

SECURITY ANALYSIS OF THE PAYLEVEN MOBILE POINT-OF-SALE PLATFORM

Radboud University Nijmegen



SAFET ACIFOVIC

SECURITY ANALYSIS OF THE PAYLEVEN MOBILE POINT-OF-SALE PLATFORM

by

SAFET ACIFOVIC

in partial fulfillment of the requirements for the degree of

Master of Science
in Computing Science

at the Radboud University,
to be defended publicly on February 19, 2016 at 11:00.

| | | |
|-------------|----------------------|-----------------------------|
| Supervisor: | dr. ir. E. Poll | Radboud University Nijmegen |
| Supervisor: | Prof. dr. E. Verheul | Radboud University Nijmegen |
| Supervisor: | V. de Vries | Fox-IT |

An electronic version of this thesis is available at
<http://ru.nl/icis/education/master-thesis/theses-archive/>.

Radboud Universiteit



PREFACE

This thesis is submitted for the degree of Master of Science in Computing Science at the Radboud University Nijmegen. The research was conducted under supervision of dr. ir. E. Poll and Prof. dr. E. Verheul in the department of the Digital Security group, Radboud University Nijmegen, between February 2015 and February 2016.

I would like to thank my supervisors, dr. ir. E. Poll and Prof. dr. E. Verheul for their assistance, ideas, knowledge and time. Also, I would like to thank Fox-IT and V. de Vries for the provision of their facilities, devices and equipment.

Safet Acifovic
Nijmegen, February 2016

ABSTRACT

This thesis experimentally evaluates the security of the Payleven smartphone-based point-of-sale (POS) system which consists of a mobile app accompanied by a small Bluetooth-based Chip & PIN card reader. The wireless and smartphone-based nature of the Payleven platform prompts the question on how secure it is. The fact that the payment platform relies on the security of smartphones is a cause of concern, especially when financial transactions are involved.

The initial part of the research was aimed at understanding the internal workings of the payment platform. The entire data flow of the platform was set out by intercepting all Internet traffic and all Bluetooth traffic. This way it was possible to map the data exchange process between the several components of the payment system and get an insight into its strengths and weaknesses.

Ultimately, the research led to a good understanding of the system. It can be stated that no major security flaws could be identified nor could an attack be demonstrated on the system. However, there is room for improvement in terms of security. One of the findings that stood out was the involvement of another party within the system. The actual processing of the payments was not done by Payleven, but by the payment service provider Adyen. It was even found that the whole system was developed by Adyen. But the fact that strikes the most and introduces the real issue is the lack of connection and cooperation between these two parties. The link between Payleven and Adyen is non-existent throughout the payment platform; that is to say, it is theoretically possible to alter Adyen related account information without affecting the Payleven instances in such way that monetary transactions might be diverted to an adversary. All this without Payleven or Adyen noticing the discrepancy.

The other issues identified were due to improper security implementations; for example, the platform lacks proper session expiration management and anti brute-force mechanisms. Since Payleven is listed as a payment institution, the lack of extended validated SSL certificates is a serious matter as it makes phishing attacks easier. Moreover, it was possible to extract the reader's firmware and gain knowledge regarding its logic. The firmware also exposed several access codes for the card reader's admin menu.

CONTENTS

| | |
|--|-----------|
| ABSTRACT | x |
| 1 INTRODUCTION | 1 |
| 1.1 MOTIVATION | 2 |
| 1.2 RESEARCH GOAL | 2 |
| 1.3 BACKGROUND & RELATED WORK | 3 |
| 1.4 SCOPE | 4 |
| 1.5 OUTLINE | 4 |
| 2 PAYLEVEN | 5 |
| 2.1 THE PAYLEVEN PAYMENT PLATFORM | 5 |
| 2.1.1 HOW DOES IT WORK? | 6 |
| 2.1.2 BUSINESS MODEL | 6 |
| 2.1.3 ADYEN: THE PAYMENT SERVICE PROVIDER | 7 |
| 2.1.4 SCHEMATIC OVERVIEW OF THE PAYMENT PLATFORM | 8 |
| 2.1.5 CERTIFICATION | 9 |
| 2.2 THE COMPONENTS OF THE PLATFORM | 10 |
| 2.2.1 CARD READER SPECIFICATIONS | 10 |
| 2.2.2 PERSONAL DASHBOARD | 11 |
| 2.2.3 PAYLEVEN PAYMENT APP | 13 |
| 2.3 USE CASES | 13 |
| 2.3.1 REGISTRATION WITH PAYLEVEN | 14 |
| 2.3.2 APP LOGIN AND INITIALIZATION | 15 |
| 2.3.3 BLUETOOTH PAIRING PROCESS | 16 |
| 2.3.4 ACCEPTING PAYMENTS | 18 |
| 2.3.5 REFUNDS | 21 |
| 2.4 HIGH-LEVEL MESSAGE SEQUENCE CHARTS OF THE VARIOUS USE CASES | 22 |
| 2.4.1 MSC: APP LOGIN & INITIALIZATION | 22 |
| 2.4.2 MSC: BLUETOOTH PAIRING PROCESS | 22 |
| 2.4.3 MSC: TRANSACTION PROCESS | 23 |
| 2.4.4 MSC: REFUND PROCESS | 24 |
| 3 METHODOLOGY | 25 |
| 3.1 RESEARCH APPROACH | 25 |
| 3.2 MATERIALS & SETUP | 26 |
| 3.3 METHODS & TOOLS | 27 |
| 3.3.1 INTERCEPTING THE BLUETOOTH TRAFFIC | 28 |
| 3.3.2 ANALYZING THE NETWORK TRAFFIC | 29 |
| 3.3.3 REVERSE ENGINEERING THE PAYLEVEN APP | 31 |

| | | |
|--------|---|----|
| 3.3.4 | ANALYZING THE CARD READER | 33 |
| 3.4 | THREAT MODEL | 34 |
| 3.4.1 | SECURITY REQUIREMENTS | 34 |
| 3.4.2 | ASSUMPTIONS | 34 |
| 3.4.3 | ATTACK POINTS | 34 |
| 3.4.4 | ATTACK VECTORS | 35 |
| 3.4.5 | ASSETS | 37 |
| 3.4.6 | ATTACKER TYPE | 38 |
| 3.4.7 | THREAT CATEGORIZATION & ATTACKER GOAL | 39 |
| 3.4.8 | DATA FLOW DIAGRAM | 41 |
| 3.4.9 | POTENTIAL THREATS | 42 |
| 3.4.10 | POTENTIAL ATTACK SCENARIOS | 45 |
| 4 | INTERNET NETWORK ANALYSIS | 51 |
| 4.1 | GLOBAL OVERVIEW OF THE NETWORK TRAFFIC | 51 |
| 4.2 | NETWORK TRAFFIC DURING APP LOGIN | 55 |
| 4.2.1 | WEB REQUEST P1A | 56 |
| 4.2.2 | WEB REQUEST P2A | 57 |
| 4.2.3 | WEB REQUEST P2B | 58 |
| 4.2.4 | LOGS 1 | 58 |
| 4.3 | NETWORK TRAFFIC DURING APP INITIALIZATION | 59 |
| 4.3.1 | WEB REQUEST A1A | 59 |
| 4.3.2 | LOG 2 | 61 |
| 4.4 | NETWORK TRAFFIC DURING BLUETOOTH PAIRING PROCESS | 62 |
| 4.4.1 | WEB REQUEST P3A | 62 |
| 4.4.2 | WEB REQUEST A2A | 63 |
| 4.4.3 | WEB REQUEST A2B | 65 |
| 4.4.4 | LOG 3 | 67 |
| 4.4.5 | WEB REQUEST A2C | 68 |
| 4.5 | NETWORK TRAFFIC DURING THE TRANSACTION PROCESS | 69 |
| 4.5.1 | WEB REQUEST P4A | 69 |
| 4.5.2 | LOG 4 | 71 |
| 4.5.3 | WEB REQUEST A3A & A4A (ENCRYPTED) | 72 |
| 4.5.4 | WEB REQUEST P5A | 72 |
| 4.5.5 | LOG 5 | 75 |
| 4.5.6 | WEB REQUEST A4B (ENCRYPTED) | 76 |
| 4.6 | NETWORK TRAFFIC DURING THE REFUND PROCESS | 76 |
| 4.6.1 | WEB REQUEST A5A | 76 |
| 5 | BLUETOOTH TRAFFIC ANALYSIS | 79 |
| 5.1 | BLUETOOTH TRAFFIC DURING THE PAIRING PROCESS | 79 |
| 5.1.1 | BLUETOOTH MESSAGE 1 | 80 |
| 5.1.2 | BLUETOOTH MESSAGE 2 | 81 |
| 5.1.3 | BLUETOOTH MESSAGE 3 | 82 |
| 5.1.4 | BLUETOOTH MESSAGE 4 | 82 |
| 5.1.5 | BLUETOOTH MESSAGE 5 | 83 |

| | | |
|--------|---|-----|
| 5.1.6 | BLUETOOTH MESSAGE 6 | 83 |
| 5.1.7 | BLUETOOTH MESSAGE 7 | 84 |
| 5.2 | BLUETOOTH TRAFFIC DURING THE TRANSACTION | 85 |
| 5.2.1 | BLUETOOTH MESSAGE 8 | 86 |
| 5.2.2 | BLUETOOTH MESSAGE 9 | 86 |
| 5.2.3 | BLUETOOTH MESSAGE 10. | 87 |
| 5.2.4 | BLUETOOTH MESSAGES 11 TO 14 | 87 |
| 5.2.5 | BLUETOOTH MESSAGE 15. | 88 |
| 5.2.6 | ENCRYPTED DATA EXCHANGE (1). | 89 |
| 5.2.7 | BLUETOOTH MESSAGE 16. | 89 |
| 5.2.8 | BLUETOOTH MESSAGE 17 & 18 | 89 |
| 5.2.9 | BLUETOOTH MESSAGE 19, 20, 21 & 22 | 91 |
| 5.2.10 | ENCRYPTED DATA EXCHANGE (2). | 92 |
| 5.2.11 | BLUETOOTH MESSAGE 23. | 92 |
| 5.2.12 | BLUETOOTH MESSAGE 24, 25 & ENCRYPTED DATA EXCHANGE (3). | 95 |
| 5.2.13 | BLUETOOTH MESSAGE 26, 27 & ENCRYPTED DATA EXCHANGE (4, 5) | 96 |
| 6 | SOFTWARE & HARDWARE ANALYSIS | 97 |
| 6.1 | ANALYZING THE APP'S SOURCE CODE | 97 |
| 6.1.1 | PROGRAM UNDERSTANDING | 98 |
| 6.1.2 | CLIENT-SIDE ENCRYPTION KEY | 99 |
| 6.2 | ANALYZING THE CHIP & PIN CARD READER | 100 |
| 6.2.1 | DISSEMBLING THE CARD READER | 100 |
| 6.2.2 | EXTRACTING THE FILE SYSTEM | 102 |
| 6.2.3 | COMMENTS REGARDING THE FIRMWARE | 104 |
| 6.2.4 | ACCESSING THE HIDDEN ADMIN MENU | 105 |
| 7 | RESULTS | 109 |
| 7.1 | SCENARIO 1: ALTERING THE ADYEN MERCHANT ACCOUNT. | 109 |
| 7.1.1 | THE ISSUE CONCERNING THE ADYEN MERCHANT ACCOUNT | 109 |
| 7.1.2 | ATTACK SCENARIO: DIVERTING TRANSACTIONS. | 111 |
| 7.1.3 | TRACEABILITY OF THE ATTACK | 112 |
| 7.1.4 | FEASIBILITY OF THE ATTACK | 113 |
| 7.1.5 | THE IDEAL APPROACH | 115 |
| 7.1.6 | EXAMINATION OF THE ATTACK | 115 |
| 7.2 | SCENARIO 2: ALTERING TRANSACTION INFORMATION. | 116 |
| 7.2.1 | ALTERING THE TRANSACTION AMOUNT. | 116 |
| 7.2.2 | REFUNDING ALTERED TRANSACTIONS. | 116 |
| 7.2.3 | FAKING TRANSACTIONS TOWARDS PAYLEVEN. | 117 |
| 7.2.4 | CASH BACKS WITH FAKE PAYMENT RECEIPTS | 117 |
| 7.2.5 | BYPASSING LOCATION RESTRICTIONS | 118 |

| | | |
|-------|--|-----|
| 7.3 | SCENARIO 3: TRIGGERING ILLEGITIMATE REFUNDS | 119 |
| 7.3.1 | BRUTE-FORCING THE REFUND AUTHORIZATION PIN | 119 |
| 7.3.2 | IMPROPER SESSION EXPIRATION MANAGEMENT | 120 |
| 7.3.3 | RETRIEVING THE LOGIN CREDENTIALS | 121 |
| 7.3.4 | THE LACK OF EXTENDED VALIDATED SSL CERTIFICATES | 121 |
| 8 | CONCLUSION | 123 |
| 8.1 | MAIN FINDINGS | 123 |
| 8.2 | CONCLUDING REMARKS & RECOMMENDATIONS | 125 |
| 8.3 | FUTURE WORK | 126 |
| | BIBLIOGRAPHY | 127 |

1

INTRODUCTION

The last several years have seen a growing trend towards the use of smartphones as a platform replacing dedicated hardware. Smartphones are, for example, steadily supplanting dedicated handheld barcode scanners in retail and logistics. In combination with other hardware, smartphones can even replace specialized tools such as light and wind meters.¹ The versatility and ubiquity of smartphones has led to a shift in focus from expensive dedicated hardware devices to cheaper smartphone-based solutions.

However, this trend has implications in terms of security, privacy and reliability. The overall security of smartphone-based applications is highly dependent on the security of smartphones. In contrast to dedicated hardware, smartphones form an accessible and widespread platform which makes them a popular target for malicious attackers. Specifically security sensitive smartphone-based point-of-sale (POS) systems are of great interest for attackers. A POS terminal is an electronic device responsible for the processing of card payments in order to facilitate the selling of goods and services to consumers.

In 2012, researchers of the University of Wisconsin-Madison performed a security audit on a collection of smartphone-based POS systems [1]. The POS systems consisted of an inexpensive magnetic stripe reader plugged into the audio-jack of the smartphone and an accompanying payment application installed on the smartphone. The smartphone acted as a POS terminal. Their research exposed several software vulnerabilities in the firmware that could be exploited by malicious mobile applications.

This research sets out to evaluate the security of a comparable but more sophisticated smartphone-based POS system: the Payleven payment platform.² The difference lies in the form factor and approach. Instead of a audio-jack dongle, the Payleven payment platform includes a small wireless mobile PIN terminal with a

¹Light meter: <http://lumu.eu> & wind meter: <http://vaavud.com>

²<https://payleven.co.uk>

smart card reader, number pad and small OLED screen. The most prominent feature is the fact that the mobile terminal uses Bluetooth to communicate with the associated payment app on the smartphone. The smartphone fulfills primarily an intermediary function between the mobile terminal and the Payleven back-end, whereas the dedicated mobile terminal processes the actual card payments.

The wireless nature of the terminal brings up questions about the security and reliability of the Payleven platform. Wireless communication introduces new attack vectors. Moreover, the fact that the smartphone serves as a relay for confidential information is a cause for concern. Such an approach demands additional security requirements, especially when financial transactions are involved.

