FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Security Analysis of Blocks Lockers
Student:	Eliška Krátká
Supervisor:	Ing. Josef Kokeš
Study program:	Informatics
Branch / specialization:	Computer Security and Information technology
Department:	Department of Computer Systems
Validity:	until the end of summer semester 2022/2023

Instructions

1) Research current methods of goods delivery using automated pickup boxes in Czech Republic (e.g. Alza boxes, Mall boxes, etc.).
2) Focus on Property Blocks' Block Lockers. Describe their implementation aspects - hardware, software, interfaces, protocols.
3) Perform a threat analysis of these boxes. Evaluate the seriousness of found threats using a suitable methodology.
4) Perform a security analysis of the source codes of the boxes with focus on the most serious threats discovered in the previous section.
5) Discuss your discoveries, propose modifications to the code that would fix or mitigate any discovered vulnerabilities.

Electronically approved by prof. Ing. Pavel Tvrdík, CSc. on 22 February 2022 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Security Analysis of Blocks Lockers

Eliška Krátká

Department of Information Security
Supervisor: Ing. Josef Kokeš

May 19, 2022

Acknowledgements

I want to express my gratitude to the thesis supervisor, Ing. Josef Kokeš, for his guidance, feedback, quick email replies, and BI-BEK lectures, which motivated me to study information security in the first place. I am also thankful for my Tangus friends, who morally supported me during my studies. Special thanks to Kuba, without his support, this thesis would never get finished. Last but not least I am incredibly grateful to my dear friend Alex who gave me the chance to examine the Blocks lockers, and to the whole Blocks team.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. I further declare that I have concluded an agreement with the Czech Technical University in Prague, on the basis of which the Czech Technical University in Prague has waived its right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act. This fact shall not affect the provisions of Article 47b of the Act No. 111/1998 Coll., the Higher Education Act, as amended.

In Prague on May 19, 2022 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Krátká, Eliška. *Security Analysis of Blocks Lockers*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2022.

Abstrakt

Tato bakalářská práce se zabývá bezpečnostní analýzou chytrých skříněk Blocks se zaměřením na admin zónu. Práce obsahuje shrnutí konkurence a popis implementace. Hrozby jsou modelovány pomocí metodiky STRIDE a risk je hodnocen dle CVSS metriky. Testovací útok potvrzuje identifikované zranitelnosti a kroky k jejich zmírnění či odstranění jsou formulovány.

Klíčová slova Blocks, chytré skříňky, výdejní boxy, modelování hrozeb, bezpečnostní analýza, 2FA, brute-force, Firebase

Abstract

This thesis covers a security analysis of the Blocks smart lockers with a focus on the admin console. A summary of competitors is added. A description of the implementation is provided. Threats are modelled using the STRIDE methodology, and the CVSS metric calculates the risk. Testing exploitation confirms identified vulnerabilities, and their mitigation is formulated.

Keywords Blocks, smart lockers, delivery boxes, threat modelling, security analysis, 2FA, brute-force, Firebase

Contents

List of Figures

List of Tables

Introduction

Due to the growth of e-commerce and the digitalisation of buildings, the market with automated deliveries is on the rise. The main goal of the thesis is to perform a security analysis of one of the leading solutions on the Czech market, Property Blocks' pickup lockers, also known as Blocks.

Blocks is a technology startup that provides a solution for automated deliveries, contactless sharing, and flexible storage. The business model focuses mainly on the office buildings, but they are also expanding in the business centres. Among Blocks' clients fit companies such as Deloitte, O2, or Penta, and monthly their boxes pass through over ten thousand packages, and the number grows fast. Last year, Blocks reported sales of 15 million CZK [1].

The most seen part of Blocks' business is the delivery boxes. Module boxes are made of wood or sheet metal and come in two versions, one with a touchscreen, realised by an iPad inside the locker, and the second one with no touchscreen, providing contactless user interaction through the mobile app.

The software behind the lockers works as a separate unit and can be used to operate boxes of any third party, which is the direction Blocks focus on in the long run. There is usually not enough focus on the security aspects of emerging technologies, and it may be displaced due to other priorities such as fast delivery to the production market. Since Blocks are quite a new startup[1] and already has clients in the big companies, it is interesting to evaluate the security of their solution.

The first chapter introduces Blocks' solution and looks closer into the implementation aspects. We also give a brief overview of goods delivery status using automated pickup boxes on the Czech market and provide research on used technologies across competitors.

We dedicate the second chapter to threat modelling. We identify and calculate the risk of threats, which helps us set priorities for what to investigate during the security analysis described in the third chapter.

[1] Operating since 2019

We focus on security evaluation of the Blocks admin console, a tool for remote management of the boxes. The admin console enables real-time data reporting, automatic software updates and most importantly, remote locker access to the admins. We observe the user interface and analyse the source codes to determine the system vulnerabilities that modelled threats can exploit.

To verify the identified vulnerabilities, we perform testing exploitation of the console and present an overview of the attack scenarios which the system could face. Lastly, we discuss our discoveries and propose security modifications to lower the risk bound with identified vulnerabilities.

Chapter 1

Blocks Introduction

This chapter presents an overview of the smart lockers providers on the Czech market with a description of technologies used for accessing the storage. We introduce the Blocks Lockers solution and architecture by profiling the system components.

1.1 Smart Lockers Producers in the Czech Republic

Over forty companies provide a smart lockers solution as a part of their business model. Most of them use smart lockers to automate package delivery, but only a few, including Blocks, enable smart lockers as a way of flexible storage.

They are several ways to access the box. The easiest way is a simple keypad, usually in the form of a touchscreen, where the customer opens the box by entering a PIN code. Another form of access is scanning a QR code, which can usually be found in a mobile application. Some producers enable contactless access through a mobile application, realized by Bluetooth Low Energy (BLE) or scanning an NFC card or keychain. Leading competitors can be seen in the table below, and the complete list is in Appendix B.

Table 1.1: Main Smart Lockers Competitors

Company Name	**Use Case**	**Ways of Access**
Blocks Lockers	storage, packages	BLE, keypad, scanner, NFC
AlzaBox	packages	keypad, scanner
PilulkaBox	packages	BLE
Z-BOX	packages	BLE

Figure 1.1: Blocks Lockers

1.2 Blocks Structure Overview

Blocks consist of module boxes made of wood or sheet metal. Some boxes have a touchscreen for entering an access code, and some are fully contactless, where the customer opens the dedicated locker through the mobile app.

Inside the module box, there are chained desks with electronic locks, furnished by an external supplier, Kerong[2]. Based on the version of the box, there is an iPad that serves as a touchscreen[3] or a hidden Raspberry PI inside the box, which provides the logic behind contactless interaction. Locks and iPads are connected to the router inside, which creates an internal network and serves as a firewall.

Raspberry PI and iPad run the same internal application, iPad only with an extra user interface (UI) layer. iPads are managed by SimpleMDM[4], a mobile device management solution for the supervision of enterprise devices. They run in the Single App Mode, which forces the iPad to have opened only the selected internal app, preventing the use of other apps.

The mobile application enables easy access to Blocks Lockers. It provides a PIN code and QR code and offers the opening of the box through BLE. To use the Blocks solution for enterprise buildings and the usage of the mobile app in general, the customer must have a user account. The Blocks' website

[2] `https://www.kerong.hk`
[3] C.2
[4] `https://simplemdm.com/`

serves mainly as promotional material and is also an option for user profile registration, so the customer is not obligated to use the mobile app.

The mobile app supports login through email or Facebook and on iOS also through Apple ID. During registration, the customer is also obligated to provide their phone number. In the app, users can see their own assigned and connected boxes and create new personal storage.

Client Apps are available on the leading mobile platforms, iOS and Android. They fulfil the same function but have specific UI differences concerning target platform design guidelines. Both apps are free to download, the iOS version as "Blocks App" from the App Store, Android one as "Blocks" from Google Play. iOS App is developed in Swift, Android version in Kotlin.

The whole system is managed by the admin console, which is provided to retailers and business centres as part of the Blocks solution and the boxes. Admin console presents an overview of all activities and data (orders, locker capacities, active users), enables software updates and allows remote locker opening. It is implemented as a web application, accessible to users on `https://admin.blocks.cz/`.

