPMMiddleware.exe executable that eventually you can run, if everything went fine.

Pay attention to antivirus software, it may block the executable. If it does, you can try to disable the antivirus temporarily. Pay attention to use the right Visual Studio's Command Prompt version: if you are on a 64 bit system, use `x64` instead of `x86`.

If there are problems in packing the executable, you can always try to run directly the python code as-is (under the condition that the dll has been compiled correctly): run `python app.py`.

APPENDIX B

API

Middleware HTTPs' API

B.0.1 /api/v0/time

Used to work with the timestamp. The timestamp is an integer in seconds. The supported HTTP methods are:

- `GET`: returns the current timestamp.

B.0.2 /api/v0/devices

Used to work with the devices. It allows to obtain all the connected boards, in particular for each device the ID, Name and Serial are returned. The API is currently not used by the Extension because it' supposed that only one device at a time is connected. The supported HTTP methods are:

- `GET`: returns the list of devices.

B.0.3 /api/v0/device/{id}/sessions

Used to manage sessions. The supported HTTP methods are:

- `GET`: allow to know if the cookie attached to the request represents a valid session or not.
- `POST`: creates a new session. The *PIN* and the *timestamp* parameters are mandatory. The *timestamp* parameter is an integer in seconds.
- `DELETE`: forces to invalidate the session attached to the cookie.

B.0.4 /api/v0/device/{id}/generate

Used to generate a new password using the exposed funcionality of the board. The supported HTTP methods are:

- `GET`: allows to obtain a new randomly generated password. The optional parameters are:

  `length`: the length of the password. The default value is 64.

  `upper`: boolean value that indicates if the password must contain uppercase letters. The default value is 1. Can be 0.

`special`: boolean value that indicates if the password must contain special characters. The default value is 1. Can be 0.

`numbers`: boolean value that indicates if the password must contain numeric characters. The default value is 1. Can be 0.

B.0.5 /api/v0/device/{id}/passwords

Used to manage passwords. The supported HTTP methods are:

- `GET`: allows to obtain the list of passwords. It supports the `hostname` parameter to filter the list of passwords by hostname. The `hostname` parameter is a string and it can be partial or complete. For example, if the `hostname` parameter is `mple.com`, then the list of passwords will contain passwords that have as hostname `www.example.com` or similar ones. Each password is represented by a JSON object with the following fields:

  `hostname`: the hostname of the password.

  `password`: the password.

  `username`: the username.

  `id`: the ID of the password.

- `POST`: allows to add and store in the board a new password. The parameters must be passed via the body in the form of a JSON object. The mandatory parameters are:

  `hostname`: the hostname of the password.

  `password`: the password.

  `username`: the username.

B.0.6 /api/v0/device/{id}/password/{id}

Allows to manage a single password. The supported HTTP methods are:

- `GET`: allows to obtain the password record. The password is represented by a JSON object with the following fields:

  `hostname`: the hostname of the password.

  `password`: the password.

  `username`: the username.

  `id`: the ID of the password.

- `DELETE`: allows to delete the password.

- `PUT`: allows to update the password. The parameters must be passed via the body in the form of a JSON object, as the one to add a new password. The mandatory parameters are:

  `hostname`: the hostname of the password.

  `password`: the password.

  `username`: the username.