1.1. MOTIVATION

It is of particular interest to determine whether appropriate security measures are taken to ensure the security and reliability of the Payleven payment platform. Insecure payment systems can cause financial damage. Additionally, smartphone-based payment systems attract due to their characteristics, even more the attention of both security experts and computer criminals.

In the long run, the development of mobile payment systems will rise with the shift to a more smartphone-centric world. Because alternative form factors will emerge and introduce new attack vectors, mobile payment vendors need to be prepared and well-informed as regards the security risks that smartphone-based solutions entail. Therefore, an extensive security assessment is necessary to bring forward new insights into the security of such payment systems. Moreover, up to now, far too little attention has been paid to academic security research of mobile payment systems.

1.2. RESEARCH GOAL

This research evaluates the security of the Payleven payment platform. In order to facilitate the security evaluation, several aspects of the platform will be highlighted in this research. Both the Bluetooth and wireless network connection will be inspected. The data flow on these connections will be set out and examined. Furthermore, the Payleven app and the card reader will also be subjected to research. Finally, the internal workings of the Payleven payment platform will become clear and with this a better picture of the system's security will be reached.

Therefore, the goal of this research is to answer the following research questions:

- Q1: How does the Payleven payment platform works in terms of data flow and (inter)dependencies between the different components of the system, and more importantly, in terms of security?*
- Q2: Based on the insights on the internal workings of the Payleven payment platform (i.e., first research question), to what extent is*

the payment platform susceptible for malicious attacks targeting the merchant, the customer or Payleven itself with the aim of causing monetary or reputation damage, information disclosure or denial-of-service to the platform?

1.3. BACKGROUND & RELATED WORK

In [1] the researcher performed a security analysis on several smartphone mobile POS systems. The systems consisted of a audio-jack magnetic stripe reader that communicates with the associated mobile app. Their main finding is that an arbitrary application running on the smartphone can permanently disable the magnetic stripe reader and extract secret cryptographic keys. It was even possible to gain privileged access needed to upload new firmware. These systems are similar to the mobile POS system designed by Payleven, but less sophisticated.

At the Black Hat 2014 information security conference in Las Vegas two researchers from MWR Labs demonstrated an attack on a mobile card reader [2] [3]. They were able to upload Flappy Bird on the reader by inserting an infected smartcard. The affected card reader is a Miura Shuttle manufactured by Miura Systems³; the same device used by the Payleven payment platform. However, since the attack dates back to mid 2014, the possibility exists that the vulnerability is already fixed through a firmware update - assuming that a weakness in the firmware was responsible. Unfortunately, little is known about the attack. MWR Labs did not provide any details about the attack.

The Payleven payment platform makes use of EMV. EMV is a global technical standard for payment cards based on chip technology and is developed by Europay, Mastercard and Visa (or: EMV). EMV transactions are often referred to with the marketing term 'Chip & PIN' since a PIN entry is required to verify whether the customer is also the card holder. EMV also supports other card holder verification methods (CVM) such as 'Signature' or 'No CVM required'. In addition, the PIN may also be verified online (i.e., 'Online PIN') by the bank or verified offline (i.e., 'Offline enciphered PIN' or 'Offline plaintext' PIN) by the chip on the banking card. For more information regarding EMV please consult the latest specifications [4]. This research will not address EMV except for the CVM as it says something about the internal workings of the payment platform.

In [5] an attack on EMV is described. The attack exploits the fact that some EMV implementations lack secure counters, timestamps or other algorithms to supply the unique number used to ensure the freshness of each transaction. During this attack Bond et al. also exposed a flaw in the EMV protocol; the random number generated by the card reader can easily be replaced by one the attacker used before when capturing an authentication code from the smartcard.

³<http://www.miurasystems.com>

1.4. SCOPE

Although the Payleven payment platform primarily processes EMV transactions and EMV has shown in the past to be vulnerable [5][6][7][8], this thesis will not research the specific EMV implementation in the Payleven payment platform.

Further, the aim of this research is not to break the Bluetooth encryption or the wireless network connection. However, it will be tried intercept Bluetooth traffic as the wireless Bluetooth connection forms an interesting attack point.

1.5. OUTLINE

Chapter 2 describes the Payleven payment platform; the several functionalities and components are addressed, and both the registration process and the various use cases are elaborated. Further, with each use case a high-level message sequence chart is given.

Chapter 3 consists of two parts. First, the methodology is given; the methods used to conduct the research and analyze the results are listed in Section 3.1. The second part focuses on the threat landscape of the Payleven payment platform. It discusses the platform's attack points and potential threats. The chapter is completed with several potential attack scenarios.

Chapter 4 sets out the Internet network traffic of the Payleven payment platform during the various use cases. The following chapter, Chapter 5, focuses on the data exchanged back and forth on the Bluetooth connection.

Chapter 6 briefly elaborates on the findings of the analysis on the Payleven app and the card reader's firmware. It also examines the hardware of the card reader.

Chapter 7 highlights a number of issues and shortcomings that have been identified in the Payleven payment platform. Possible attack scenarios based on these issues are outlined.

Finally, Chapter 8 concludes the research by restating the most important findings. It also gives recommendations on future research.

2

PAYLEVEN

This chapter introduces the reader to the Payleven payment platform. Section 2.1 describes the functionalities of the Payleven payment platform and gives a schematic overview of the different components of the platform and their interaction. Section 2.2 further examines these components and Section 2.3 elaborates on the different use cases of the Payleven payment platform. Lastly, Section 2.4 visualizes through Message Sequence Charts (MSCs) the interaction among the different components and actors of the Payleven platform during the different use cases.

2.1. THE PAYLEVEN PAYMENT PLATFORM

The Payleven payment platform is a mobile card payment system that consists of:

- a smartphone payment app;
- a merchant service area (personal dashboard);
- and a wireless Chip & PIN (EMV) card reader.

The Chip & PIN card reader is a separate mobile device that connects to the smartphone via Bluetooth technology and is managed through the Payleven payment app. The Payleven payment app is available for free for Android and iOS and works on both smartphone and tablet devices.

The Payleven dashboard can be accessed via the Payleven service website.¹ The dashboard allows merchants to view their revenue and transaction history, process refunds and manage sub-accounts.

¹service.payleven.com

2.1.1. HOW DOES IT WORK?

The Payleven payment platform is easy to use. In order to use the payment platform, the merchant first needs to register with Payleven. During the registration process, an account is created with which the merchant can log into Payleven app and access the personal dashboard. The registration process is discussed in more detail in Section 2.3.1.

The next phase on the road to accepting payments applies to setting up the Bluetooth connection between Payleven app and the Chip & PIN card reader. The merchant first needs to log into and initialize the Payleven app to accept payments before the card reader can be paired to the smartphone². Once the card reader is paired to the smartphone through Bluetooth, it is possible to initiate payment transactions provided that the smartphone has a stable Internet connection via the cellular network (3G/4G) or WiFi and has location services enabled. The Bluetooth pairing process is elaborated in more detail in Section 2.3.3.

The merchant can start a payment transaction by entering the transaction amount into the Payleven app. The transaction request is then communicated to the card reader via the Bluetooth connection. The customer then enters his or her banking card and corresponding personal identification number (PIN) to authorize the transaction request. The transaction process is described in more detail in 2.3.4. The merchant is also capable of processing refunds of certain payment transactions.

Overall, the Payleven payment platform can be split into the following use cases (or, phases of use):

1. **Registration process** - registering with Payleven;
2. **App login process** - login into Payleven app;
3. **App initialization process** - registering the Payleven app for Chip & PIN (EMV) payments;
4. **Bluetooth pairing process** - pairing the card reader to the smartphone that runs the Payleven app;
5. **Transaction process** - conducting a payment transaction on the Payleven payment app;
6. **Refund process** - processing the refund of a payment transaction.

2.1.2. BUSINESS MODEL

Payleven charges fees for the processing of payment transactions. The fee rate is dependent on the merchant's monthly revenue. The rate starts at 2.75% for revenues till €2,000 and drops to 1.5% for revenues from €12,000. The flexible fee applies to all major payment methods such as Mastercard, Maestro, Visa

²The term smartphone also encompasses tablets and other smart devices running Android or iOS

and V Pay. American Express (AMEX) is excluded from the flexible fee; AMEX payments are always charged 2.75%.

Further, no additional monthly fees are charged. There are no fixed costs except for the purchase of the Chip & PIN card reader (€80). Merchants only pay for the service provided by the Payleven payment platform.

The 2.75% fee is applied by default on every transaction. If a lower fee is applicable due to higher revenues, the difference will be paid out by Payleven at the beginning of the next month.

The minimum transaction amount accepted by the Payleven payment platform is €1. The maximum amount is €750 for individuals (i.e., merchant accounts using a private bank account) and €10,000 for business accounts (i.e., accounts coupled to a corporate bank account).

2.1.3. ADYEN: THE PAYMENT SERVICE PROVIDER

The payment transactions carried out on the Payleven payment platform are processed by Adyen.³ Adyen is a payment service provider (PSP), headquartered in Amsterdam, The Netherlands. The role of a PSP is to enable (online) shops to accept electronic payments by a variety of payment methods such as credit cards, bank transfers and local online payment methods (e.g., iDeal in the Netherlands) via a single payment gateway. A PSP connects merchants to multiple payment networks and financial institutions via one simple payment platform, making them free of setting up the payment infrastructure themselves and less dependent on the acquiring banks and card associations (i.e., Mastercard, VISA). Adyen supports 124 payment methods and is active in 64 countries across several continents.⁴

Adyen is an important actor of the Payleven payment platform. However, from the viewpoint of the merchant, the interference of Adyen within the platform is not directly evident. The presence of Adyen only becomes clear on the merchant's bank statement; that is, the merchant is paid out by Adyen as will become apparent in Section 2.3.4. Payleven does not mention Adyen's involvement anywhere on their product website or on the personal dashboard. Payleven does, however, mention the existence of a third party or 'Acquirer' in their terms and conditions⁵.

The research in this thesis makes a few things clear with regard to the involvement of Adyen within the Payleven payment platform:

- the Payleven app directly communicates with Adyen during the transaction phase. Transaction data is directly transmitted to Adyen.
- Adyen is responsible for the registration of the Payleven app and the Chip & PIN card reader during the app initialization and Bluetooth pairing process;

³<https://www.adyen.com/nl>

⁴<https://www.about-payments.com/knowledge-base/provider/adyen>

⁵<https://payleven.co.uk/terms/>

- Adyen is responsible for the functioning of the card reader; e.g., Adyen provides the card reader with the necessary firmware updates;
- some essential parts of the Payleven payment app are developed by Adyen;
- the framework for the Bluetooth message exchange is provided by Adyen.

All in all, the actual payment infrastructure of the Payleven payment platform, the card reader and the core part of the payment app are all fully provided and maintained by Adyen, whereas Payleven is responsible for the personal dashboard, customer support and app support (i.e., the issuing of updates).

2.1.4. SCHEMATIC OVERVIEW OF THE PAYMENT PLATFORM

Figure 2.1 shows a schematic overview of the Payleven payment platform. The overview depicts the various components and actors of the system and their connections. The components and actors are numbered, whereas the connections are labeled with letters.

The Payleven app and the smartphone are considered as two separate components, despite the fact that the two are strongly interwoven. This separation is necessary when describing certain functionalities; e.g., the card reader connects to the smartphone and not to the app. More importantly, this separation becomes useful when identifying the threat landscape for the Payleven payment platform.

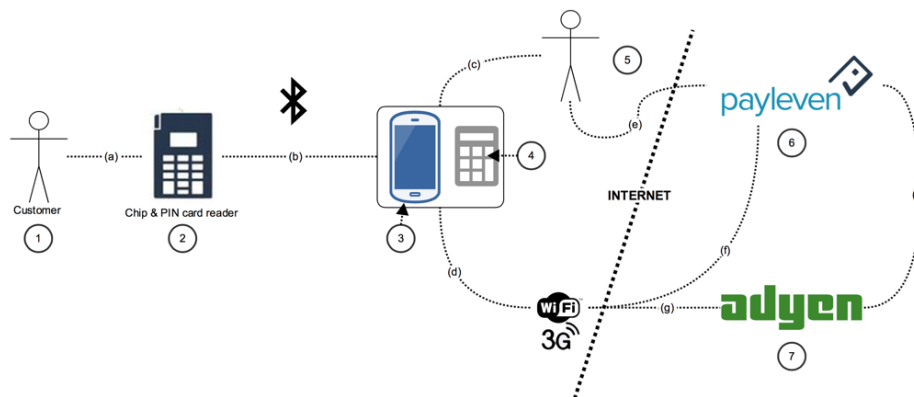


Figure 2.1: Simple schematic overview of the Payleven payment platform; (1) Customer, (2) Chip & PIN card reader, (3) the smartphone, (4) Payleven app, (5) merchant, (6) Payleven back-end, (7) Adyen (PSP).

The components and actors of the Payleven payment platform as shown in the schematic overview are:

1. the customer;
2. the Payleven Chip & PIN card reader;
3. the smartphone (Android or iOS device);

4. the Payleven app;
5. the merchant;
6. the Payleven back-end (including the personal dashboard);
7. the PSP (Adyen)

In the overview, each connection denotes a relationship or interaction between two components or actors. These connections are described as follows:

- a) the customer interacts with the card reader by entering his or her banking card and confirming the payment transaction with the corresponding PIN;
- b) the card reader and the Payleven app/smartphone exchange transaction messages via Bluetooth technology - the Bluetooth connection;
- c) the merchant enters a transaction amount on the Payleven app and starts a payment transaction;
- d) the smartphone connects via the cellular network (3G) or WiFi to the Internet - the wireless network connection;
- e) the merchant consults the transaction overview on the personal dashboard via the Payleven service website.
- f) the Payleven app communicates with the Payleven back-end;
- g) the Payleven app exchanges transaction data with the back-end of the PSP;
- h) Adyen and Payleven must share a communication channel with one another. This communication channel is not visible for the outside observer, therefore it is unknown what data is exactly exchanged between the two entities. However, the existence of such a communication channel is evident; for example, a refund request is sent to Payleven, but processed by Adyen.

2.1.5. CERTIFICATION

The Payleven payment platform complies with the following Chip & PIN (security) standards: EMV Level 2 and PCI PTS 3.1.⁶ The EMV Level 2 Type Approval certification concerns the card reader resident application software.⁷ It tests the compliance with card application requirements according to the EMV specifications. PCI PIN Transaction Security (PTS) standard applies to Point of Interaction (POI) devices; i.e., the card reader. The standard sets out several security requirements against which card readers must comply to in order to obtain the PCI PTS certification such as: tamper-resistance, use of strong encryption, side-channel resistance etc [9].

⁶<https://payleven.co.uk/security/>

⁷<https://www.emvco.com/approvals.aspx?id=39>

Furthermore, according to the Payleven product website, all data is encrypted using SSL/TLS. Sensitive data is never stored on smartphones or the card reader. Bluetooth traffic is end-to-end encrypted.

Additionally, Payleven is certified with PCI-DSS Level 1. The PCI-DSS standard provides a baseline of technical and operational security requirements with the goal to enhance cardholder data security [10].

2.2. THE COMPONENTS OF THE PLATFORM

This section elaborates on the three main components of the Payleven payment platform, that is: (1) the Chip & PIN card reader, (2) the Payleven personal dashboard and (3) the Payleven payment app.