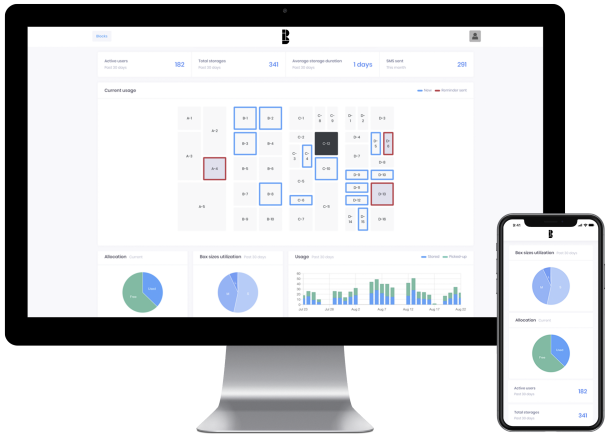

Figure 1.2: Blocks' Admin Console

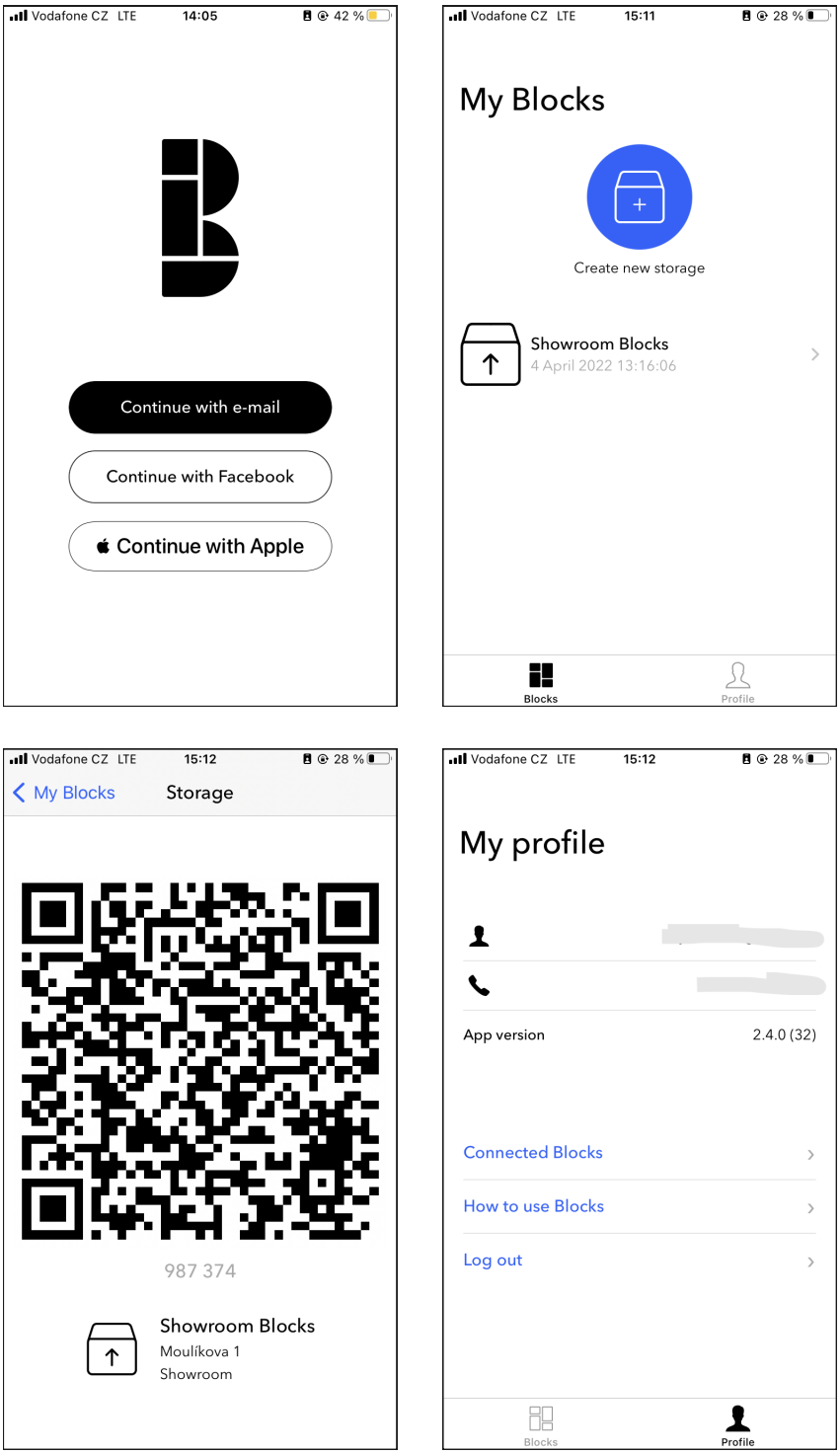

Figure 1.3: iOS Mobile App Overview

1.3 Profile of the System Components

This section divides the Blocks system into smaller components, which gives us an overview of the attack surface and helps us look for the threats during the threat modelling.

1.3.1 Users

First, we describe the system's users and their access rights. We gather them into four user groups listed below, ordered by their access right from the lowest to the highest.

Customer uses Blocks as storage or for package delivery. They access the box contactless through the mobile application or the touchscreen on the locker. For using the mobile app, they must have a user account. They log in with an email and password, and they can register in the app or on the Blocks website.

Messenger delivers packages to the lockers. They access the box through the touchscreen on the locker.

Administrator uses the admin console, which provides an overview of all lockers' activities and data. Through the console, the admin can remotely open any box locker.

Supervisor solves problems and defects with the boxes on the spot and takes care of the software updates. They can cancel the Single App Mode on the iPad, open the boxes, and service Raspberry PI. They have the highest access rights.

1.3.2 Interfaces

The users communicate with the system through the interfaces.

Mobile Application customers to access their lockers easily. It provides a PIN code and QR code and offers the opening of the box through BLE. The app requires the customer to log in to the user account.

Web Application - Blocks' website serves mainly as promotional material. Additionally, it allows the customers to create a user account.

Touchscreen is realized by the iPad inside the locker. The customers use it to enter delivery PIN/QR code and manage their box storage. Through the touchscreen, the messenger saves the packages into the designated boxes. The supervisor takes care of the iPad software updates.

Raspberry PI is hidden inside the lockers, invisible to the typical customer. It handles the logic behind the contactless box access and is accessible only to the supervisor.

Admin Console allows administrators to manage the whole system. It provides real-time data reporting, automatic software updates and remote locker access.

1.3.3 Data Stores

Data stores are system components which control and store the data.

Firebase is a third party database server which handles data about users, boxes and deliveries.

SimpleMDM is a mobile device management tool for monitoring enterprise Apple devices, alias iPads in the Blocks system.

Chapter 2

Threat Modeling

In order to provide a well-grounded security evaluation of the Blocks Lockers, we follow verified threat modelling methodologies. This chapter describes the outline of the threat analysis process and introduces techniques we use for finding and classifying threats. Terms such as "threat" or "vulnerability" are phrases belonging to the common language. To avoid misunderstandings, we start by introducing these terms in the context of information security.

An asset is any data, device or another component of an organisation's systems that is valuable [2]. In the security analysis, we investigate how these assets are protected against attacks by malicious outsiders or insiders [3]. We do it in the risk assessment [4], a process by which risks are identified and the impact of those risks determined. A risk is a measure of the extent to which an entity is threatened by a potential circumstance or event [5], in other words, a chance of something happening which will have an unwanted or adverse impact on the asset [6, 7]. A threat is any incident that could negatively affect an asset [2] and a vulnerability is a flaw that can be exploited by a threat to destroy, damage, or compromise an asset [2]. Therefore, to calculate the risk, we determine the threats that are likely to exploit the system and identify vulnerabilities that allow them to do so.

During the threat modelling, we use the software-centric approach. We start with an analysis of Blocks' architecture. We use gathered information from the previous chapter, and the components and users described we consider as the targets of the attack.

Before looking for threats, we answer a simple question. What do we want to protect and why? It is essential to define the security objectives – what is the target user, what are the assets worth protecting, and what would be the impact if compromised. It helps us prioritise what we should focus on during the threat modelling process.

For the identification of the threads, we use the STRIDE model. We go over each system component and examine if any threats that fall into the STRIDE categories exist. By using STRIDE, we enumerate the things that

might go wrong. The goal of the model is not to categorize every single thing we find but to help us find possible threats we would not otherwise think of. We consider how the attacker could achieve the attack and what he needs to make the attack successful for each threat, which helps identify the vulnerabilities we later look for [8, 9].

We classify the threads and calculate their risk using the CVSS framework. We pick the threats that pose the highest security risk and focus on them while examining the software source codes. We look for the vulnerabilities that threats can exploit and allow us to perform an attack on the system.

Lastly, we evaluate the results and formulate security considerations. We ask if we should accept the risk, address the threat, avoid altogether, or transfer.

The outline of the analysis process is inspired by a paper published by the Institute of Electrical and Electronics Engineers (IEEE), *A Threat Analysis Methodology for Security Evaluation and Enhancement Planning* [3] which proposes a generic methodology for threat modelling.

2.1 STRIDE

STRIDE is a verified threat modelling tool developed by Microsoft, designed to help software developers to identify various types of threats [10].

Acronym STRIDE stands for six threat categories, Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.