2.2.1. CARD READER SPECIFICATIONS

The Chip & PIN card reader used by Payleven is a Miura Shuttle, designed and manufactured by Miura Systems (UK). The card reader operates on a 32-bit ARM 9 processor in combination with 64MB of RAM and 256MB FLASH memory and runs on a Linux-based operating system: MSCLE™[11]. The card reader can be recharged through the micro USB-B connection port or the cradle charging point. A Li-ion cell is integrated within the card reader. Text output is displayed on a OLED graphics display with a resolution of 128 x 64 pixels. The key pad is similar to other PIN entry devices; it consists of 10 numeric keys (0-9), a green enter key, a red cancel key and a yellow single clear key. The card reader also features a restore point next to the powerbutton. Pressing the restore point initializes a factory reset (i.e., user data is wiped). Furthermore, the smart card reader module complies to ISO 7816 T=0 & T=1. The card reader also supports magnetic swipe (triple track).

The following figures show the appearance of the Payleven Chip & PIN card reader.



(a) The front of the Chip & PIN card reader showcasing the OLED graphics display and the keypad.

(b) The bottom side of the card reader demonstrating the card slot.

(c) The powerbutton and the restore points are located on the left side of the card reader.

(d) The USB-B port is located on the right side. The cradle charging points are located on both sides.

Figure 2.2: The Payleven Chip & PIN reader.

2.2.2. PERSONAL DASHBOARD

The Payleven personal dashboard can be accessed via the Payleven service website (service.payleven.com). The dashboard can also be approached via the 'Login' button on the Payleven product website (payleven.co.uk). Once logged in, the merchant can perform several actions on the Payleven dashboard, for example:

- view the revenue over a certain period;
- sort the revenue per payment method (i.e., credit card, debit card, cash);
- consult the transaction overview;
- look into transaction details (i.e., date and time, transaction ID, payment ID);
- print and (re)send transaction receipts;
- process refunds;
- manage employees;
- and, register card readers.

Figure 2.3 shows the starting page of the Payleven dashboard. The starting page showcases the revenues of a certain period. The length of the period can be adjusted in the filter menu, as well as the payment method and the transaction status.

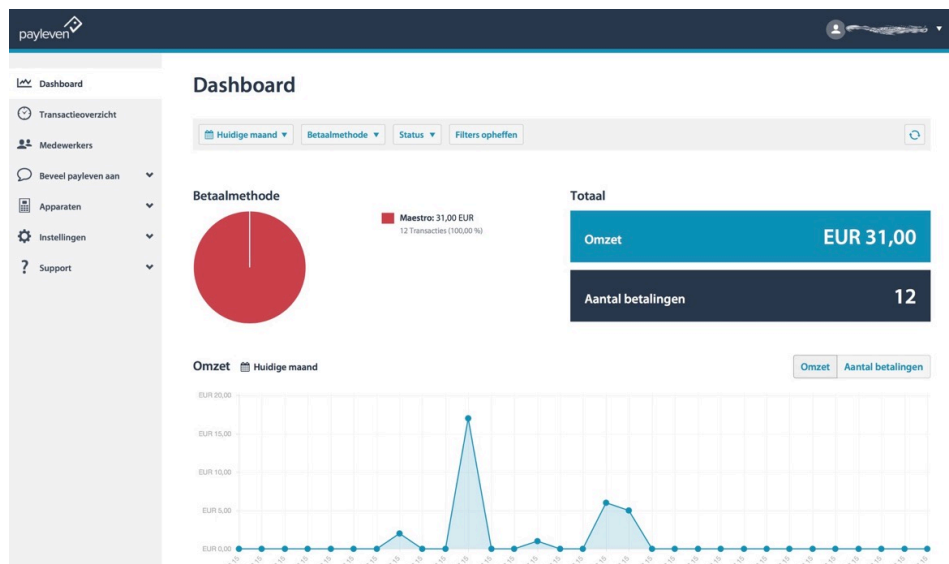
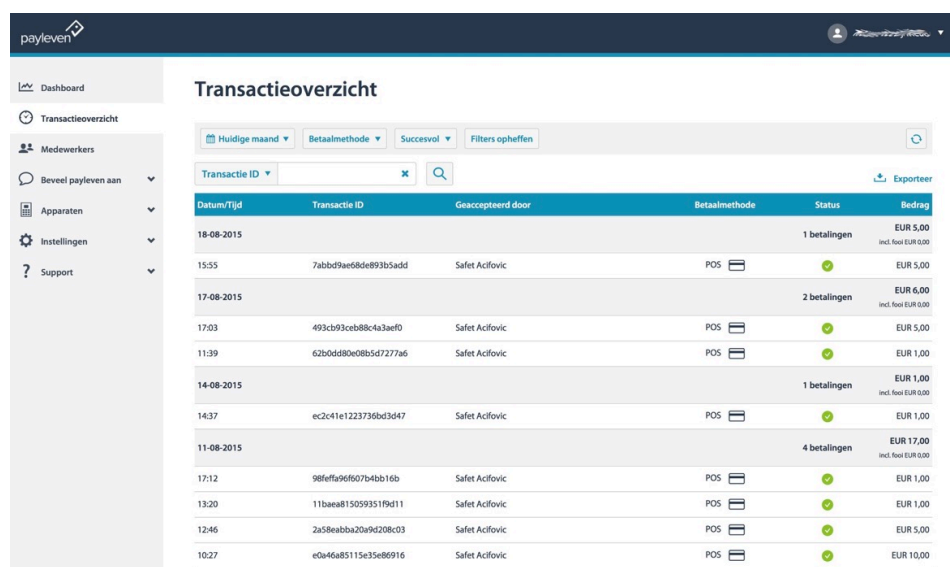


Figure 2.3: The starting page of the Payleven personal dashboard displaying the revenues of the current month. The Payleven username (right top corner) is redacted for privacy reason.

The menu of the Payleven dashboard is located on the left side of the . From there it is possible to access the transaction overview (transactieoverzicht). Figure 2.4 displays the transaction overview. The transactions are sorted from new to old; the most recent transaction is listed on top. Each transaction entry lists the time of the transaction, the transaction ID, the name of the employee that accepted the payment transaction, the payment method, the status of the transaction, and the transaction amount.

More transaction details can be retrieved by selecting a transaction entry in the overview. Figure 2.5 displays the transaction details of a payment transaction. The payment ID (Betaling ID) is now also listed at the bottom of the page. On the right top corner are four action buttons located (from left to right): (1) print receipt, (2) send receipt, (3) add receipt as PDF to Dropbox, and (4) start refund process.



| Datum/Tijd | Transactie ID | Geaccepteerd door | Betaalmethode | Status | Bedrag |
|------------|----------------------|-------------------|---------------|--------------|----------------------------------|
| 18-08-2015 | | | | 1 betalingen | EUR 5,00 incl. fooi EUR 0,00 |
| 15:55 | 7abbd7ae68de893b5add | Safet Acifovic | POS | ✓ | EUR 5,00 |
| 17-08-2015 | | | | 2 betalingen | EUR 6,00 incl. fooi EUR 0,00 |
| 17:03 | 493cb93ceb88c4a3ae0d | Safet Acifovic | POS | ✓ | EUR 5,00 |
| 11:39 | 62b0dd80e08b5d7277a6 | Safet Acifovic | POS | ✓ | EUR 1,00 |
| 14-08-2015 | | | | 1 betalingen | EUR 1,00 incl. fooi EUR 0,00 |
| 14:37 | ec2c41e1223736bd3d47 | Safet Acifovic | POS | ✓ | EUR 1,00 |
| 11-08-2015 | | | | 4 betalingen | EUR 17,00 incl. fooi EUR 0,00 |
| 17:12 | 98fffa96f607b4bb16b | Safet Acifovic | POS | ✓ | EUR 1,00 |
| 13:20 | 11baea815059351f9d11 | Safet Acifovic | POS | ✓ | EUR 1,00 |
| 12:46 | 2a58ea8ba209d208c03 | Safet Acifovic | POS | ✓ | EUR 5,00 |
| 10:27 | e0a46a85115e35e86916 | Safet Acifovic | POS | ✓ | EUR 10,00 |

Figure 2.4: The transaction overview on the Payleven dashboard.

Each payment transaction can be provided with a short description in the Payleven app when it is initiated. This description is then listed on the transaction details page. Except that for the example in Figure 2.5, the description field was left empty (stated by 'Geen beschrijving').

Furthermore, it is possible to create and add co-operator sub-accounts under the main account via the employee portal (see 'Medewerkers' in menu). Each sub-account must be provided with a fore- and surname and email address. An invitation is then sent to the entered email address after which the new employee can activate the sub-account. The access privileges of the sub-accounts are also managed via the employee portal. Sub-accounts can be set with the following access privileges: (1) only access to own transaction list and (2) access to the whole transaction list and the right to conduct refunds.

The registration of the Chip & PIN card reader is an important part of the

Transactiegegevens

Geen beschrijving EUR 1,00
Oorspronkelijk transactiebedrag EUR 1,00

17-08-2015, 11:39 Succesvol

POS

Kaartgegevens niet beschikbaar

Verwerkt door
Safet Acifovic

Transactie ID
62b0dd80e08b5d7277a6

Betalingsgeschiedenis

| Datum/Tijd | Betalings ID | Type betaling | Status | Bedrag |
|-------------------|----------------------|---------------|--------|----------|
| 17-08-2015, 11:39 | 355a562a88312bb634af | Verkoop | ✓ | EUR 1,00 |

Figure 2.5: Transaction details.

registration with Payleven. This step is completed by entering the serial number of the card reader in the device portal (see ‘Apparaten’) on the personal dashboard. The card reader is then coupled to the Payleven account. Multiple card readers can be registered under the same Payleven account. The registration of the card reader is also covered in Section 2.3.1.

2.2.3. PAYLEVEN PAYMENT APP

The Payleven payment app is available for Android and iOS and works on both smartphone and tablet devices. The app can be downloaded for free from Google’s Play Store or Apple’s App Store. The Payleven app offers practically the same functionalities as the personal dashboard: it offers the possibility to view the transaction list, (re)send receipts and start refunds. However, it does not offer the possibility to manage sub-accounts or register card readers (i.e., link to the Payleven account) . Ultimately, the function of the Payleven app is to initiate and accept payments.

For security reasons, the Payleven payment platform can only be used in the country the merchant has registered in.⁸ Hence, the Payleven app can only be used with location services enabled.

2.3. USE CASES

This section elaborates on the various use cases of the payment platform as listed in Section 2.1.1 in more detail. The Payleven payment platform covers the following use cases in the following order:

⁸<http://help.payleven.co.uk/why-does-app-need-my-location>

1. First, the merchant registers with Payleven during the **registration process**. This process includes (1) filling in company information, (2) setting up the Payleven merchant account, (3) providing a copy of ID and a recent bank statement, (4) verification of the bank account and merchant identity, and (5) linking the card reader to the merchant account;
2. The next use case concerns the **app login process**. After downloading the Payleven app from the smartphone's app store, the merchant logs into the Payleven app with the Payleven merchant account credentials (specified during the previous use case);
3. Once logged in, the app needs to be initialized for first use. This action is performed in the **app initialization process**;
4. During the **Bluetooth pairing process**, the Chip & PIN card reader is paired to the smartphone that runs the Payleven app;
5. In the end, the merchant can start payment transactions (**transaction process**) or process refunds on previous transactions (**refund process**).

It should be noted that the registration with Payleven is a one-time event. The same can be stated for the initialization of the Payleven app (although, initialization of the app is necessary for every fresh installation of the app). The Bluetooth pairing process needs to be repeated once in a while after a long period of not being used.

Moreover, these use cases are interdependent; that is to say, in order to process a transaction, use cases (1), (2), (3) and (4) must be performed first. On the other hand, in order to process a refund only use cases (1) and (2) are required; it is not necessary to pair the card reader. Use case (3) only occurs during initial use. After the app has been initialized, this step is not repeated. However, reinstalling the app or deleting its memory or cache undoes the initialization. Consequently, use case (3) needs to be repeated.

2.3.1. REGISTRATION WITH PAYLEVEN

The merchant is required to register with Payleven in order to use the Payleven payment platform. The registration is conducted online via a form on the Payleven website.⁹ It is not necessary to already be in possession of the Payleven Chip & PIN card reader during the registration process, however a card reader is required to complete the application. The hardware can be ordered afterwards via the Payleven product website, personal dashboard or purchased at any participating store.

The account details such as email address and password are specified at the beginning of the registration process. These details form the login credentials for the Payleven account. It is not possible to change the email address after registration.

⁹<https://payleven.nl/registration/?login=email@address.com>

The Payleven payment platform can only be used by business owners; private individuals cannot participate. Hence, business details need to be specified such as business type (Ltd, partnership, self-employed sole trader/freelancer)¹⁰, registered company name, company number¹¹, trading company name, business category and the corresponding business registered (trading) address. Business owners are also requested to specify personal details such as full name, date of birth and personal address and provide a copy of ID Card for verification.

The last part of the application involves the specification of the bank details such as account holder name, bank name, IBAN and BIC. The type of the bank account depends on the business type specified previously. Freelancers may use their personal bank account. Other business types require a business bank account, or a bank account shared by all partners.

A Payleven account is created after all information is provided. From that point, it is possible to access the merchant service area; a personal dashboard that is accessible through the Payleven service website. The registration with Payleven is however not completed. What remains is the registration of the Chip & PIN card reader. During this registration, the card reader is permanently linked to the merchant's Payleven account. It is now not possible to use the card reader with an another Payleven account. Extra card readers may be linked to one Payleven account, provided that the readers are not already linked to an another Payleven account. The registration of the Chip & PIN card reader can be completed by entering the serial number of the reader in the Payleven dashboard. The serial number of the card reader is located on the back of the device by means of sticker.

Once the card reader is registered, Payleven will start with the verification of the details specified earlier during the registration process. The bank details are verified via a bank transfer from Payleven. Figure 2.6 shows the bank entry on the merchant's personal bank statement. The entry consists of a unique verification code in the description field ('Omschrijving') and an arbitrary monetary amount. The verification code ('PLYN6VPXW9') and the transfer amount ('1.32') must then be entered on the Payleven dashboard in order to complete the verification process. In addition, a recent bank statement from the concerning bank account is requested by Payleven to validate the validity of the bank account; name, date and the IBAN need to be visible on the statement. The account holder name needs to match the name of the business owner (i.e., the merchant).

2.3.2. APP LOGIN AND INITIALIZATION

The merchant can start accepting payments once the registration step is fully completed; that is, the identity of the merchant is verified, the specified bank account is verified, and at least one card reader is registered under the Payleven account. The merchant then only needs to download the Payleven app and log into the app with the account details specified during the registration process.