Table 2.1: STRIDE Model [10]

Threat	**Property Violated**	**Definition**
Spoofing	Authentication	Pretending to be something or someone else
Tampering	Integrity	Modifying data
Repudiation	Non-repudiation	Claiming to not performed an action
Information Disclosure	Confidentiality	Providing information to someone not authorized to see it
Denial of Service	Availability	Absorbing resources needed to provide service
Elevation of Privilege	Authorization	Allowing someone to do something they are not authorized to do

2.2 CVSS

The Common Vulnerability Scoring System (CVSS) is a threat ranking method providing a consistent approach to addressing threats. CVSS is owned and managed by a non-profit organization FIRST.Org, Inc. and opened to the public freely for use [11]. For the calculation of the score, we use the latest version, CVSS 3.1 and we do so with an online tool also published by FIRST[5]. We apply only the Base metric group, without the addition of temporal and environmental scoring.

The Base metric group represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments [11]. It consists of the following three groups of metrics.

2.2.1 Exploitability Metrics

Exploitability metrics reflect the characteristics of the vulnerable system component [11].

Attack Vector (AV) reflects from "where" the attack can occur [12]. The metric values are **Network (N)** if the set of possible attackers extends beyond the other options, up to and including the entire Internet, **Adjacent (A)** if the attack is limited at the protocol level to a logically adjacent topology, **Local (L)** if the vulnerable component is not bound to the network and the attacker's path is via read/write/execute capabilities, and **Physical (P)** if the attack requires the attacker to physically touch or manipulate the vulnerable component [11].

Attack Complexity (AC) describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability [12]. It is divided into two categories, **Low (L)** if specialized circumstances do not exist and attacker can repeatedly perform a successful attack to the vulnerable component or **High (H)** if a successful attack depends on conditions beyond the attacker's control [11].

Privileges Required (PR) represent the level of privileges an attacker must possess before successfully exploiting the vulnerability. It consist of three metric values, **None (N)** if the attacker is unauthorized prior to attack, **Low (L)** if the attacker requires privileges that provide basic user capabilities or can access only non-sensitive resources, and **High (H)** if the attacker requires privileges that provide significant control over the vulnerable component allowing access to component-wide settings and files [11].

User Interaction (U) captures the need for action from human user [12]. It is either **None (N)** or **Required (R)** [11].

[5] `https://www.first.org/cvss/calculator/3.1`

2.2.2 Impact Metrics

Impact metrics express how much can exploited vulnerability impact the component that suffers the worst from the outcome of the attack [11].

Confidentiality Impact (C) measures whether a successful attack can lead to the acquisition of confidential data, regardless of their importance, and to what extent, whether partially, **Low (L)**, or completely, **High (H)** [12]. **None (N)** means there is no loss [11].

Integrity Impact (I) expresses the extent to which there is a risk of data corruption or modification [12]. **None (N)** if there is no loss, **Low (L)** if the amount of modification is limited and **High (H)** there is a complete loss of protection [11].

Availability Impact (A) of a successfully exploited vulnerability [13]. **None (N)** if there is no loss, **Low (L)** if there are interruptions in resource availability and **High (H)** if here is a total loss [11].

2.2.3 Scope (S)

Scope (S) has its own category. It express whether a vulnerability in one system or component can have carry over impact on another system or component (**Changed (C)**) or not (**Unchanged (U)**) [14].

2.2.4 Base Score Formula

The Base Score acquires a value between 0 to 10, with 10 representing the highest risk. Each metric has an associated value, and the overall score depends on sub-formulas for Impact Sub-Score (ISS), Impact, and Exploitability [11].

Table 2.2: **Base Metrics** [11]

Metric	Metric Value	Numerical Value
Attack Vector	Network	0.85
	Adjacent	0.62
	Local	0.55
	Physical	0.2
Attack Complexity	Low	0.77
	High	0.44
Privileges Required	None	0.85
	Low	0.62 (0.68 if Scope is Changed)

	High	0.27 (0.5 if Scope is Changed)
User Interaction	None	0.85
	Required	0.62
Confidentiality / Integrity / Availability	High	0.56
	Low	0.22
	None	0

2.3 Identification of Threats

In this section, we focus on identifying threats with the help of STRIDE modelling. We go over each STRIDE category and analyse threats that could exploit the Blocks system with the focus on our primary asset, the typical Blocks customer, someone who either uses Blocks lockers as a storage solution or for package delivery. Therefore, our primary goal is to secure the belongings and deliveries stored in the locker and ensure no unauthorised access. Compromising the box access would be unpleasant for the user and cost the company's reputation.

2.3.1 Spoofing

Spoofing is the act of pretending to be someone or something from a known, trusted source [15]. The attacker can spoof a particular person, declares themselves to be a role a person represents in a system to perform an attack or spoof a machine [10]. The system property spoofing violates, is the authentication, which is the process of verifying the identity or other attributes of an entity (user, process, or device) [16].

2.3.1.1 Taking Over Real Customer Account

The attacker takes over a real customer account and forbids the customer to log in. One way to do this is by stealing login credentials to Facebook/Apple ID, which the customer uses for login. The attacker hacks into it and changes the password for that service. The other way is by changing the user account password. Both the mobile app and website do not offer the "change password" feature, and the only way to change the password is through the "forgot your password" option in the mobile app. Therefore, the attacker must have access to the customer's email and initiate a password reset attempt through the app.

Base metrics: AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L

ISS =	1 - [(1 - Confidentiality) × (1 - Integrity) × (1 - Availability)]
Impact =	
If Scope is Unchanged	6.42 × ISS
If Scope is Changed	7.52 × (ISS - 0.029) - 3.25 × (ISS - 0.02)^{15}
Exploitability =	8.22 × AttackVector × AttackComplexity ×
	PrivilegesRequired × UserInteraction
BaseScore =	
If Impact \<= 0	0, *else*
If Scope is Unchanged	Roundup (Minimum [(Impact + Exploitability), 10])
If Scope is Changed	Roundup (Minimum [1.08 × (Impact + Exploitability), 10])

Figure 2.1: Base Score Formula [11]

Score: 5.6[6]

2.3.1.2 Breaking Into Admin Account

The attacker breaks into a real admin account to perform an attack. They can do so by breaking the login to the admin console, which can be achieved by either cracking the account password or performing a password reset if the attacker has access to the admin's email.

Base metrics: AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H

Score: 9.8[7]

2.3.1.3 Breaking Into Blocks' Member Apple ID

The attacker breaks into Blocks' member Apple ID in the Apple Developer Program. Therefore they have access to all Blocks' apps (the mobile app for customers and the enterprise app running on the iPad). The attacker has to know the email of the targeted Blocks member and break their password. Ideally, the Blocks member does not have active two-factor authentication with their Apple ID.

Base metrics: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L

Score: 7.3[8]

[6] `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L`

[7] `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`

[8] `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L`

2.3.1.4 Spoofing User Identity

The attacker claims to be a customer to perform an attack (for example, stealing the box's content) which can be achieved in ways listed below:

1. The attacker steals the user's mobile phone, breaks the phone's security (phone lock) and accesses the logged-in Blocks app.

2. The attacker knows or has already guessed (for example, from misleading error messages) the user's login email and cracks the account password.

3. The attacker has access to the user's email and performs a password reset.

4. The attacker hacks into the customer's Facebook or Apple ID and uses it to log in to the application.

Base metrics: AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L
Score: 5.6[9]

2.3.1.5 Spoofing Supervisor Identity

The attacker pretends to be a supervisor. They can achieve it by cancelling the Single App Mode on the iPad. To break the Single App Mode, the attacker must know the gesture and the supervisor's PIN code.
Base metrics: AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
Score: 7.4[10]

2.3.1.6 Spoofing Blocks' Identity

The attacker pretends to represent the company by sending phishing emails that seem like password reset attempts or emails with PIN codes. They know what the email structure looks like to make it look credible and know the emails of Blocks customers. They can use the error messages in the mobile app to find out if the email is in the database or directly break into the database containing customers' emails.
Base metrics: AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Score: 3.1[11]

[9] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:L/A:L
[10] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H
[11] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

2.3.2 Tampering

Tampering is the intentional but unauthorized action of modifying data [17]. The attacker can tamper with a specific file (modification of JSON), memory (logs), or data flowing over the network [10]. Tampering violates integrity which refers to system property that data or information have not been altered or destroyed in an unauthorized manner [18].

2.3.2.1 Changing of Access Rights to the Storage

The attacker modifies access rights to the storage and blocks access to eligible users (for example, the customer cannot get to his package). The attacker does not need complete predominance over the account when targeting a particular customer. All there is to do is break into it and change the access rights to the storage (delete the storage in the app). When the attacker targets the whole userbase, he can pursue denial of service (see section 2.3.5.1).

Base metrics: AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L
Score: 6.3[12]

2.3.2.2 Modifying List of iPads in MDM List

The attacker modifies the list of enterprise iPads so they can no longer be tracked or managed (for example, in case of robbery). To do so, they need access to the SimpleMDM.

Base metrics: AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:L/A:L
Score: 3.8[13]

2.3.2.3 Forging Bluetooth Communication

The attacker spoofs the customer's identity during BLE communication. Thanks to monitoring, they can kick out the customer from the connection and pretend they are the customer by sending modified data leading to information leakage.