¹⁰In the Netherlands other business forms exist: eenmanszaak, V.O.F., B.V., N.V., vereniging, stichting, maatschap

¹¹Dutch: KVK nummer



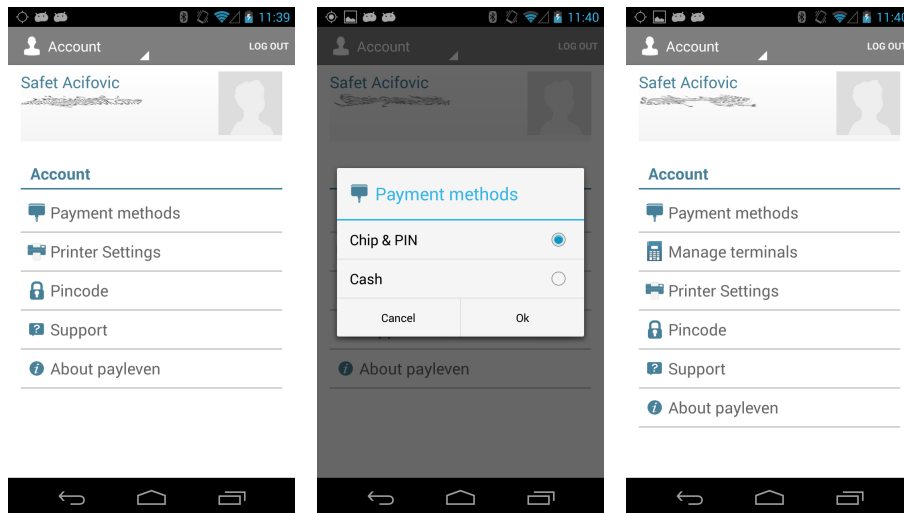
Figure 2.6: Bank transfer from Payleven listed on the merchant's bank statement. The unique verification code (PLYN6VPXW9) and the transfer amount ('1.32') listed within the bank entry must be entered on the Payleven dashboard to verify the bank account.

The pairing of the card reader and the smartphone is handled via the Payleven app. However, during the first start up of the Payleven app it is not possible to initiate a pairing process. This is because the option to start the pairing process is missing in the account menu. The Payleven app first needs to be 'initialized' by selecting the Chip & PIN payment method in the 'Payment methods' option in the account menu (see Figure 2.7). After selecting and confirming 'Chip & PIN', the Payleven app is activated and ready to accept payments. As a result, a menu option to manage card readers appears in the account menu ('Manage terminals', see Figure 2.7c). It is now possible to start the Bluetooth pairing process between the card reader and the smartphone. Note that the app refers to the card reader as terminal.

2.3.3. BLUETOOTH PAIRING PROCESS

The Bluetooth pairing process between the card reader and the Payleven app proceeds as follows:

1. activate Bluetooth on the smartphone;
2. in the Payleven app, select 'Manage terminals' in 'Payment > Account' (Figure 2.8a);
3. press the '0' key on the card reader for 5 seconds to activate Bluetooth. The message 'Accept new device' will appear on the card reader;
4. the Payleven app will display the message that a card reader has been found: 'Terminal found';
5. a Bluetooth pairing request will appear on both devices. Figure 2.8b shows the pairing request on the smartphone. Numeric comparison is used as the pairing method. Both the card reader and the smartphone will display a 6-digit numerical passkey. The devices are paired by confirming that the two passkeys are the same on both devices;



(a) The account menu of the Payleven app prior to the app initialization. A menu option to pair the Chip & PIN card reader is missing.

(b) The app is activated after selecting the Chip & PIN payment method.

(c) The account menu after the Chip & PIN payment method was selected. An option to manage the card readers is now listed in the menu (i.e., 'Manage terminals').

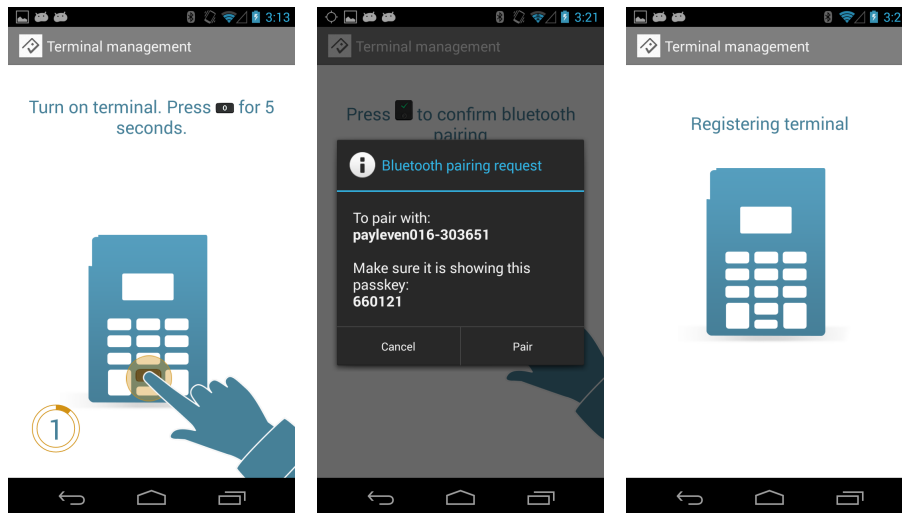
Figure 2.7: 'Initialization' of the Payleven app by selecting the Chip & PIN payment method.

6. the card reader is then being registered. This process takes some time (Figure 2.8c);
7. the card reader is then ready to use.

It should be noted that the above-mentioned steps only cover the Android version of the Payleven app. The Bluetooth pairing process is slightly different on iOS. Also worth mentioning, the Bluetooth pairing process covers a standard pairing procedure and is not specific for the Payleven Chip & PIN card reader or the Payleven app.

The Bluetooth pairing process can also take place outside of the Payleven app. That is to say, the process can be initiated via the Bluetooth settings menu in Android provided that the initial pairing process has been conducted via the Payleven app. The above-mentioned steps cover the pairing process via the app.

Normally, the Bluetooth pairing request must be confirmed on both devices by pressing the 'Pair' button on the smartphone and the green enter key on the card reader. However, interestingly, when the pairing process is initiated in the Payleven app, pressing only the green enter key on the card reader is sufficient to confirm and complete the pairing request. When initiated via the Android settings menu, the pairing request must be confirmed on both devices.



(a) The Payleven app is searching for a Chip & PIN card reader (terminal). Bluetooth on the card reader is activated by pressing the '0' key for 5 seconds.

(b) Bluetooth pairing request listing the passkey. The same key is also displayed on the card reader. The merchant must then confirm whether the two passkeys are the same.

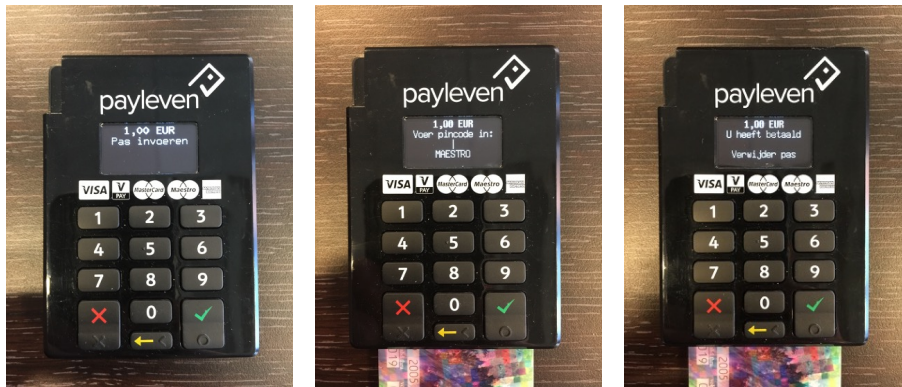
(c) The Chip & PIN card reader is then being registered after the pairing request has been confirmed. The registration process may take a few moments.

Figure 2.8: The Bluetooth pairing process.

2.3.4. ACCEPTING PAYMENTS

Payments can be carried out once the Bluetooth pairing process has been completed. Figure 2.10a shows the payment start screen of the Payleven app. The numeric keypad consists of 12 keys and is used to enter a monetary amount. It is furthermore possible to add an image and description to the payment. The Payleven app also provides the possibility to scan barcodes.

The 'Continue' button is enabled once an amount is entered, as can be seen in Figure 2.10b. The minimum transaction amount is €1. The payment transaction is started at the moment the 'Continue' button is pressed. The merchant can then choose whether to conduct the payment process via the Chip & PIN card reader or in cash. Payments done in cash can also be consulted in the Payleven dashboard. The Payleven platform then primarily serves as an administrative tool. By selecting the Chip & PIN card reader, the payment request is generated and sent to the card reader. The Payleven app will then enter the transaction stage as shown in Figure 2.10c, 2.10d and 2.10e. On the other end, the customer is prompted to insert a bank card into the card reader and confirm the payment transaction as demonstrated in Figure 2.9a and 2.9b.



(a) The display of the card reader shows the transaction amount with the message 'Pas invoeren'. (English: 'Insert card').

(b) The card reader asks for the PIN when the payment card is inserted. The payment type card is also displayed; i.e., Maestro. (English: 'Enter PIN')

(c) The card readers shows the message that the payments has been successfully completed. (English: 'You have paid. Remove card'.)

Figure 2.9: The Chip & PIN card reader during the transaction stage.

Figure 2.9 showcases the different stages of the Chip & PIN card reader during the transaction. The card reader will prompt the customer to insert a payment card and enter the corresponding PIN in order to authorize the payment transaction. After the correct PIN has been entered, the card reader will process and authorize the payment and the transaction amount will be debited from the customer's bank account. Figure 2.10f and 2.9c demonstrate a successful transaction on respectively the Payleven app and the card reader. From there, the merchant can choose to send the receipt to the customer by entering the customer's email address or to skip the receipt and to initiate a new payment.

Figure 2.11 shows the bank entry of the debited transaction amount on the customer's bank statement. The description of the bank entry states the merchant's company name and its location preceded by the three letter code 'PLV'. The letter code is an abbreviation for Payleven and indicates that the transaction was conducted on the Payleven payment platform.



Figure 2.11: The debited transaction amount as seen on the customer's bank statement.

Figure 2.12 shows the bank entry of the credited transaction amount on the merchant's bank statement. The credited amount is €0.97; the remaining €0.03 form the commission fee of the Payleven payment platform (2,75% rounded up).

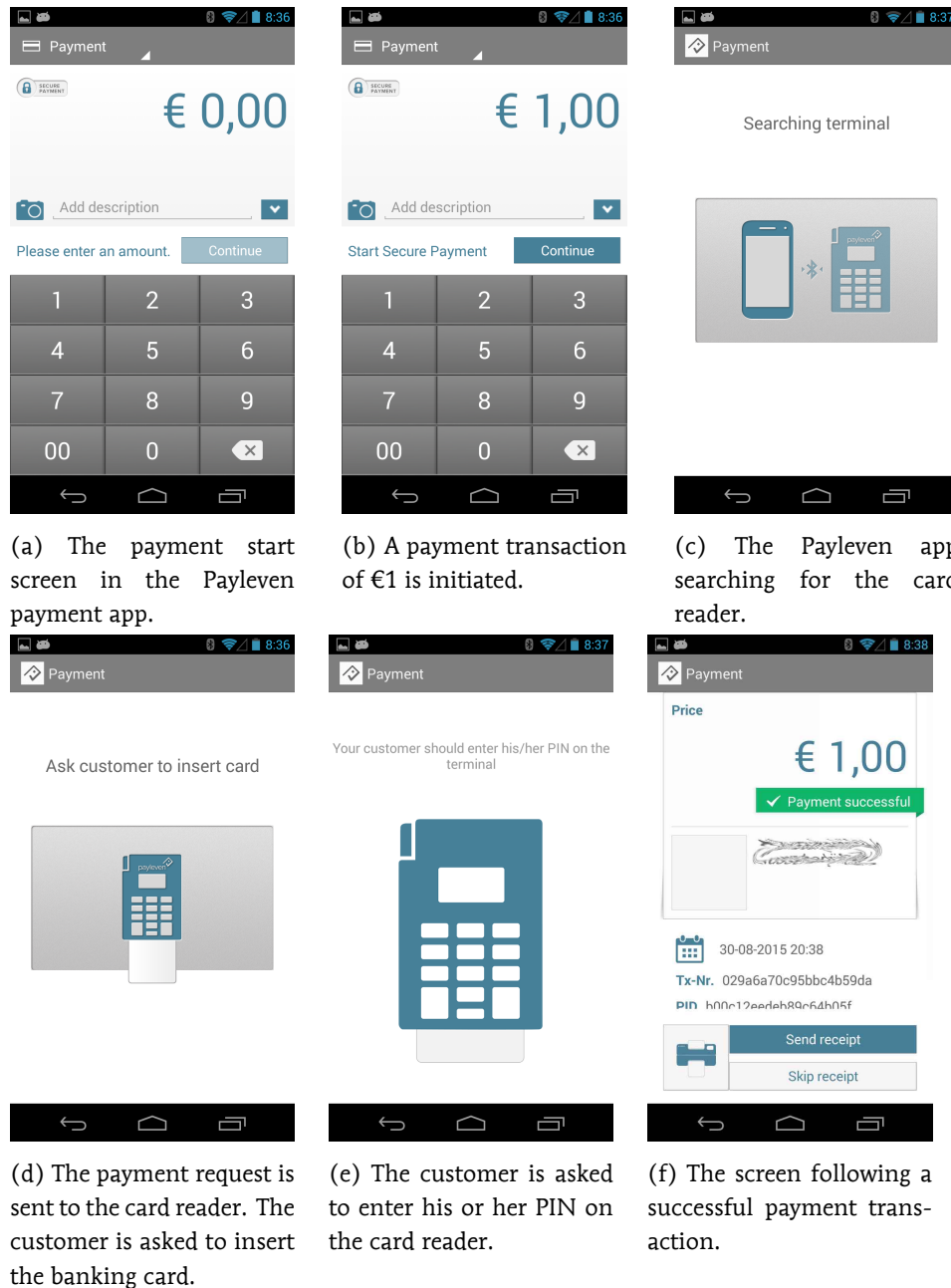


Figure 2.10: Payment transaction in the Payleven app.

It should be noted that the transaction is not directly credited on the merchant's bank account. It takes approximately 1 to 3 days for the transaction to be processed and paid out. According to Payleven, payouts are performed on Tuesdays, Thursdays and Fridays.

As can be seen in Figure 2.12, the transaction is paid out by STG ADYEN. The entry does further not list the customer's bank account number or bank card number.






| Omschrijving | Type  | Bedrag (€) |
|--|--|--|
| Naam:STG ADYEN Omschrijving:TX3960409125XT batc | OV | 0,97  |
| Tegenrekening  | | |
| Mutatiesoort Overschrijving | | |
| Mededelingen Naam:STG ADYEN Omschrijving:TX3960409125XT batc h 1,  | | |
| Kenmerk:TWDJGAEAJBCFWT | | |
| IBAN:NL44 0153 0505 0001 0001 0001  | | |

Figure 2.12: The credited transaction amount as seen on the merchant's bank statement.

2.3.5. REFUNDS

The Payleven payment platform offers the possibility to refund payments. Refunds can be processed both via the personal dashboard and the Payleven app. Initially, the refund option is disabled. To activate the possibility to process refunds, a numeric 4-digit refund authorization PIN needs to be set up first for security purposes. This PIN can be created via the personal dashboard or the Payleven app. When a refund action is initiated, the merchant is prompted to enter the PIN to authorize the refund. It is furthermore possible to change the refund authorization PIN or disable the refund option at any time. The current PIN is then asked for confirmation.

Figure 2.13 shows the refund of a transaction in the transaction overview on the personal dashboard. The refunded transaction is grayed out and the transaction status holds a yellow circle under the 'Status' column.



| Datum/Tijd | Transactie ID | Geaccepteerd door | Betaalmethode | Status | Bedrag |
|------------|----------------------|-------------------|---|---|--|
| 30-08-2015 | | | | 0 betalingen | EUR 0,00 <small>Incl. fooi EUR 0,00</small> |
| 20:38 | 029a6a70c95bbc4b59da | Safet Acifovic | POS  |  | EUR 1,00 |

Figure 2.13: The refund of a transaction in the transaction overview.

As can be seen in Figure 2.14, the refund entry is also added in the payment history in the transaction details portal of the refunded transaction. Note that the refund transaction holds a different payment ID. Also, refunds can only be performed on transactions that are not paid out (i.e., credited on the merchant's bank account). Transactions that are already successfully paid out cannot be refunded via the Payleven platform.

Betalingsgeschiedenis





| Datum/Tijd | Betalings ID | Type betaling | Status | Bedrag |
|---|----------------------------------|---------------|---|-----------|
|  31-08-2015, 12:47 | d4a71462b89d47edb41c51f38e516e3e | Terugbetaling |  | EUR -1,00 |
|  30-08-2015, 20:38 | b00c12eedeb89c64b05f | Verkoop |  | EUR 1,00 |

Figure 2.14: The refund ('Terugbetaling') as shown in the payment history ('Betalingsgeschiedenis') in the transaction details portal.

2.4. HIGH-LEVEL MESSAGE SEQUENCE CHARTS OF THE VARIOUS USE CASES

This section visualizes the interaction among the different components of the Payleven payment platform during the various use cases on the basis of Message Sequence Charts (MSCs). These charts also give a rough outline of the observed communication with Payleven and Adyen; the MSC's do not set out the exact contents of the message exchange, but rather indicate when a particular communication channel with Payleven or Adyen was established. A more detailed view on the network traffic is given in the network analysis in Chapter .

2.4.1. MSC: APP LOGIN & INITIALIZATION

Figure 2.15 shows the MSC of the app login and app initialization process. The arrows P1 and P2 represent the message exchange with Payleven respectively before and after a successful login. A1 states the message exchange with Adyen.

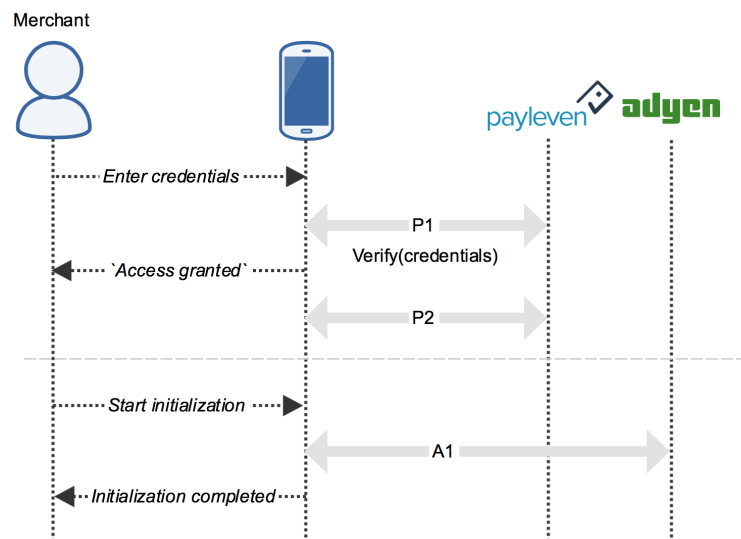


Figure 2.15: High-level message sequence chart of the app login and app initialization process.

- **P1** and **P2** are elaborated in more detail in Section 4.2;
- **A1** is examined in Section 4.3.

2.4.2. MSC: BLUETOOTH PAIRING PROCESS

Figure 2.16 visualizes the MSC of the Bluetooth pairing process. The messages are numbered according to the steps listed in Section 2.3.3. The Bluetooth connection between the Payleven app and the Chip & PIN card reader is denoted by a gray area. The Bluetooth message exchange is discussed in Chapter 5.

P3 and A2 represent the network traffic during the Bluetooth pairing process. This network communication takes place during the registration of the Chip & PIN card reader *after* a successful pairing request (see Figure 2.8c).

- **P3** is discussed in more detail in Section 4.4;
- **A2** is also highlighted in Section 4.4.

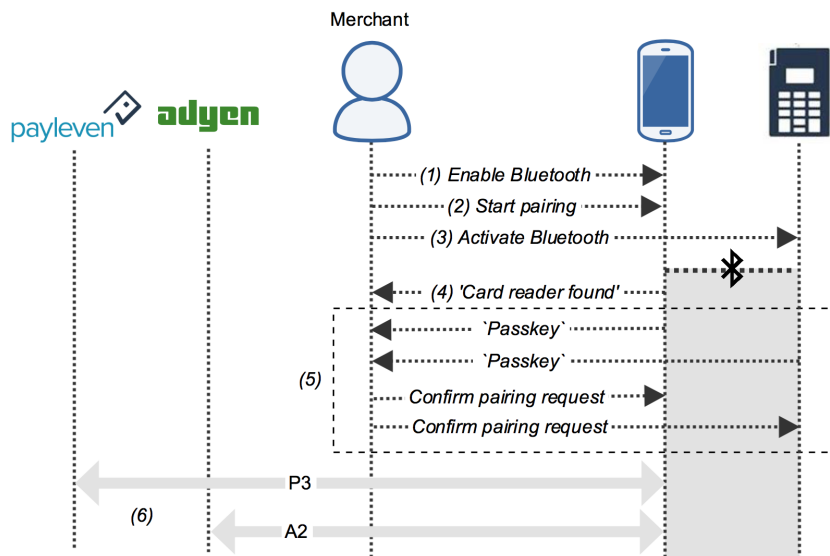


Figure 2.16: High-level message sequence chart of the Bluetooth pairing process.

2.4.3. MSC: TRANSACTION PROCESS

Figure 2.17 presents the MSC of the transaction process. The chart also covers the interaction among the customer and the Chip & PIN card reader. The gray area represents the Bluetooth message exchange between the smartphone and the card reader. A detailed description of the Bluetooth traffic is given in Chapter 5. The network traffic during the transaction phase can be divided into four parts:

- P4 represents the communication with Payleven at the start of the transaction;
- A3 includes the message exchange with Adyen after the banking card has been inserted by the customer;
- A4 states the message exchange with Adyen after the PIN has been entered by the customer;
- P5 represents the communication with Payleven at the end of the transaction.

A detailed description of the network traffic during the transaction process is given in Section 4.5.

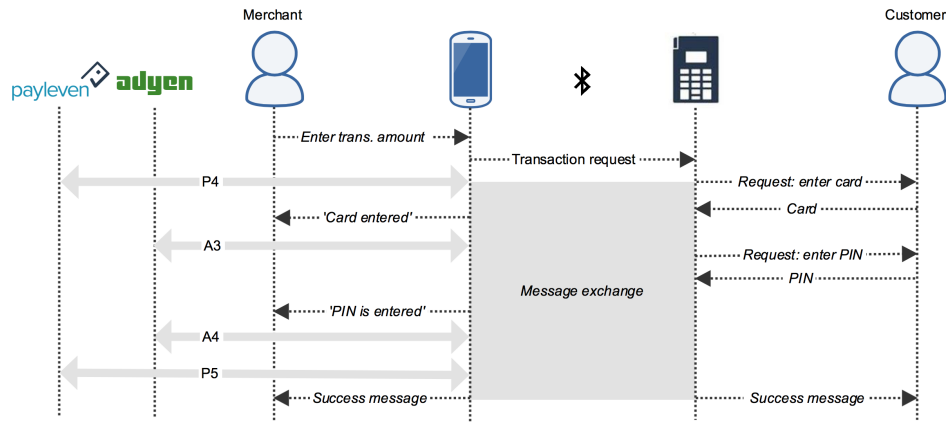


Figure 2.17: High-level message sequence chart of the transaction process.

2.4.4. MSC: REFUND PROCESS

Figure 2.18 shows the interaction among the merchant, the Payleven app and the Payleven back-end. What is interesting in this chart is the absence of Adyen. During the refund process, the Payleven app provides the Payleven back-end with the refund details and the refund authorization PIN.

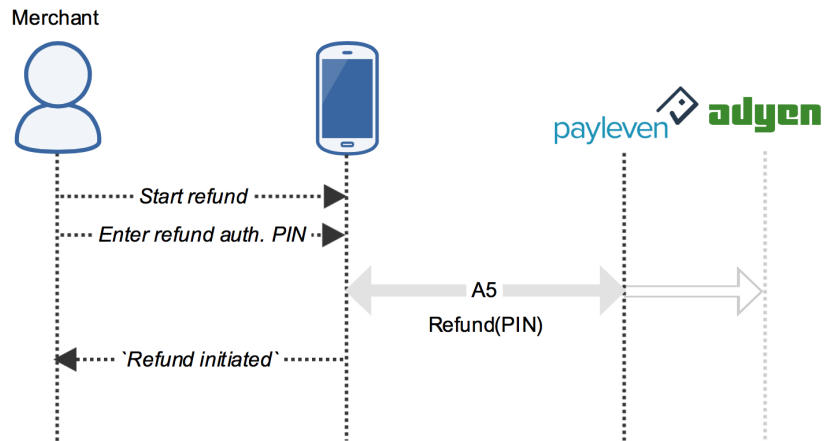


Figure 2.18: High-level message sequence chart of the refund process.

Overall, this observation indicates that Payleven maintains an internal connection with Adyen. The Payleven back-end probably forwards the refund request to Adyen after validating the refund authorization PIN. The network traffic during the refund process is elaborated in Section 4.6 in more detail. Unfortunately, it is not possible to monitor the internal traffic between Payleven and Adyen.

3

METHODOLOGY

This chapter comprises two parts. The first part describes the research approach, the used materials and the applied research methods. The second part outlines the threat landscape of the Payleven payment platform including the several attack points and attack vectors.

3.1. RESEARCH APPROACH

This research evaluated the information security of the Payleven payment platform. The security evaluation was not determined by a general roadmap, but was mainly based on a flexible ad hoc approach. That is to say, the research direction was continuously subject to change due to shifting factors as the research unfolded and new findings were uncovered.

This chapter describes the methodology, methods and tools employed to evaluate the information security of the Payleven payment platform. The security evaluation was aimed at:

1. providing useful insights about the workings of the payment platform - a system evaluation;
2. and, evaluating the payment platform based on these insights against a set of security requirements and from the point of view of a malicious attacker.

The first objective automatically contributes to the feasibility of the second objective; a thorough understanding of the underlying system is required in order to perform a useful security evaluation.

Correspondingly, the system evaluation was split into multiple research stages:

1. analysis of the **Bluetooth traffic**;
2. analysis of the **network traffic**;

3. analysis of the **Payleven smarthphone app**;
4. analysis of the **card reader hardware**;
5. and, analysis of the **card reader firmware**.

Each separate research stage contributed to a better understanding of the Payleven payment platform. The analysis covered the information flow and the underlying (infra)structure of the platform, and was performed with information security in mind. Section 3.3 describes the methods and tools that were used to apply the analysis on the above-mentioned sub-systems.

Furthermore, a proper security evaluation requires the mindset of an attacker. Therefore, section 3.4 lays out the threat landscape of the Payleven payment platform. This includes the identification of:

- possible entry points (*What should be attacked?*);
- attack vectors (*How could the system be attacked?*);
- assets (*What should be protected?*);
- attacker types (*Who could attack the system?*);
- and, attack goals (*Why should the system be attacked?*).

The section also provides a list of possible threats relevant for the Payleven payment platform.

In the end, the security evaluation consisted of (1) scrutinizing the Payleven payment platform and (2) identifying the relevant threat model, and (3) combining the knowledge obtained in the last two points to draw a conclusion with respect to the information security of the Payleven payment platform.

3.2. MATERIALS & SETUP

In terms of material selection and research setup, the global setting was mostly already predefined by the Payleven payment platform. The materials consisted of:

- two Payleven Chip & PIN card readers;
- one rooted Galaxy Nexus (Samsung GT-I9250) with Android 4.3;
- the Payleven payment app (version 2.23, May 2015);
- one laptop running OSX and Santoku Linux (VM);
- and, one ING Maestro debit card.

In order to inspect the insights of the card reader, it has been taken apart. This rendered the card reader unusable. Therefore, an extra reader was purchased. Moreover, the two card readers had the same model number: M006-PROD03-V2-6. Both devices were purchased in a physical store in the Netherlands. The

firmware and OS version of the tested card readers were respectively ‘adyen-pl-v1_32p7’ and ‘M000-OS-V7-1’. This information was not accessible via the card reader, but required the read-out of network traffic.

The smartphone used in the setup was a Galaxy Nexus (Samsung GT-I9250) running Android 4.3. Further information on the smartphone is listed in Table 3.1.

| | |
|------------------|-----------------|
| Baseband version | I9250XXLJ1 |
| Kernel version | 3.0.72-gfb3c9ac |
| Build number | JWR66Y |

Table 3.1: Information on the Samsung Galaxy Nexus

The laptop served several purposes: first, it was used to access the smartphone to read log files, enter the file system and (un)install applications; second, it was used to perform analyses on the smartphone and the Payleven app; and last, it was used as a network proxy for the Payleven payment platform.

As regards the banking cards, the use of multiple brands of cards (e.g., Mastercard, VISA) from different Dutch banks (e.g., ING, ABN AMRO, SNS) did not affect the use and behavior of the Payleven payment platform. Therefore, all tests were conducted with one Maestro card from ING.

The final research setup for Payleven payment platform is shown in Figure 3.1.

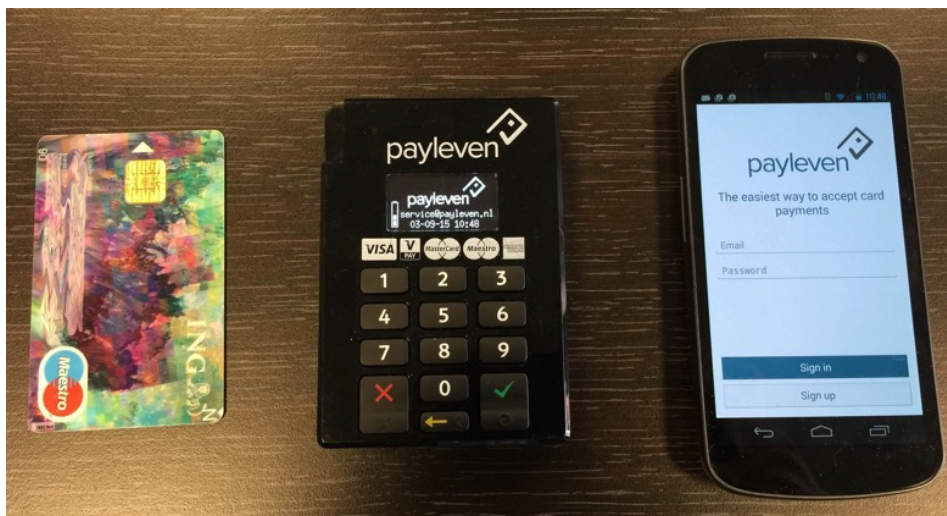


Figure 3.1: The test setup: (from left to right) the ING Maestro banking card, the Payleven Chip & PIN card reader and the Galaxy Nexus with the Payleven app.

3.3. METHODS & TOOLS

This section describes the applied methods and the tools used to conduct the research.