Base metrics: AV:L/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N
Score: 3.6[14]

2.3.3 Repudiation

Repudiation is claiming not performing an action. Threats in this STRIDE category are often associated with gathering logs. If a system does not log,

[12] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L

[13] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:N/I:L/A:L

[14] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N

does not retain logs, or does not analyze logs, repudiation threats are hard to find [10]. A property violated is non-repudiation, system protection against an individual or entity falsely denying having performed a particular data related action [19].

2.3.3.1 Stealing Content of the Lockers

The attacker spoofs a supervisor identity, breaks into the admin account, or already has the admin/supervisor rights in the first place (the attacker can be someone from the inside), opens specific locker or lockers and steals the content inside. The attack's success lies in the logging system behind the admin interface (the admin console) and the supervisor's interface (the iPad which serves as the locker's touchscreen).

Base metrics: AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
Score: 6.7[15]

2.3.4 Information Disclosure

Information disclosure is the act of allowing information to someone not authorized to see it [10]. Information disclosure can be against the process, allowing the attacker to extract secrets by reading error messages, against the data store such as reading data from the admin console, or against data flow by analyzing traffic.

2.3.4.1 Reading Error Messages

The attacker extracts secrets from error messages which contain information the attacker should not see. Error messages visible to the attacker with no privileges can occur in the mobile app and in the admin console during the login, registration or password reset attempt.

Base metrics: AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Score: 5.3[16]

2.3.4.2 Reading Data From the Admin Console

The attacker breaks into the admin account and gets access to sensitive data.

Base metrics: AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
Score: 7.2[17]

[15] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

[16] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

[17] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

2.3.4.3 Reading Data From the Apple Developer Account

The attacker has access to Blocks' member Apple ID and analyses sensitive data about Blocks' applications.

Base metrics: AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L
Score: 6.3[18]

2.3.4.4 Monitoring Bluetooth Communication

The attacker analyses communication between the client (Blocks' customer) and the box. The BLE communication must be unencrypted, or the attacker found out the the key.

Base metrics: AV:L/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Score: 2.5[19]

2.3.5 Denial of Service

Denial of service (DoS) threats absorb resources needed to provide service. The impact is on the availability of the attacked component in the system, in the worst case making it inaccessible to its intended users. DoS threats accomplish it by consuming network resources, absorbing memory, filling the data store or making enough requests to slow down the system [10, 20].

2.3.5.1 Changing the Access Rights of the Storage

The attacker blocks eligible users from accessing their locker by modifying storage data in the admin console. The attacker can either break into the console or have access already.

Base metrics: AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H
Score: 7.2[20]

2.3.5.2 Changing Data in the Apple Developer Account

The attacker has access to the Apple ID of Blocks' **developer** (member of the team who has developer rights in the Apple system) and changes data about Blocks' application running in the Apple environment. Since the developer can remove the application from the App Store in a few clicks, the attacker could cause a DoS of the Blocks lockers because the iPads run an iOS application.

Base metrics: AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

[18] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:L

[19] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

[20] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

Score: 7.2[21]

2.3.5.3 Putting Box Service Out of Function

The attacker puts the box out of function so it cannot give out and store packages. They can achieve it by spoofing the supervisor's identity or consuming local network resources.

Base metrics: AV:L/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H
Score: 5.1[22]

2.3.6 Elevation of Privilege

An elevation of privilege occurs when an application allows someone to do something that should not be available to them [21]. The system property violated is, therefore, an authorization, a process of determining whether a subject is allowed to have the specified types of access to a particular resource [22].

We have identified an existing vulnerability falling into the category later in the analysis, see section 3.2.5.

2.4 Findings

As a result of the threat analysis, we gathered a list of threats that could put Blocks' lockers at risk. Breaking into the admin account is a threat with the highest CVSS score and, together with other threats such as stealing the content of the lockers, poses a high risk on the admin console. The admin console allows administrators to manage the whole Blocks environment. Breaking into the admin account would compromise our primary security objective, to secure the belongings and deliveries stored in the locker and ensure no unauthorised access.

Table 2.3: List of Threats

Threat	CVSS Score
Breaking Into Admin Account	9.8
Spoofing Supervisor Identity	7.4
Breaking Into Blocks' Member Apple ID	7.3
Reading Data From the Admin Console	7.2
Changing the Access Rights of the Storage	7.2
Changing Data in the Apple Developer Account	7.2
Stealing Content of the Lockers	6.7

[21] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H

[22] https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:L/AC:H/PR:N/UI:N/S:U/C:N/I:N/A:H

Changing of Access Rights to the Storage	6.3
Reading Data From the Apple Developer Account	6.3
Taking Over Real Customer Account	5.6
Spoofing User Identity	5.6
Reading Error Messages	5.3
Putting Box Service Out of Function	5.1
Modifying List of iPads in MDM List	3.8
Forging Bluetooth Communication	3.6
Spoofing Blocks' Identity	3.1
Monitoring Bluetooth Communication	2.5

Chapter 3

Security Analysis of the Admin Console

We conclude that the admin console could be the significant weak spot of the Blocks' system. If the attacker breaks into the admin account, they can gather sensitive information, such as delivery data, lockers' capacities and Blocks' customers. Most importantly, they can open any locker that the system admin controls remotely. Therefore, this chapter aims to provide a security analysis of the admin console, focusing primarily on the sign-in user flow.

We split the analysis process into four parts, starting with the static application security testing (SAST), a testing methodology performed on the source codes while the program is not running. SAST is a helpful tool for identifying bugs and security flaws and can save time and effort in the early stage of the security analysis. However, it has its limitations, especially with identifying security vulnerabilities such as authentication and access control issues [23, 24]. Therefore, in the second part of the analysis, we focus on these potential problems while examining the login page from the UI perspective. In the third part, we perform exploitation of the found vulnerabilities using the password brute-force attack. At last, we connect the gathered information and discuss our findings.

3.1 Static Code Analysis

The admin console is written in TypeScript, a programming language which extends JavaScript by adding type definitions. This section describes the tools we use for the static code analysis and demonstrates the configuration.

3.1.1 SonarQube

SonarQube[23] is a powerful SAST toolset which focuses on code security. It supports multiple programming languages, including JavaScript/TypeScript. For testing purposes, we use the open-source Community Edition. We run SonarQube locally using a Docker container on port 9000. For the analysis, an additional module sonar-scanner must be installed.

From the analysis report, we can see no significant security issues. SonarQube detected two bugs and eight security hotspots with a low priority, referring to the same issue. Security hotspots are pieces of code which have to be reviewed manually to be able to tell if they pose a security risk or not. We can see that SonarQube presents a false positive warning when looking closer. One bug is connected to the node_modules, part of the npm package manager, which we installed locally as a prerequisite for performing SAST and therefore, it is not part of the admin console application. The only relevant bug is connected with localized strings, a file needed to translate strings to different languages.

Figure 3.1: Analysis Report - Modules

Figure 3.2: Analysis Report – localized strings

3.1.2 Supply Chain Attack

Supply Chain Attack exploits security vulnerabilities of software project dependencies. If a project relies on a vulnerable component, it could be a potential security issue [25, 26]. The examined admin console is a JavaScript/TypeScript application that relies on numerous third-party system dependencies. Each of them can have many more dependencies on its own. We use an npm module audit-ci[24] to help us find if there are any reported vulnerabilities in admin console project dependencies. As we can see from the screenshot, the project passed the test.

[23] `https://www.sonarqube.org/`
[24] `https://docs.npmjs.com/cli/v8/commands/npm-audit`

```
> blocks-admin@1.0.0 audit-dependencies
> audit-ci --config audit-ci.json

audit-ci version: 6.2.0
NPM audit report summary:
{
  "vulnerabilities": {
    "info": 0,
    "low": 0,
    "moderate": 0,
    "high": 0,
    "critical": 0,
    "total": 0
  },
  "dependencies": {
    "prod": 1,
    "dev": 84,
    "optional": 0,
    "peer": 0,
    "peerOptional": 0,
    "total": 84
  }
}
Passed npm security audit.
```

Figure 3.3: Security Audit by nmp

3.2 Weak Spots

The admin console is implemented as a web application, accessible on `https://admin.blocks.cz/`. The page's entire content for the unlogged visitor consists of a simple sign-in form, and the "forgot password" option. We inspect the login page from the UI perspective and list the weak spots.

3.2.1 Format of URL

The admin console should not be accessible to the general public by default, but even though the link is not presented anywhere on the Blocks website, the URL, thanks to its format, is not hard to determine.

The admin console is vulnerable to "forced browsing", a method when the attacker, through brute force or just simple guessing, finds a link to the hidden page. The issue presents a typical example of "security by obscurity" when

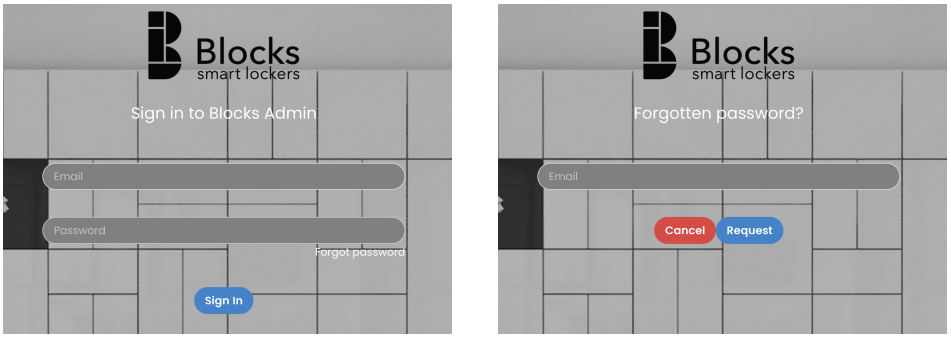

Figure 3.4: Sign-in and Forgotten Password

the admin console not being public serves as one layer of security protection [27, 28, 29].

3.2.2 HTML Autocomplete Enabled

The console has the autocomplete feature enabled, which might improve the user experience but contains a security risk. Autocomplete means that the admin does not have to enter their login credentials every time they use the console but can store them in the browser cache [30]. Most modern browsers implement integrated password management, which is generally seen as a good practice due to the fact the users do not have to remember their passwords and therefore, they can choose a stronger encrypted password than they would typically do. For this reason, setting autocomplete "off" on purpose in the web development process does not prevent the browser from asking the user if they would like to save the login information [31, 32].

However, the other function of the attribute is recalling previous values entered in the same input field [33], which can pose a security risk, as it suggests the attacker the previously used admin emails.

Figure 3.5: Web Inspector

From the web inspector and the source codes, we can see that the autocomplete function is allowed because of the missed implementation by the web developer. The email and password fields do not contain the HTML autocomplete attribute. Because the web developer did not explicitly set the attribute's value to off, the browser can use the entered email credentials as suggestions for future users filling the fields [30].

3.2.3 One Factor Authentication

The admin console has only one layer of authentication implemented, the user password. Since the admin has access to and can modify sensitive data, it is not an excellent approach to rely on a single protection method. The sign-in process misses the 2-Factor authentication of the user, which would lower the

risk associated with weak or compromised passwords or automatic password competition mentioned in 3.2.2 [34].

3.2.4 No Limit on Failed Login Attempts

From the UI investigation and verification in the source code, we can notice no limit on failed login attempts implemented, which makes the application vulnerable to password brute-force attacks. During exploitation we describe in 3.3.2, we discovered the vulnerability is not, in fact, real. The application does solve this issue.

3.2.5 Authorization

Blocks provided us with an account for testing purposes of the system in general. The account does not have admin rights to manage any lockers, yet we can successfully log in to the console. We cannot log in at all with a regular customer's account. The testing account presents an issue with the elevation of privilege. Even though the UI consists of just a blank screen without any data, we should be able to access the console only as an admin. OWASP describes the vulnerability as the "Security Misconfiguration[25]".

3.2.6 Error Messages

The application has just one general error case for all scenarios, which is misleading. For example, suppose the user enters a valid email with the wrong password during sign-in. In that case, they are presented with an "Invalid email" error message, the same as when they enter a truly invalid email. This scenario may be intended to confuse the malicious user (because the real admin would know that their email is valid). Alternatively, it happens due to the neglect of the web developer who implemented just one general UI flow for handling all kinds of errors.

During the password recovery, the attacker can detect if the email is in the database or not from the presented error message. If the user enters a valid email, they receive a confirmation message from the database. In case the email is invalid, an error message is shown. We can find out from the error message that Blocks use Firebase and that the entered email is not present in the users' database.

3.3 Exploitation

We discovered the system vulnerabilities which could allow the attacker to perform the threat with the highest CVSS score, breaking into the admin account. This section demonstrates testing exploitation as evidence.

[25] `https://owasp.org/Top10/A05_2021-Security_Misconfiguration/`

The admin console misses multi-factor authentication and puts no limit on the failed sign-in attempts, which makes it vulnerable to password brute-force attacks. We simulate the brute-force attack using an open-source tool called Hatch. We assume the attacker already knows the admin's login email. They could find it in numerous ways, such as HTML autocompletion, guessing from error messages, or social engineering.

3.3.1 Hatch

Hatch[26] is an open-source tool intended for cracking passwords on websites by a brute-force attack. We choose Hatch to show how easy it is to perform the attack, even for beginners with little hacking skills. Nonetheless, Hatch has its limitations, and we use it only for display purposes.

Hatch consists of a simple Python script and an attached password list used as a default one if the attacker does not provide his own. The original Hatch tutorial listed on NULL BYTE[27] is intended for Windows users only, but the script can be easily modified for use on other platforms. For using Hatch, the following prerequisites must be met:

- Google Chrome web browser
- ChromeDriver[28]
- Python 2 with imported libraries Selenium and Requests

[26] https://github.com/nsgodshall/Hatch

[27] https://null-byte.wonderhowto.com/how-to/brute-force-nearly-any-website-login-with-hatch-0192225/

[28] https://chromedriver.chromium.org/

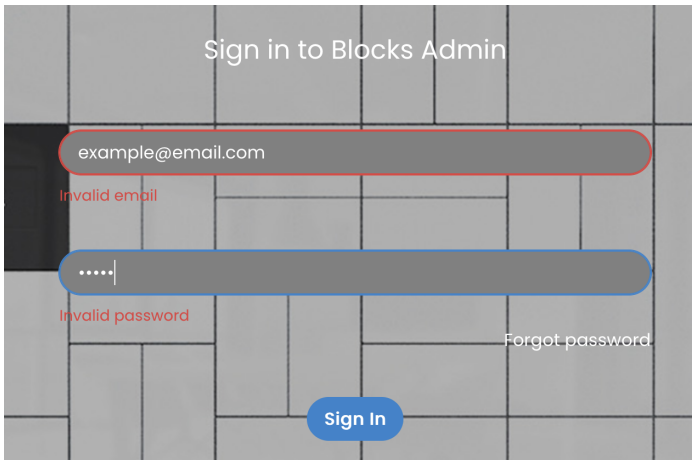

Figure 3.6: Invalid Email Error Message

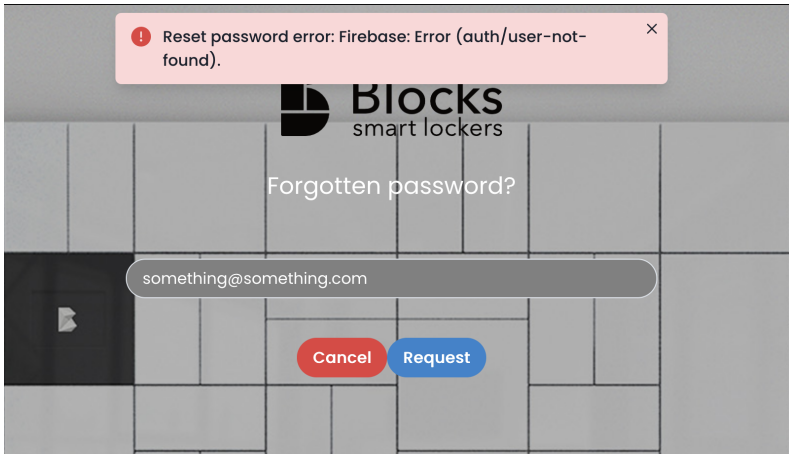

Figure 3.7: Firebase Error Message

Hatch is used from the command line by running the Python 2 script. The user must enter the URL of the target website and the selectors of fields for username, email in our case, password and the sign-in button. How to access the selectors is explained in the online tutorial.

Afterwards, the user chooses if they want to use the default password list or their own and enters the target username (email) for the brute-force attack. The script starts its job and performs the attack.

```
 _     _       _       _
| |   | |     | |     | |
| |__ | | __ _| |_ ___| |__
|  __ |/ _` | __/ __| '_ \
| |  | | (_| | || (__| | | |
|_|  |_|\__,_|\__\___|_| |_|
 [-]--> V.1.0
 [-]--> coded by Metachar
 [-]--> brute-force tool

[~] Enter a website: https://admin.blocks.cz
[!] Checking if site exists [OK]
[~] Enter the username selector: #email
[~] Enter the password selector: #password
[~] Enter the Login button selector: #app > div > div > form > div.css-ftowou > button
[~] Enter the username to brute-force: example@email.com
[~] Enter a directory to a password list: passlist.txt
------------------------
Tried password: 123456
for user: example@email.com
------------------------
------------------------
Tried password: password
for user: example@email.com
------------------------
```

Figure 3.8: Hatch Usage Example

We edit the original script to use the proper ChromeDriver directory by changing the CHROME_DVR_DIR variable value. We also change the website, pass_list, username and selector's variables to the target values, so we do not

need to enter them through the command-line interface.

3.3.2 Brute Force Attack

We found out that the password is successfully broken if it is among the first five entries in the password list. Otherwise, the webpage does not allow the attacker to log in even though the password cracking script entered the correct admin password. The UI continues to show the generic "invalid password/e-mail" error message with every subsequent attempt, and the script keeps running in the background. This behaviour does not respond to the assumption we made in 3.2.4.