3.3.1. INTERCEPTING THE BLUETOOTH TRAFFIC

The wireless Bluetooth communication channel between the Chip & PIN card reader and the smartphone is the most prominent feature of the Payleven payment platform. It serves for the exchange of security-sensitive transaction data, while on the same time the wireless nature of the communication channel introduces new attack vectors. It is therefore not difficult to imagine why an attacker would target the Bluetooth connection. The impact of such an attack is analyzed in 3.4

However, monitoring the Bluetooth communication channel is not an easy task. Features such as frequency hopping and data whitening add to the difficulty of eavesdropping. Furthermore, the use of SSP (i.e., number comparison) indicates end-to-end encryption.

This research aimed at investigating the feasibility of passively eavesdropping the Bluetooth connection between the Chip & PIN card reader and the smartphone. To do so, a setup comprising a Ubertooth One had been designed. Ubertooth One is an open-source Bluetooth hardware platform directed at testing and monitoring Bluetooth traffic [12]. It is the first affordable Bluetooth monitoring platform: the price of a Ubertooth One is ranged around \$120¹. Other commercial tools cost the tenfold. Figure 3.2 demonstrates the Ubertooth One hardware. More information regarding the Ubertooth One hardware and the corresponding software can be found at the Github page of the Ubertooth project [13].



Figure 3.2: Ubertooth One

Unfortunately, the Ubertooth One setup was not very successful as it soon became apparent. The setup was capable of capturing some Bluetooth packets exchanged between the card reader and the smartphone; however, these packets only had a length of 14 bytes and contained no application data, while it was certain that the card reader and the smartphone exchanged data. The packets only listed the last part of the source address as can be seen in Figure 3.3. Wireshark was used to read-out the captured Bluetooth traffic [14].

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|------------|-------------------|-------------------|-----------|--------|------|
| 118 | 122.098629 | 00:00:00_ad:12:cd | 00:00:00_00:00:00 | Bluetooth | 14 | ID |
| 119 | 122.662220 | 00:00:00_b0:3e:db | 00:00:00_00:00:00 | Bluetooth | 14 | ID |
| 120 | 124.602348 | 00:00:00_ad:12:cd | 00:00:00_00:00:00 | Bluetooth | 14 | ID |
| 121 | 125.521110 | 00:00:00_ad:12:cd | 00:00:00_00:00:00 | Bluetooth | 14 | ID |
| 122 | 126.947579 | 00:00:00_ad:12:cd | 00:00:00_00:00:00 | Bluetooth | 14 | ID |

Figure 3.3: Bluetooth packets captured with the Ubertooth One setup listed in Wireshark.

As it turned out later, these nondescript packets could be explained by the fact that the Ubertooth One does not support Bluetooth EDR [15][16]. Both the

¹<http://hakshop.myshopify.com/products/ubertooth-one>

card reader and the smartphone could already have agreed on an EDR connection. Ubertooth is then only capable of monitoring the low speed headers for the packets sent over EDR; it cannot switch mode to capture EDR data.

It could be concluded that eavesdropping an already established Bluetooth connection is practically impossible due to lack of capable hardware equipment. Furthermore, even if Ubertooth was capable of capturing all packets exchanged between the card reader and the smartphone, the experiment would soon come up against encrypted application data.

Therefore, the setup of the experiment had been slightly adjusted. In order to understand the underlying structure of the Payleven payment platform, it was of great importance to figure out what data is exactly exchanged between the card reader and the Android smartphone. In effect, the focus of the setup had been shifted from eavesdropping the in-air Bluetooth connection to monitoring the Bluetooth traffic on the connection points (i.e., the Android device itself).

Fortunately, Android offers the possibility to log the Bluetooth traffic on the device. This option is available in the Developers option menu as of Android version 4.4 under 'Enable Bluetooth HCI snoop'. When enabled, Android will capture the Bluetooth traffic on the Host Controller Interface (HCI) of the device in question. HCI is a standardized communication channel between the host (i.e., Android) and the Bluetooth integrated circuit. The advantage of capturing traffic on the HCI is that the Bluetooth-encryption has not been applied yet, thus any data captured on the HCI is displayed in plaintext. This makes it possible to read-out any information sent over Bluetooth between the card reader and the smartphone.

The Galaxy Nexus used in this setup was provided with Android 4.3, which means that the HCI snoop option was missing in the Developers option menu. Luckily there is an Android app available that offers exactly the same functionality: Bluetooth HCI Logger [17]. The only downside of the HCI Logger app is that it requires root access, whereas the native HCI snoop option in the Developers menu does not. Moreover, the app does not work on every Android smartphone - it did, however, work properly on the Galaxy Nexus.

HCI Logger writes the Bluetooth traffic in a CAP (Network Packet Capture) file. Every time the HCI Logger or Bluetooth is enabled/disabled on the Android smartphone, a new CAP-file was created. These files were located in the file system (that is `sdcard/rsap`) of the Android operating system. After extracting the CAP-files from the Android device, it was possible to read out the Bluetooth data traffic in Wireshark.

3.3.2. ANALYZING THE NETWORK TRAFFIC

In order to get a glimpse of the data flow of the Payleven payment platform, a simple setup to monitor the network traffic of the platform had been designed. The experimental setup made use of the Wireshark packet analyzer software (version 1.12.6) and a laptop. The laptop running Wireshark was connected to the same WiFi network as the Payleven payment platform with the intention to cap-

ture the network traffic of the platform. The experiment provided information on what web addresses the Payleven app connected to. However, as expected, the contents of the captured network packets were inaccessible due to the use of a secure channel (TLS).

Therefore, the setup had been modified to get around this protection. A possible way to do so, is by deploying a man-in-the-middle (MITM) attack on the network connection between the Payleven app and the Payleven back-end. This can be achieved using a proxy. The proxy used in this setup was Burp Proxy [18].

Figure 3.4 demonstrates the experimental MITM setup. The idea behind the MITM setup is as follows:

- the proxy machine presents a self-generated certificate to the Payleven app pretending to be the genuine Payleven back-end;
- the Payleven app falls into the trap and establishes a secure channel with the proxy machine while thinking that the connection is established with the Payleven back-end;
- the Payleven app sends data over the secure channel to the proxy machine;
- the proxy machine has access to this data as the self-generated certificate was used to establish the secure channel;
- the proxy machine establishes a secure channel with the Payleven back-end using the original certificate and forwards the data over the secure channel;
- responses from the Payleven back-end are forwarded to the Payleven app;
- both the Payleven app and the Payleven-back-end do not notice that a third party is engaged in the communication.

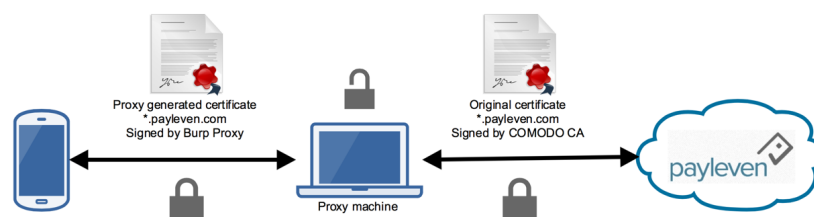


Figure 3.4: MITM setup

The self-generated certificate is similar to the original Payleven certificate; both hold the same name (i.e., *.Payleven.com). However, the self-generated certificate is signed by Burp Proxy, while the original certificate is signed and issued by a known certificate authority (Comodo). In order to ensure that the self-generated

certificate is trusted by the Payleven app, the Burp Proxy root certificate needs to be added to the smartphone (i.e., Android's trust-store). This way, the Payleven app has no reason to reject the certificate provided by the proxy machine; after all, the smartphone considers Burp Proxy now as a trustworthy certificate authority.

Unfortunately, this experimental setup did not lead to the desired results. The certificate provided by the proxy machine was rejected. The Payleven app refused to establish a connection using the self-generated proxy certificate. A plausible explanation for this would be the use of certificate pinning. Typically, certificates are validated by checking the signature hierarchy; however, it is also possible to only accept a specific set of certificates. With certificate pinning, the received certificate is matched against a set of expected certificates. The certificate is then accepted if it complies with the expectations. It should be clear that the Payleven app utilizes this technique. The Payleven app knows in advance to which hosts it will talk to, therefore it can apply certificate pinning to enhance the security.

Luckily, it is possible to disable certificate pinning on a rooted Android device by installing Cydia Substrate² and Android-SSL-Trustkiller.³ Android-SSL-Trustkiller applies various hook methods in order to bypass certificate pinning by accepting any certificates. Following the installation of these two apps on the Android device, the Payleven app started accepting the self-generated proxy certificate. The MITM setup described in Figure 3.4 ended up being effective and provided insights into the contents of the messages exchanged between the Payleven app and the Payleven back-end.

3.3.3. REVERSE ENGINEERING THE PAYLEVEN APP

The Payleven payment app is an essential part of the Payleven payment platform. It enables the communication between the Chip & PIN card reader and the Payleven back-end and it transforms the Android smartphone into a hatch for confidential information. Hence, this research also focused on the Payleven app and the Android OS. The research consisted of two parts: in the first part, the app was reverse-engineered and the source code was manually evaluated, and in the second part, static-analysis tools were applied on the Payleven app. Furthermore, the possibilities of Android hooking and app manipulation were investigated.

Android applications are distributed in the APK package file format. The APK is an archive file built on the ZIP file format and contains all the application's code, files, resources, assets and certificates (similar to the JAR file format). The core of the application is contained in `classes.dex`. The DEX file contains all the classes of the application compiled into the DEX file format, a format understandable by the Dalvik virtual machine.

Figure 3.5 demonstrates the toolchain used for reverse engineering the Payleven payment app. Firstly, the APK of the Payleven app was extracted from the Android device. There are several ways to get an APK file; for example, it is possible to

²<https://play.google.com/store/apps/details?id=com.saurik.substrate&hl=en>

³<https://github.com/iSECPartners/Android-SSL-TrustKiller>

download the APK from the Playstore using the 'APK Downloader' plug-in⁴ for Google Chrome or extract the APK from the SD-card of the device using the 'APK Extractor' Android app⁵. It is also possible to use the Android Debug Bridge (adb) command line tool to pull the APK from the Android device.

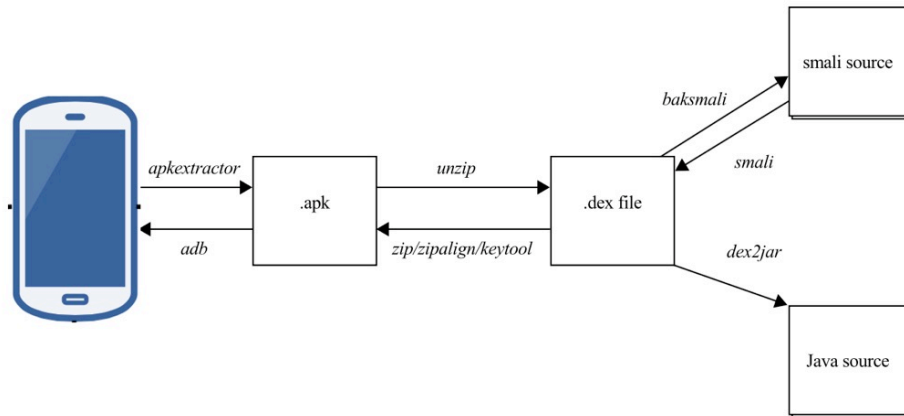


Figure 3.5: Android toolchain for reverse engineering the Payleven payment app.

Secondly, the APK file was unpacked using Apktool⁶. Apktool is a tool for reverse engineering Android binaries. It decodes resources to nearly original form and makes it possible to view smali source code. Baksmali/smali is the disassembler/assembler for the DEX format and supports the full functionality of the DEX format⁷. It transforms DEX machine code into a more readable assembly format.

Lastly, the DEX file (i.e., `classes.dex`) was decompiled into a JAR file containing Java class files. The tool used for this was dex2jar.⁸ JAD was used for decompiling the class files into java source code.⁹

ANDROID HOOKING

This research also aimed at investigating the possibilities of Android Hooking and its implications on the Payleven app. With Android Hooking, it is possible to *hook* certain method calls in runtime and replace them with custom methods. This way, the behavior of the Android app can be altered in a malicious way.

Xposed is a framework that enables Android Hooking.¹⁰ Initially, Xposed is a base-system focused on the use of third-party modules (add-ons) which can change the behavior of the Android system and apps.

⁴<https://chrome.google.com/webstore/detail/apk-downloader/cgihflhdpokeobcfimiamffejfnmfii>

⁵<https://play.google.com/store/apps/details?id=com.ext.ui&hl=nl>

⁶<https://ibotpeaches.github.io/Apktool/>

⁷<https://github.com/JesusFreke/smali>

⁸<https://github.com/pxb1988/dex2jar>

⁹<http://www.javadecompilers.com/jad>

¹⁰<http://repo.xposed.info>

3.3.4. ANALYZING THE CARD READER

The analysis of the card reader consisted of two parts: firstly, the hardware of the card reader was investigated; and secondly, the firmware was subjected to examination. As regards the hardware, the card reader was torn apart and its insides were investigated. The firmware, on the other hand, needed first to be extracted from the card reader before any research could be done. However, this caused some struggles because plugging in the card reader on a USB port on a computer did not immediately reveal its file system. The reader was not recognized as a USB device. It was just charging. What then seemed to help was to restart the card reader while being plugged-in. During startup, the card reader appeared in a pop-up on the screen asking permissions to be mount (only Linux, did not work on OSX). Unfortunately, mounting the device still did not give any access to its file system and the firmware. Two tools needed to be applied:

- `dd` - is a command-line utility tool to copy and convert files [19];
- `Binwalk` - is a tool for analyzing and extracting firmware images [20].

The `dd` tool was used to copy the file system located on the card reader into an ISO-file. `Binwalk` was then used to analyze the data contained within the ISO-file. This way it was possible to identify and extract the firmware of the card reader.

3.4. THREAT MODEL

This section describes the threat landscape for the Payleven payment platform.

3.4.1. SECURITY REQUIREMENTS

Before the threat landscape can be defined, it is first useful to make a list of general security requirement relevant to the Payleven payment platform. The following requirements are of importance to the payment platform:

- integrity of data¹¹, the card reader and the Payleven app;
- confidentiality of data;
- authenticity of data;
- availability of data, the card reader and the Payleven app;
- and, non-repudiation of transactions.

A violation of one of these requirements may cause damage to the Payleven payment platform and all involved parties.

3.4.2. ASSUMPTIONS

The following assumptions were taken into account when defining the threat landscape.

- The attacker has full access to the (wireless) communication;
- The attacker has access to the card reader (slot);
- The attacker has access to the Payleven app;
- The attacker has access to the smartphone;
- The attacker has no access to the Payleven back-end, the database(s) and the internal connection with Adyen and other financial situation. The back-end is a black box. The attacker has however access to the Payleven personal dashboard.

3.4.3. ATTACK POINTS

Attack points define the interface through which potential attackers can interact with the application or supply it with malicious data. Several potential attack points can be identified in the Payleven application overview shown in Figure 3.6. These attack points are listed in Table 3.2.