The source codes show that the Firebase library function realizes the sign-in authorization logic. Function "signInWithEmailAndPassword()" from the Auth package is used for asynchronous sign-in using an email, and password [35] and fails with an error if the email address and password do not match. In case there are too many invalid login attempts made in a short period, Firebase responds with an auth error code [35].

As the official documentation says, Firebase blocks all requests from the device due to unusual activity, and the user should try it again later [36]. The documentation does not specify how "too many attempts" looks like and what is the acceptable delay between them. By trial and error, we found out that if we set the delay between individual password tries to at least 60 seconds, our script can bypass the Firebase auth error and successfully can continue to bypass the password after five failed attempts.

Nevertheless, as the documentation says, Firebase does not lock the admin account completely, only the specific IP address the requests are coming from. We tried to log in from a different IP address with the correct password, and we succeeded.

3.3.3 Findings

We incorrectly assumed the missing limit on failed login attempts, as we found out from the testing exploitation. There is a limit, which is handled on the backend by the Firebase. If there are too many requests from a single IP address, Firebase temporarily blocks any activity incoming from the device. However, that is the standard behaviour of that particular library function. It does not provide a suitable layer of protection, and the attacker can still bypass the password.

We showed that the main vulnerability of the admin console is missing multiple-factor authentication. In combination with other mentioned vulnerabilities, it can lead to a successful attack and allow the attacker to break into the admin account or, as discussed below in the possible attack scenarios, pursue the DoS of the admin console.

1. **Brute-Force Attack**
   A single attacker (bounded to one specific IP address) is attacking a single account by trying a lot of different passwords. This scenario is limited by the 60 seconds delay between password entries so the particular IP address would not get locked out. It considerably slows down the progress but leads to successful password crack.

2. **Brute-Force Attack & Denial of Service**
   Multiple attackers (with different IP addresses) are attacking a single account by trying a lot of different passwords. It expands the previous scenario by reducing the delay between single password entries. Since the Firebase password lockout is bounded only locally to a particular IP address, the attacker can, during the 60 seconds, try as many passwords as the IP addresses available. The system is vulnerable to the DoS because of the many requests sent to Firebase. Firebase has frequency limitations on performing authorization operations for provided plans [36].

3. **Reverse Brute-Force Attack & Denial of Service**
   Multiple attackers (with different IP addresses) are attacking multiple accounts by trying a single password. This method can either lead to finding a user with a particular password or, more likely, cause a denial of service. The admin console is vulnerable to the DoS attack due to the presented password lockout. The attacker can target several accounts by locking the accounts after five failed login attempts. Due to the misleading error messages, the user would not even know why they cannot log in.

3.4 Evaluation of Results

Based on the gathered knowledge, we conclude that a successful attack requires the attacker to invest effort in preparation and gather non-trivial knowledge about the admin console's implementation. Therefore we reevaluate the CVSS score of the related threats introduced in the threat modelling.

Table 3.1: Reevaluation

Threat	**Reevaluated CVSS Score**
Breaking Into Admin Account	8.1[29]
Reading Data From the Admin Console	6.6[30]

[29] `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H`

[30] `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H`

Changing the Access Rights of the Storage	6.6[31]
Stealing Content of the Lockers	6.6[32]

The security analysis identified potential weak spots and by testing exploitation, we confirmed the vulnerabilities and presented the attack scenarios which the system could face.

Weak Spots

- Format of URL
- HTML Autocomplete Enabled
- Authorization
- Misleading Error Messages

Exploited Vulnerabilites

- One Factor Authentication
- Limit on Failed Login Attempts

3.4.1 Proposed Solutions

This section describes our recommendations for the mitigation of discovered vulnerabilities.

3.4.1.1 Adding Multifactor Authentication

The most significant vulnerability of the admin console is only one authentication layer, the admin password, which enabled us to perform a brute force attack. There are many techniques for preventing brute force attacks. The most obvious one is to add a second authentication factor. Implementing two-factor authentication (2FA) would effectively defend against brute-force attacks and substantially lower the risk of breaking the account. The attacker would have to crack the account password and gain access to the second factor. Blocks already use a phone number verification code through SMS in the mobile client app, which Firebase handles on the backend. Implementing a similar feature to the console would not be difficult and lower the risk significantly. 2FA through SMS has its weak spots and is considered deprecated, but it still provides better protection than the current solution [37].

[31] `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H`

[32] `https://www.first.org/cvss/calculator/3.1#CVSS:3.1/AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:H`

3.4.1.2 Using CAPTCHA

CAPTCHA is a program which can distinguish human from machine input by tests that are easy to complete for humans but difficult for computers [38, 39]. While not wholly stopping them, adding CAPTCHAs provides a layer of protection against automated brute-force attacks.

3.4.1.3 Account Lockouts With Progressive Delays

Locking the account entirely is not a good practice and can lead to DoS attacks. Even though it provides a defence against brute-force password cracking attacks, it is insufficient and does not stop the attacker. However, account lockouts only for a set amount of time still can be a suitable trade-off. It increases the brute-force attack complexity and lowers the risk of DoS since users do not have to unblock their accounts.

We found that the temporary account lockout is implemented on the Firebase side. However, it is not communicated to the user due to misleading error messages, which could be easily fixed by adding a specific error message for this scenario. Blocks already use such message in the mobile app, so they can just propagate it to the admin console [39, 40].

Conclusion

The goal of the thesis was to provide an overview of the smart lockers providers and describe used technologies across competitors with a focus on Blocks lockers. The research was provided together with the description of Blocks' architecture in the first chapter. We found that only a few of several dozen competitors, including Blocks, enable contactless locker access.

The main goal was to perform a security analysis of the Blocks technology. With the help of the STRIDE methodology, we performed a threat analysis and identified threats the system could face. We evaluated their risk score using the CVSS metric, and as a result, we chose to focus on the security evaluation of the admin console. The admin console serves as a management tool and enables remote access to any locker controlled by the admin. By breaking into the admin account, the attacker would gain significant power over the system.

We discovered a few weak spots, most importantly missing multi-factor authentication, which allowed us to successfully perform a brute-force attack and crack the password to the testing admin account.

The security analysis results serve as a background for Blocks developers on what to focus on in improving the software's security measures. We proposed solutions which would lower the risk of identified vulnerabilities. Adding multi-factor authentication would be the most significant improvement and easy to implement. We found out the security measures of the admin console extensively rely on Firebase. Therefore, the final proposal is to implement an additional security layer independently.

We focused only on a portion of the Blocks system solution in detail, and there are still aspects to cover. Future research could, for example, cover the implementation of the Blocks' internal Bluetooth protocol. However, the security analysis resulted in the detection of a major system's vulnerability which would have a high-security impact if exploited. The main goal of the thesis was fulfilled.

Bibliography

[1] Brejlová, I. *Chytré výdejní boxy se nenápadně rozšířily po Česku, obsadily Slovensko a vyrostly o 1 500 procent.* In: *CzechCrunch* [online], April 2022, [accessed 2022-04-23]. Available from: `https://cc.cz/chytre-vydejni-boxy-se-nenapadne-rozsirily-po-cesku-obsadily-slovensko-a-vyrostly-o-1-500-procent/`

[2] Irwin, L. *Risk terminology: Understanding assets, threats and vulnerabilities.* In: *Vigilant Software - Compliance Software Blog* [online], July 2020, [accessed 2022-04-28]. Available from: `https://www.vigilantsoftware.co.uk/blog/risk-terminology-understanding-assets-threats-and-vulnerabilities`

[3] Stango, A.; Prasad, N. R.; et al. *A Threat Analysis Methodology for Security Evaluation and Enhancement Planning.* In: *The Third International Conference on Emerging Security Information, Systems and Technologies* [online], June 2009: pp. 262–267, ISSN 2162-2116, doi:10.1109/SECURWARE.2009.47. [accessed 2022-04-16]. Available from: `https://ieeexplore.ieee.org/document/5210987`

[4] NICCS. *Risk Assessment.* In: *Cybersecurity Glossary* [online]. March 2022, [accessed 2022-05-05]. Available from: `https://niccs.cisa.gov/about-niccs/cybersecurity-glossary`

[5] Editor, CSRC Content. *Risk.* In: *Glossary CSRC* [online]. [accessed 2022-05-05]. Available from: `https://csrc.nist.gov/glossary/term/risk`

[6] Hogganvik, I.; Stolen, K. *Risk analysis terminology for IT-systems: does it match intuition?* In: *2005 International Symposium on Empirical Software Engineering, 2005.* [online], November 2005: pp. 10 pp.–, doi:10.1109/ISESE.2005.1541810. [accessed 2022-05-18]. Available from: `https://ieeexplore.ieee.org/document/1541810`

[7] NICCS. *Risk.* In: *Cybersecurity Glossary* [online]. March 2022, [accessed 2022-05-05]. Available from: https://niccs.cisa.gov/about-niccs/cybersecurity-glossary