¹¹all data processed by the Payleven payment platform

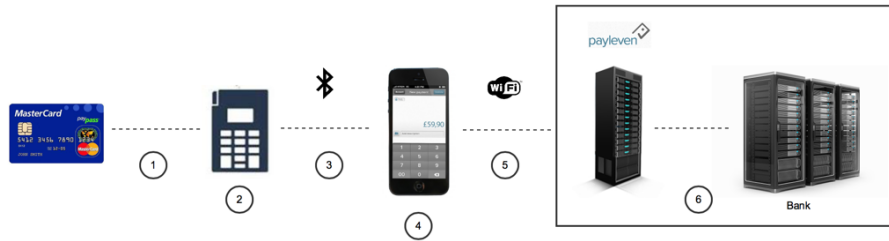


Figure 3.6: Application overview of the Payleven payment platform.

| | Attack Point |
|---|---|
| ① | <ul style="list-style-type: none"> • Smart card slot • Communication channel between the smart card and card reader |
| ② | <ul style="list-style-type: none"> • Card reader hardware • Card reader firmware |
| ③ | <ul style="list-style-type: none"> • The Bluetooth connection • The Bluetooth pairing process |
| ④ | <ul style="list-style-type: none"> • Smartphone hardware • Mobile operating system • Payleven payment app |
| ⑤ | <ul style="list-style-type: none"> • WiFi connection • Cellular network connection |
| ⑥ | <ul style="list-style-type: none"> • Payleven back-end servers and databases • Adyen service provider servers |

Table 3.2: Potential attack points for the Payleven payment platform.

3.4.4. ATTACK VECTORS

Multiple potential attack vectors can be devised based on the attack points listed in Table 3.2. An attack vector is a path or method that an attacker *could* use to attack the target.

MALICIOUS HARDWARE

Potential attackers could add malicious hardware skimmers to individual Chip & PIN card reader in order to gather PIN data or alter transaction information. The smartphone itself could also be modified - although this is not very common. Hardware skimmers could be inserted whenever the card reader is left unattended. This can even take place before the hardware actually reaches the merchant. However, the form factor of the hardware could form an obstacle to attackers. It increases the difficulty of developing and installing unobtrusive hardware skimmers.

Attackers could target the communication channel between the smart banking card and the card reader. As a result, confidential card details and PIN could be disclosed.¹²

Attackers could also add probes into the electronic circuit to capture keystrokes or display incorrect data on the OLED graphics display. The *What You See Is What You Sign* principle is then violated; the display shows an different amount than the amount being debited by the merchant.

Because of the relative low price of the Payleven Chip & PIN card reader (ca. €80), attackers may choose to perform replacement attacks, in which the legitimate card reader is replaced with a malicious one. Although it is very easy to replace the card reader - it is after all completely wireless and mobile - the effectiveness and feasibility of such attack is uncertain given that the Payleven Chip & PIN card reader is linked to a specific Payleven user account during the registration with Payleven.

MALICIOUS FIRMWARE

The firmware is essential for the functionality and security of the Chip & PIN card reader. The exploitation of a vulnerability by an attacker could fully comprise the card reader and could cause significant damage to all involved parties. The firmware can be attacked through the four entry points of the card reader: (1) via the USB-B port, (2) via the Bluetooth component (e.g., via a malicious app), (3) via the smart card slot (e.g., with an infected smart card), and (4) via the keypad. The card reader may also be attacked through the JTAG interface - at least if present on the device - that allows full debugging control of the card reader. JTAG is a standardized hardware access point for testing purposes [21].

SMARTPHONE SOFTWARE

Various attack vectors can be described in the context of smartphone software. Attackers may develop malicious mobile applications that alter the behavior of the Payleven app. These malicious apps could, for example, pass on sensitive information (e.g., read from local storage), modify transaction information or spoof transactions. The possibilities are comprehensive. The only obstacle is getting the malware on the users' mobile device. However, mobile users are often easily tricked into installing (fake) malicious mobile applications.

Attackers may also develop a fake Payleven app (so called rogue app) that phishes login credentials or rewrites the money from fake transactions to a malicious destination account. In order to be able to take advantage of faults in the Payleven payment platform, attackers first need to obtain knowledge of the logics behind the payment app. This can be achieved in three ways: (1) decompiling the corresponding installation file and analyzing the source code, (2) testing several usage scenarios in order to induce faults and find inconsistencies, and (3) fuzz-testing the mobile app. Once a vulnerability is found, it can be exploited. These exploits can be then used to alter the behavior of the app. Additionally, attackers may also apply *hooking* (catch and manipulate method calls) or abuse

¹²The PIN can only be intercepted when Static Data Authentication is used.

vulnerabilities in the mobile operating system to alter the behavior of the Payleven app.

NETWORK

The wireless nature of the Payleven payment platform makes it susceptible to remote attacks. The smartphone and the card reader could be externally manipulated to disrupt the correct behavior of the Payleven payment platform. Furthermore, an attacker could eavesdrop the wireless communication channel and intercept sensitive information (i.e., passive attack). The attacker may also choose to actively alter messages containing transaction data (i.e., active attack).

3.4.5. ASSETS

Assets are the reason that threats exist. Attacks always target the assets of an information systems. In order to get a better view on the threat landscape, it is important to first identify the assets of the Payleven payment platform. The assets are listed in Figure 3.3.

| | Assets | Description |
|----|------------------------|---|
| A1 | User Login Credentials | The credentials that a user will use to log into the Payleven app or dashboard. |
| A2 | Merchant Data | Personal information of the merchant account, such as name and bank account information. |
| A3 | Customer Data | Personal information of the customer. |
| A4 | Card Data | Information stored on or related to the banking card, such as bank account number, card holder name, expiration data etc. |
| A5 | PIN Data | Information exchanged between card and terminal, and card and bank server, such as the PIN and the EMV cryptograms. |
| A6 | Money | Money involved in transactions conducted by the Payleven payment platform. |
| A7 | Availability | The availability of the Payleven payment platform. The platform should be available 24 hours a day and should always be accessible by all Payleven users. |
| A8 | Login Session | The login session of a legitimate Payleven user to the Payleven app or dashboard. |
| A9 | Pairing Session | The session of a Bluetooth connection between the card reader and the smartphone. |

Table 3.3: Assets of the Payleven payment platform.

The assets are divided into three categories: (1) information related assets

(A1 until A5), (2) monetary related assets (A6), and (3) information availability related assets (A7 until A9). Money is, obviously, the most valuable asset of the Payleven payment platform. It forms the main motivation of any attack on the payment platform. The remaining assets are related to the information processed by the payment platform, and the system availability.

Other key assets are card and PIN data, since these may lead to monetary gain. The login credentials may also be of some certain significance for an attacker, however this depends fully on the capabilities of the login credentials (i.e., to what do they give access to).

The availability of the payment platform is also listed as an asset, since any disruption of it will cause financial damage, both for the merchant as for Payleven. Additionally, disrupting the login and pairing sessions will also lead to the absence of availability. Moreover, session hijacking may indirectly lead to monetary gain or information gathering.

3.4.6. ATTACKER TYPE

Two types of attackers can be defined: active and passive. An active attacker aims to alter data on the information system or data en route to the information system, whereas a passive attacker attempts to gather information on the system without affecting it. A passive attacker may also monitor the system for vulnerabilities. An attacker can have physical access to the system resources (e.g., card reader, smartphone, mobile application) or operate from a distance. Passive attackers often operate from a distance in order not to be noticed. In practice, an attacker can exhibit characteristics of both types; for example, an attacker may first scan the information system from a distance in order to discover vulnerabilities which then can be exploited to alter the behavior of the system.

Attacks can also be differentiated based on their origin. An authorized and trusted entity can initiate an *inside* attack and affect all components of the system. The inside attacker has legitimate access to the system resources and can easily alter system behavior in their advantage. An attacker from the *outside*, on the other hand, has no authorized access and gains access by breaking into the system. The definition of an inside attack can be redefined more specifically or differently in the context of the Payleven payment platform: an inside attacker is an attacker that sets up its own payment setup with the objective of conducting attacks. In other words, the merchant is the attacker. Correspondingly, an attack performed on a legitimate Payleven payment setup is considered to be initiated by an outside attacker. This can be the customer, but the attack can also be initiated by a third party. Note that the definition for an inside and outside attack differs slightly from the conventional definitions. According to the conventional definition [22], a customer could also be considered as a inside attacker since (1) it is part of the protocol and (2) it has authorized and legitimate access to the system resources (i.e., the card reader).

In the end, an attacker can play three roles namely merchant, customer and third party. Figure 3.4 sorts these three roles based on their capabilities and attack

power. The merchant is the most powerful attacker.

| | | |
|---|-------------|---|
| ① | Merchant | Has legitimate access to the card reader, Payleven, dashboard and probably the network. |
| ② | Customer | Has legitimate access to the card reader and the smart card. |
| ③ | Third party | Has no legitimate access. |

Table 3.4: Role of attacker in the payment platform sorted on capabilities and attack power where the merchant is the most powerful attacker.

3.4.7. THREAT CATEGORIZATION & ATTACKER GOAL

Threats can be categorized according to the STRIDE methodology [23]:

1. **Spoofing.** Masquerading as a legitimate user and falsifying data.
2. **Tampering.** Modification of persistent data or alteration of data en route.
3. **Repudiation.** Action that aims to perform illegal operations without leaving traces due to the system's defects.
4. **Information Disclosure.** Read data that one was not granted access to, or read data en route.
5. **Denial of Service.** Action aimed to disrupt the availability of the system.
6. **Elevation of Privilege.** Gaining unauthorized access to information or compromise the system.

The above-mentioned set of threat categories is very useful in the classification of the intentions or goals of an attacker. However, these categories only give a basic lower-level view on the goals of an attacker. In the end, the ultimate goal differs from system to system; and may embody multiple STRIDE threat categories. That is to say, each STRIDE threat category represents a mean or a set of means for achieving this ultimate goal.

In this research, the intentions of an attacker are redefined in the context of the Payleven payment platform. The intentions of an attacker can be classified into three groups:

1. **disruption;**
2. **monetary gain;**
3. and, **information gathering.**

Disruption can be induced through various methods; the most well-know is denial-of-service, as already listed as one of the threat categories. DoS attacks are performed to prevent legitimate use of a system of service. This can be achieved

by saturating the system resources, jamming the (wireless) connection or just by preventing a specific entity to access a system. Services can also be disrupted through data corruption or deletion.

Monetary gain is the most obvious goal of an attacker. It is important to realize that the purpose of information gathering in the long term is often monetary-related; sensitive user data can be (re)sold to criminal organizations or gathered to construct a user profile for future attacks. The intentions of an attacker are not clearly distinguishable and may be interdependent. For example, DoS attacks may indirectly have a monetary-related goal or contribute to the process of information gathering.

However, in the end, an attacker seeking monetary gain will focus on *direct* ways to steal money, be it by altering transaction exchange messages (*tampering*) or by faking transactions (*spoofing*), whereas an attacker with the aim of gathering information will focus on account and transaction data gathering (*information disclosure*).

As mentioned earlier, the final intention of an attacker may cover multiple STRIDE threat categories. Monetary gain, for example, can be achieved through altering the transaction amount or by repudiating a transaction. The Venn diagram in Figure 3.7 shows the attacker goals in the context of the Payleven payment platform and the means by which these goals can be achieved categorized in accordance with STRIDE.

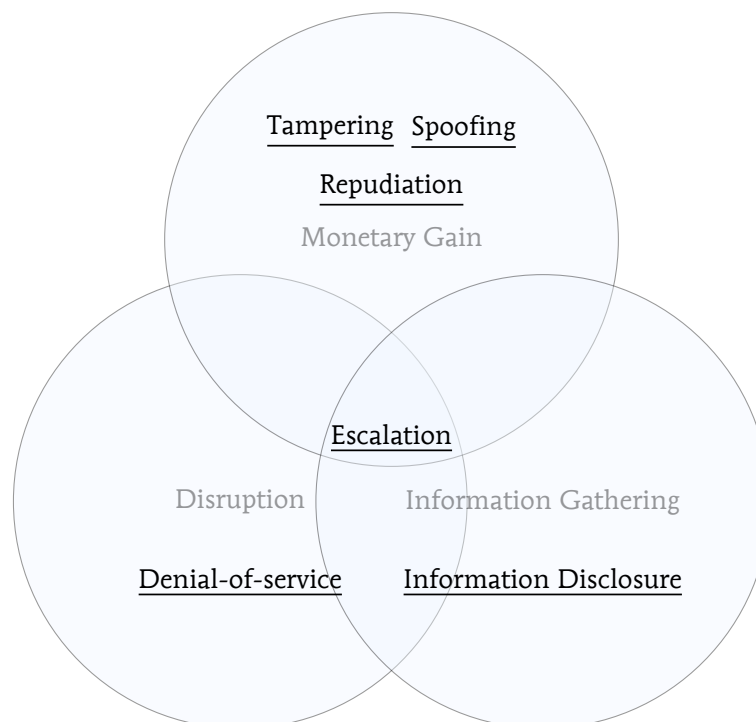


Figure 3.7: Venn diagram showing the attacker goals and the corresponding threats categories.

3.4.8. DATA FLOW DIAGRAM

The data flow diagram in Figure 3.8 shows how data moves within the Payleven payment platform. The visual representation of the application's data flow will help make the threat landscape more evident.

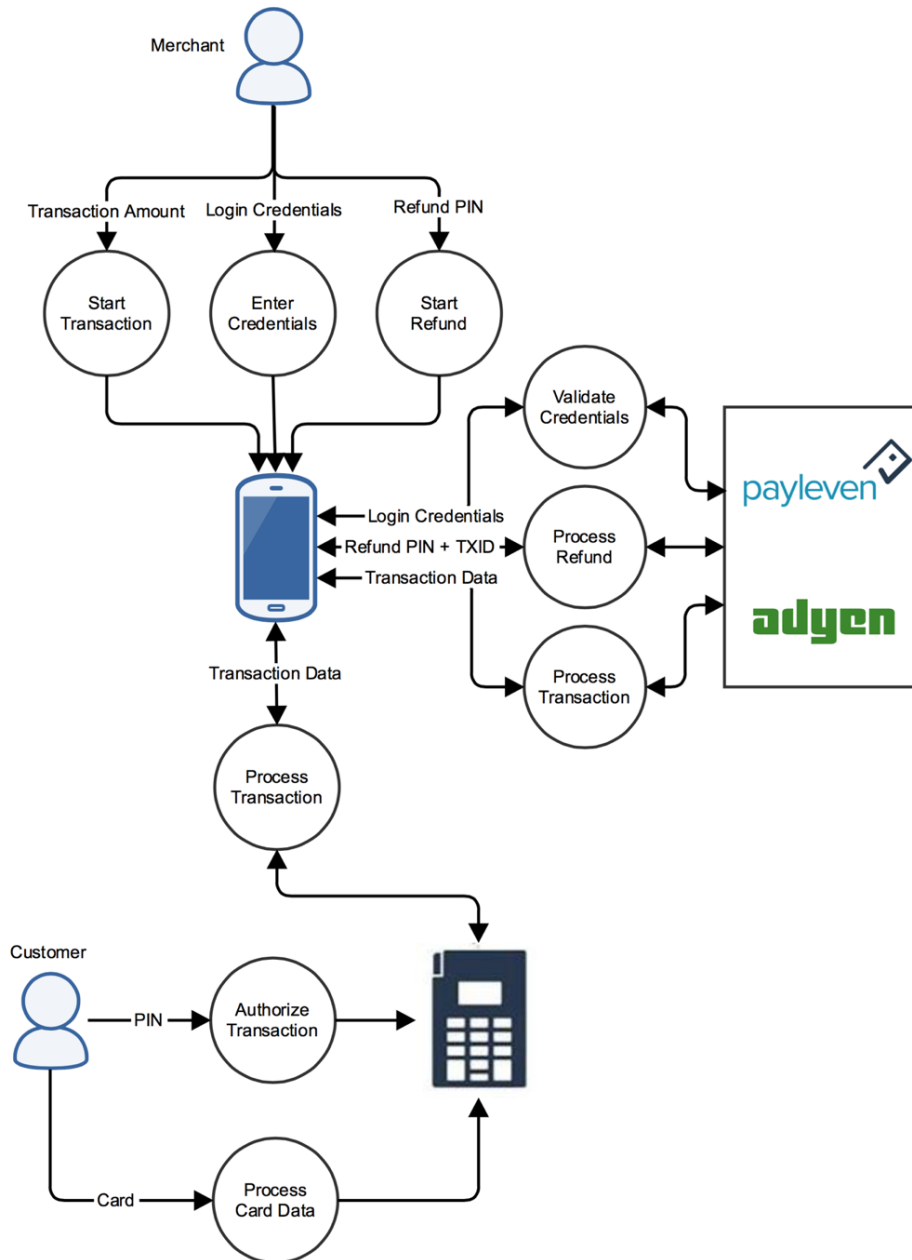


Figure 3.8: Data flow diagram demonstrating an outline on how data moves through the Payleven payment platform.