[8] Geib, J. *Threats - Microsoft Threat Modeling Tool.* In: *Azure* [online], March 2022, [accessed 2022-05-06]. Available from: https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats

[9] Microsoft. *Applying STRIDE.* In: *Microsoft Build* [online], November 2009, [accessed 2022-05-06]. Available from: https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee798544(v=cs.20)

[10] Shostack, A. *Threat Modeling: Designing for Security.* New York, UNITED STATES: John Wiley & Sons, Incorporated, 2014, ISBN 978-1-118-82269-2, [accessed 2022-05-05]. Available from: http://ebookcentral.proquest.com/lib/techlib-ebooks/detail.action?docID=1629177

[11] FIRST.Org, Inc. *CVSS v3.1 Specification Document* [online]. [accessed 2022-05-11]. Available from: https://www.first.org/cvss/specification-document

[12] Pokorný, Filip. *Modelování rizik zranitelností s CVSS.* In: *SystemOnline* [online], April 2019, ISSN 1802-615X. [accessed 2022-05-06]. Available from: https://www.systemonline.cz/it-security/modelovani-rizik-zranitelnosti-s-cvss.htm

[13] IBM. *Common Vulnerability Scoring System* [online]. November 2021, [accessed 2022-05-13]. Available from: https://prod.ibmdocs-production-dal-6099123ce774e592a519d7c33db8265e-0000.us-south.containers.appdomain.cloud/docs/en/qradar-on-cloud?topic=vulnerabilities-common-vulnerability-scoring-system-cvss

[14] Balbix. *CVSS Base Score Explained* [online]. April 2020, [accessed 2022-05-13]. Available from: https://www.balbix.com/insights/base-cvss-scores/

[15] Forcepoint. *What is Spoofing?* [online]. August 2018, [accessed 2022-05-13]. Available from: https://www.forcepoint.com/cyber-edu/spoofing

[16] NICCS. *Authentication.* In: *Cybersecurity Glossary* [online]. March 2022, [accessed 2022-05-05]. Available from: https://niccs.cisa.gov/about-niccs/cybersecurity-glossary

[17] Editor, CSRC Content. *Tampering.* In: *Glossary CSRC* [online]. [accessed 2022-05-13]. Available from: https://csrc.nist.gov/glossary/term/tampering

[18] Editor, CSRC Content. *Integrity.* In: *Glossary CSRC* [online]. [accessed 2022-05-13]. Available from: https://csrc.nist.gov/glossary/term/integrity

[19] NICCS. *Non-repudiation.* In: *Cybersecurity Glossary* [online]. March 2022, [accessed 2022-05-05]. Available from: https://niccs.cisa.gov/about-niccs/cybersecurity-glossary

[20] Editor. *What is a denial of service attack (DoS)?*, [online]. [accessed 2022-05-13]. Available from: https://www.paloaltonetworks.com/cyberpedia/what-is-a-denial-of-service-attack-dos

[21] Hollasch, L. *Elevation of Privilege - Windows drivers* [online]. [accessed 2022-05-13]. Available from: https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/elevation-of-privilege

[22] NICCS. *Authorization.* In: *Cybersecurity Glossary* [online]. March 2022, [accessed 2022-05-05]. Available from: https://niccs.cisa.gov/about-niccs/cybersecurity-glossary

[23] Wichers, D. *Source Code Analysis Tools.* In: *OWASP Foundation* [online]. [accessed 2022-05-16]. Available from: https://owasp.org/www-community/Source_Code_Analysis_Tools

[24] Zalecki, M. *How to use static code analysis to write quality JavaScript/TypeScript* [online]. June 2020, [accessed 2022-05-16]. Available from: https://blog.logrocket.com/how-to-use-static-code-analysis-to-write-quality-javascript-typescript/

[25] Springett, S. *Component Analysis* In: *OWASP Foundation* [online]. [accessed 2022-05-17]. Available from: https://owasp.org/www-community/Component_Analysis

[26] Grozev, N. *Improve Your TypeScript With Static Analysis* [online]. March 2020, [accessed 2022-05-17]. Available from: https://nikgrozev.com/2020/03/22/improve-your-typescript-with-static-analysis/

[27] OWASP. *The Ten Most Critical Web Application Security Vulnerabilities.* [accessed 2022-05-17]. Available from: https://owasp.org/www-pdf-archive/OWASP_Top_10_2007.pdf

[28] VERACODE. *Failure to Restrict URL Access.* In: *Veracode* [online], [accessed 2022-05-16]. Available from: https://www.veracode.com/security/failure-restrict-url-access

[29] Zahradnický, T.; Kokeš, J. *Bezpečný kód - 1. Introduction to Secure Code, Thread Modeling* [Lecture]. Prague: Faculty of Information Technology, CTU in Prague, February 2022. [accessed 2022-05-19]. Available from: https://courses.fit.cvut.cz/BI-BEK/media/lectures/bek01en.pdf

[30] Yalcin, M. *What Impact Does the Autocomplete Feature Have on Web Security?* [online]. March 2019, [accessed 2022-05-15]. Available from: https://www.invicti.com/blog/web-security/impact-autocomplete-feature-web-security/

[31] Mozzila. *Autocomplete*. In: *mdn web docs* [online], [accessed 2022-05-16]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/autocomplete

[32] Mozzila. *How to turn off form autocompletion* [online]. In: *mdn web docs* [online], [accessed 2022-05-17]. Available from: https://developer.mozilla.org/en-US/docs/Web/Security/Securing_your_site/Turning_off_form_autocompletion

[33] Mozzila. *The Input element*. In: *mdn web docs* [online], [accessed 2022-05-16]. Available from: https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input

[34] TechWeb. *Why Use 2FA?* [online]. [accessed 2022-05-15]. Available from: https://www.bu.edu/tech/support/information-security/why-use-2fa/

[35] Firebase. *Auth package* [online]. [accessed 2022-05-16]. Available from: https://firebase.google.com/docs/reference/js/auth.md#auth_package

[36] Firebase. *Authentication Limits* [online]. [accessed 2022-05-18]. Available from: https://firebase.google.com/docs/auth/limits

[37] Grassi, P. A.; Garcia, M. E.; et al. *Digital identity guidelines: revision 3*. Technical report NIST SP 800-63-3, National Institute of Standards and Technology, Gaithersburg, MD, June 2017, doi:10.6028/NIST.SP.800-63-3. Available from: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf

[38] LEXICO. *CAPTCHA*. In: *Lexico* [online]. [accessed 2022-05-16]. Available from: https://www.lexico.com/definition/captcha

[39] Wichers, D. *Blocking Brute Force Attacks Control*. In: *OWASP Foundation* [online], [accessed 2022-05-18]. Available from: https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks

[40] PhoenixNAP. *How To Prevent Brute Force Attacks With 8 Easy Tactics* [online]. December 2018, [accessed 2022-05-16]. Available from: `https://phoenixnap.com/kb/prevent-brute-force-attacks`

Appendix A

Acronyms

BLE Bluetooth Low Energy

CAPTCHA Completely Automated Public Turing test to tell Computers and Humans Apart

CVSS Common Vulnerability Scoring System

DoS Denial of Serice

HTML Hypertext Markup Language

IEEE Institute of Electrical and Electronics Engineers

ISM Band Industrial, Scientific, and Medical Radio Band

MDM Mobile Device Management

NFC Near Field Communication

OWASP Open Web Application Security Project

QR Code Quick Response code

SAST Static Application Security Testing

STRIDE Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege

UI User Interface

URL Uniform Resource Locator

Appendix B

Smart Lockers Providers

The full list of smart lockers providers in the Czech Republic.

Table B.1: Smart Lockers Providers

Company Name	**Use Case**	**Ways of Access**
Blocks Lockers	storage, packages	BLE, keypad, scanner, NFC
AlzaBox	packages	keypad, scanner
PilulkaBox	packages	BLE
Z-BOX	packages	BLE
DPD Box	packages	keypad, scanner
PPL Parcelbox	packages	keypad, scanner
Rohlík Point	packages	keypad
GLS Parcel Locker	packages	keypad
Mall Box	packages	keypad
OX Box	packages	keypad, NFC
DistriBox	packages	keypad, scanner
FurtodoBox	packages	keypad, scanner
GO-BOX	packages, storage	code lock
ParcelBox	packages	keypad, scanner
PostCube	packages	BLE
WE|DO BOX	packages	keypad
KePol Parcel lockers	packages	keypad, scanner
Bringme Box	packages	keypad, scanner
SmartPoint	packages	keypad, scanner
MyPup Pick Up Point	packages	keypad
Simplicity Lockers	storage	-

Table B.2: Providers' websites

Company Name	**Website**
Blocks Lockers	`https://blockslockers.com/`

AlzaBox	https://www.alza.cz/alzabox?alzaboxy-pro-zakazniky
PilulkaBox	https://www.pilulka.cz/pilulka-box
Z-BOX	https://www.zasilkovna.cz/zbox
DPD Box	https://www.dpd.com/cz/en/receiving-parcels/dpd-boxes/
PPL Parcelbox	https://www.ppl.cz/cs/-/spusteni-parcelboxu
Rohlík Point	https://www.rohlik.cz/tema/rohlikpoint
GLS Parcel Locker	https://gls-group.eu/SK/en/services-for-senders/occasional-shipping/parcel-locker
Mall Box	https://www.mall.cz/mallbox
OX Box	https://cp4u.cz/
DistriBox	https://www.distribox.cz/
FurtodoBox	https://www.furtodo.com/
GO-BOX	https://en.go-box.cz/
ParcelBox	https://parcelboxy.cz/
PostCube	http://postcube.cz/
WE|DO BOX	https://www.wedo.cz/we-do-box
KePol Parcel lockers	https://www.keba.com/en/logistics-solutions/parcel-locker-portfolio/parcel-locker
Bringme Box	https://www.bringme.com/office/virtual-reception/bringme-box
SmartPoint	https://smartpoint.ai/en/portal-servicios
MyPup Pick Up Point	https://my-pup.com/our-offer/pick-up-points/
Simplicity Lockers	https://simplicitylockers.co.uk/products/convergence/

Appendix C

Additional Images

Figure C.1: Opened Blocks Locker

Figure C.2: iPad serving as a touchscreen

Appendix D

Error Messages

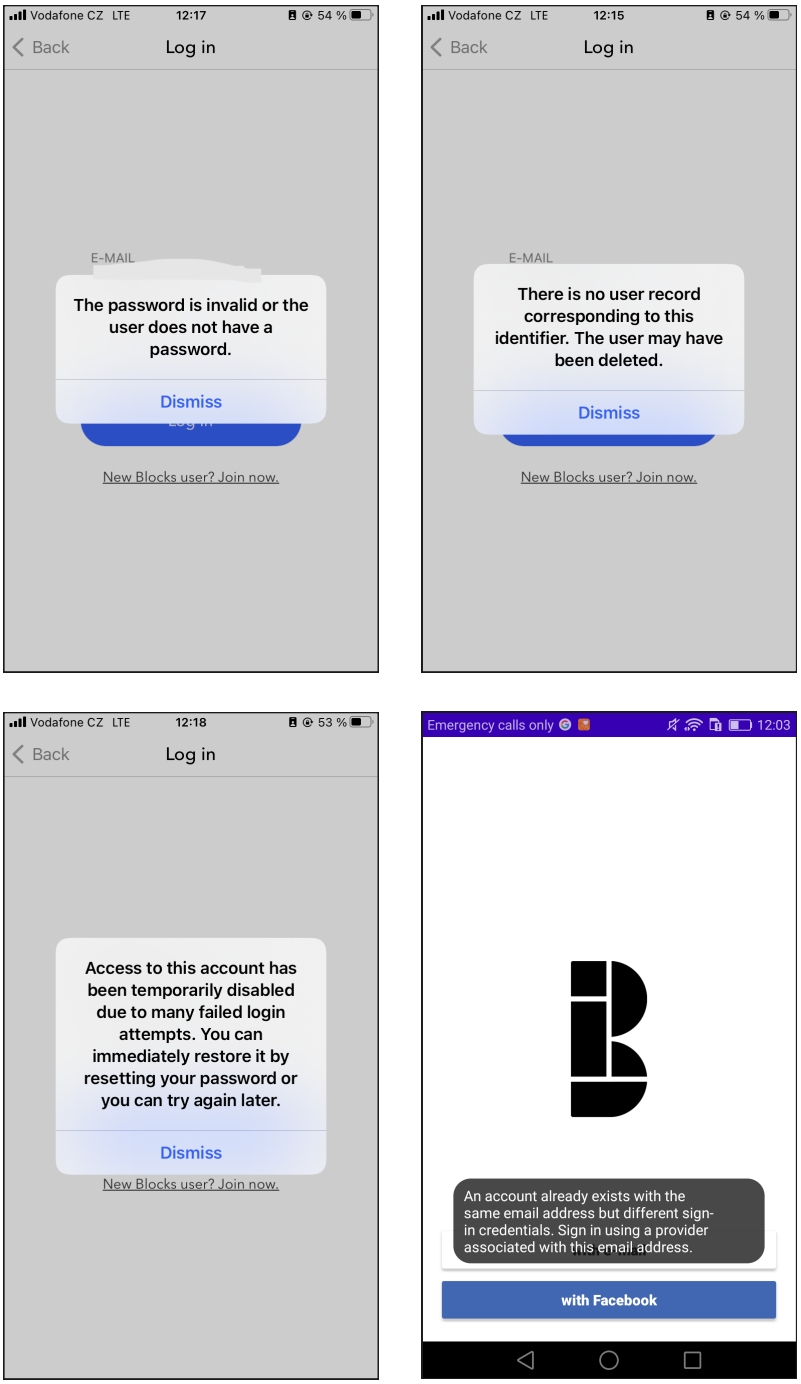

Figure D.1: Error Messages During the Login Process

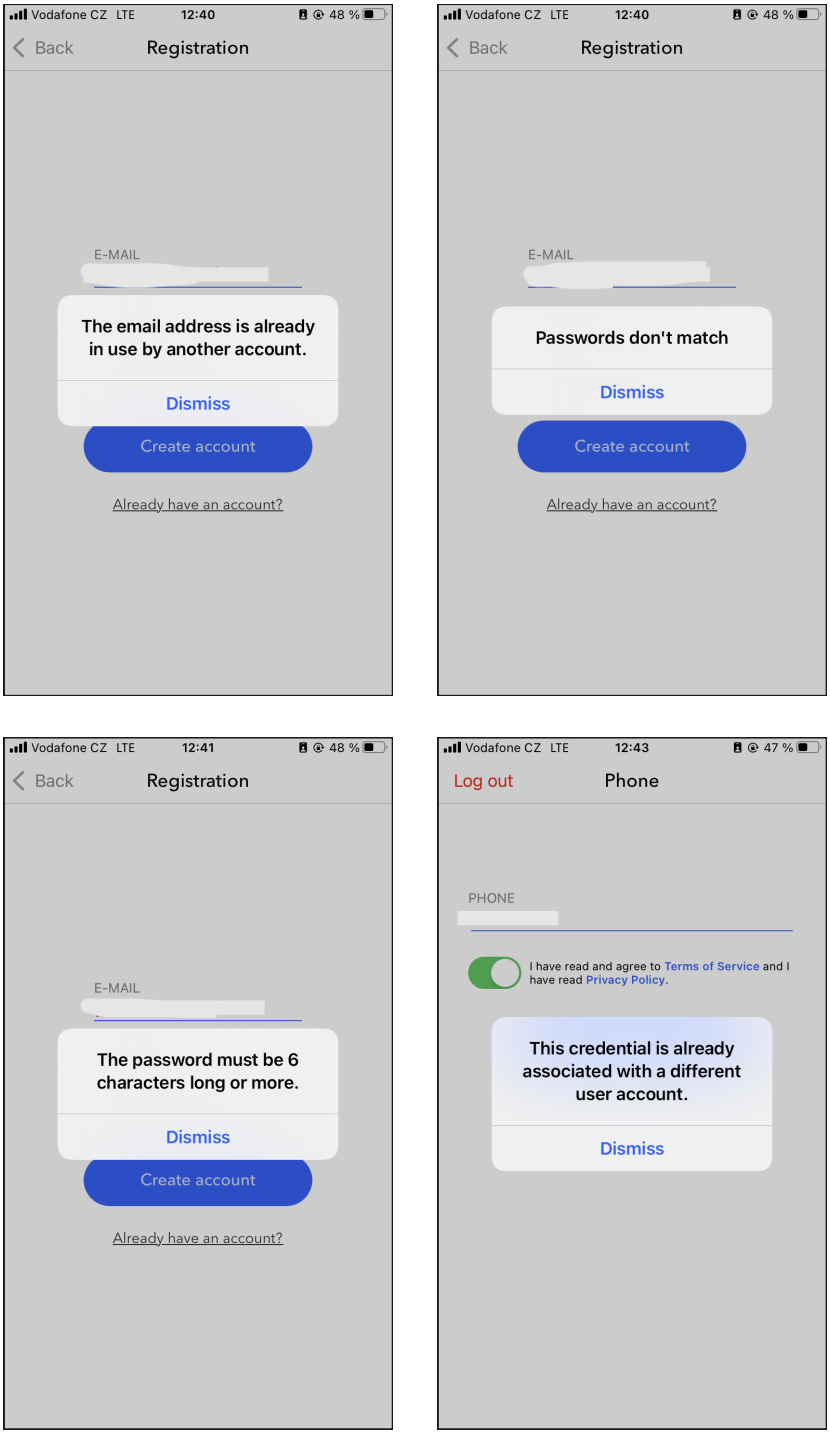

Figure D.2: Error Messages During the Registration Process

Appendix E

Contents of Attached Media

- `src` ..... the directory of source codes
  - `main.py` ..... edited Hatch script
  - `thesis` ..... the directory of LaTeX source codes of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format