The diagram covers the most common use cases of the Payleven payment platform as described in Chapter 2; that is, the app login process, the transaction process and the refund process.

The merchant can interact with the Payleven payment platform in three ways: (1) by logging into the Payleven app, (2) by starting a transaction or (3) by conducting a refund. These three use cases are represented in Figure 3.8 respectively by the data flow processes ‘Enter Credentials’, ‘Start Transaction’ and ‘Start Refund’.

Each process handles a specific set of data:

- the app login process handles the Payleven login credentials;
- the transaction process handles the transaction amount;
- and, the refund process handles the merchant’s refund authorization PIN.

This data is then allocated by the Payleven app to the remaining two components of the payment platform; the Payleven back-end including the PSP (Adyen) and the Chip & PIN card reader. The transaction amount is then used by the Payleven app to initiate a transaction request and the associated transaction data is transferred to the card reader. The data related to the refund and login processes does not require the involvement or presence of the card reader and is sent via the Payleven app directly to the Payleven back-end to be processed.

The Payleven back-end comprises the following three data flow processes: (1) ‘Validate Credentials’, (2) ‘Process Refund’ and (3) ‘Process Transaction’. The first process validates the login credentials entered by the merchant in the process ‘Enter Credentials’ and grants the merchant access to the Payleven payment platform. The second data flow process concerns the handling of the refund request. For this, the process requires the refund authorization PIN entered by the merchant in the process ‘Start Refund’ and the transaction ID (TXID) of the transaction which is to be refunded. The transaction ID is provided by the Payleven app. The Payleven back-end is, together with the card reader responsible for the processing of the transaction data. The card reader and the Payleven back-end keep exchanging data via the Payleven app during the course of the transaction. In Figure 3.8, this is pointed out by the process ‘Process Transaction’ which appears twice in the diagram; between the app and the card reader, and between the app and the Payleven back-end.

The customer only has to interact with the card reader. The interaction consists of two parts: first, the customer inserts the banking card (‘Process Card Data’) into the card slot of the card reader, and lastly, enters the corresponding PIN (‘Authorize Transaction’) on the keypad of the card reader. The data that is processed here comprises the PIN code and card data.

3.4.9. POTENTIAL THREATS

This section gives an overview of the threats relevant for the Payleven payment platform. The threats can be identified based on:

- the attack points in Section 3.4.3;
- the attack vectors in Section 3.4.4;
- the assets listed in Table 3.3;
- the attacker goals described in Section 3.4.7;
- and, the data flow diagram shown in Figure 3.8.

Table 3.5 summarizes these threats for a clearer overview.

| | Threat |
|-----|---|
| T01 | Disclosure of sensitive information |
| T02 | Tampering transaction data |
| T03 | Faking transactions |
| T04 | Relaying or replaying transactions related requests |
| T05 | Repudiation of transactions |
| T06 | Denial-of-service |

Table 3.5: Threats relevant to the Payleven payment platform.

Threat T01 concerns the disclosure of transaction data, card data, the refund authorization PIN, but also refer to disclosure of login credentials, session tokens, and personal information on the merchant and customer. These data may be intercepted en route and in-air, or logged and captured (unintentionally) by the card reader and smartphone. The disclosure of sensitive information may harm the reputation of Payleven or the merchant, and could indirectly lead to financial damage.

Threat **T01** can be applied to several aspects of the payment platform:

- Wireless Bluetooth connection - As described in Section 3.3.1, monitoring a Bluetooth connection is practically impossible without professional hardware. Also, the fact that the Bluetooth connection is encrypted makes eavesdropping on the Bluetooth connection even more unlikely.
- Wireless network channel - Eavesdropping the wireless network communication channel is more likely - however also hardly feasible. Firstly, the WiFi (WPA2) and cellular data connection both provide security in form of encryption. In order to monitor the wireless communication, the attacker has to break this layer of encryption first, which is very unlikely. The best option then is to conduct an MITM-attack on the network channel. Section 3.3.2 described the methodology for such an attack as has been applied in this research. However, the feasibility of such an attack in a real-life situation is very arguable.
- Smartphone and card reader - Sensitive information could be extracted directly from the smartphone; for example, via smartphone malware or a rogue app. As regards the Chip & PIN card reader, information could be extracted from the reader via the method described in Section 3.3.4.

The question to be answered then is how serious this threat is. Information disclosure is attainable; however is the information disclosed of high value?

Threat T02 covers the dangers of tampered data within the payment platform. The modification of the transaction data or bank information may lead to financial loss or result indirectly in other types of damage. These modifications could be applied on the wireless channels or introduced on the devices self.

Threat T03 relates to the spoofing of transactions towards the (payment) back-end or the merchant by falsely signaling a successful transaction. For example, the smartphone may be misled into believing that the card reader has authorized a payment transaction. Also, the merchant or customer may also be tricked into believing that the transaction is successfully processed. This can be achieved by altering the transaction status message displayed on the card reader and smartphone. Spoofing can be executed, for example, on the smartphone by altering the behavior of the Payleven app or on the wireless communication channels (e.g., changing a NOK message to OK), or by faking web requests towards the back-end.

Threat T04 concerns the relay and replay of transaction related request. This threat is highly dependent on attacks based on protocol or design specific vulnerabilities. Its impact is high as it may lead (indirectly) to financial loss. Relay or replay attacks can be initiated by intercepting messages exchanged on the wireless channels (Bluetooth and WiFi/cellular) and resending them at a later stage to the Payleven and Adyen back-end. This threat does not cover EMV messages or cryptograms.

Threat T05 is about the repudiation of transactions. An actor of the Payleven payment platform may deny that a transaction actually has been conducted, which results in financial loss. It is also a threat if an actor cannot prove that a certain transaction has not occurred; that is, denying a fake transaction.

Threat T06 covers threats on the availability of the Payleven payment platform. Denial-of-service can be achieved by saturating the communication channels or rendering the devices unusable.

3.4.10. POTENTIAL ATTACK SCENARIOS

A proper security analysis requires the point of view of an attacker. This section therefore specifies several potential attack scenarios based on the threat landscape of the Payleven payment platform.

Attack scenario 1: modifying the transaction amount

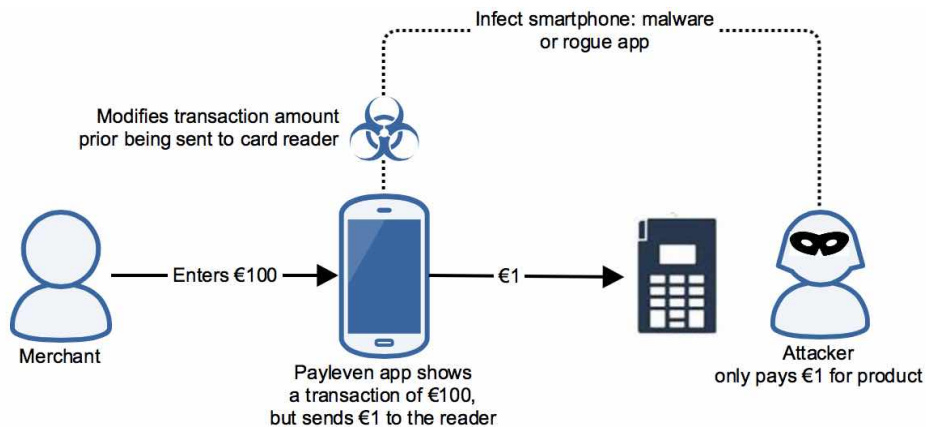


Figure 3.9: Attack scenario 1: modifying the transaction amount on the Payleven app.

Figure 3.9 gives a schematic overview of a attack scenario. The attacker takes the role of a malicious customer. The goal of the attacker is to buy products via the Payleven payment platform for the minimum transaction amount. The merchant initiates the payment transaction by entering the transaction amount in the Payleven app. The app then generates a transaction request and sends the request including the transaction amount to the Chip & PIN card reader. However, the smartphone is infected with a mobile malware that hijacks the Payleven app and changes the transaction amount to the minimum transaction amount of €1 before it is transmitted to the card reader. The card reader starts processing the payment transaction of €1.

This attack scenario is based on threat **T02**. The vulnerable point is the smartphone and/or Payleven app. This scenario assumes that the payment platform allows the modification of the transaction amount. Or in other words, it is expected that no detection mechanism is built into the Payleven payment platform - including the card reader and the back-end.

Attack scenario 2: modifying destination bank account

The second attack scenario involves an outside attacker. The goal of the outside attack is monetary gain. The attack scenario in Figure 3.10 describes an attack in which all the money associated with the transactions conducted on the

Payleven payment platform is diverted to the attacker. This can be achieved by changing the original destination bank account into the attacker's account prior being forwarded to the Chip & PIN card reader. It should be noted that the card reader needs to be fed with the malicious bank account details in an early stage. Changing the bank account at a later stage will not work as will become apparent in Chapter 4.5: the card reader exchanges transaction data via an encrypted channel. However, the authentication that precedes the encrypted channel may be a weak spot.

The destination bank account can be changed at two points in the platform: (1) before being sent to the Payleven app - after all, the Payleven does not have this information during first use; (2) on the smartphone via malware or a rogue variant of Payleven app; (3) on the Bluetooth channel. The latter is not feasible as pointed out in Section 3.3.1. The other two options belong to the possibilities. The attack scenario shown below visualizes an attack in which the smartphone is targeted with malware. In the end, money meant for the merchant is diverted to the attacker.

Although not pointed out above, the personal dashboard may also be an attack point for the destination bank account. The dashboard also lists the merchant's bank account. Unfortunately, changing the bank account via the personal dashboard triggers only more paperwork. The merchant is then expected to submit transcripts from the newly changed bank account stating the merchant's (company) name. Clearly, this will not work if an attacker does this.

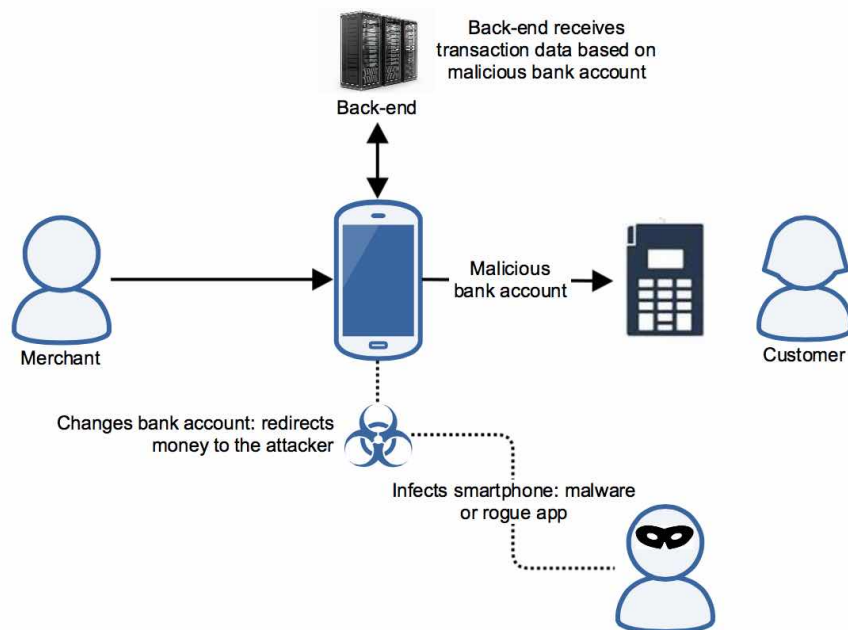


Figure 3.10: Attack scenario 2: the attacker modifies the original destination bank account by infecting the smartphone. Money is then diverted to the attacker.

This attack scenario is based on threat **T02**. The vulnerable point is the smartphone and/or Payleven app. Moreover, with the attack scenario it is assumed that the implementation of the Payleven payment platform will allow the modification of the destination bank account. However, it should be expected that undesirable changes will be detected by the payment system.

Attack scenario 3: unauthorized refund requests

Figure 3.11 visualizes attack scenario 3. This scenario is aimed at the refund process. In this attack, a malicious customer triggers unauthorized refund requests. In order to accomplish the attack, an attacker would need to be in possession of the refund authorization PIN and valid Payleven login credentials. The attacker is already in possession of the transaction ID (TXID); it is already noted on the invoice. As regards the refund authorization PIN, the attacker could intercept or extract the PIN from the payment platform. Moreover, since it is only 4 digits, brute-forcing is also among the possibilities. The login credentials can be stolen via an MITM-attack, phishing or mobile malware.

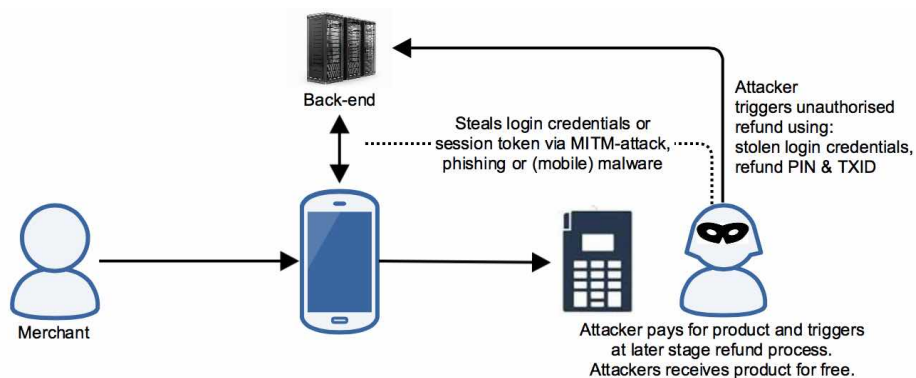


Figure 3.11: Attack scenario 3: the attacker triggers an illegitimate refund request.

This attack scenario is based on threat **T01**. Threat **T04** could also be applied as it would be possible to intercept a random refund request and re-use it at a later stage; only the TXID needs to be changed, assuming that the session is still valid. Moreover, this scenario depends on several vulnerabilities; e.g., how well protected are the refund authorization PIN code and the corresponding login credentials? What about session expiration management?

Attack scenario 4: fake successful transaction message

Figure 3.12 describes an attack scenario in which a failed transaction is falsely signaled as successful. The transaction is canceled by the attacker and is never processed by the card reader. However, malware or a rogue variant of the Payleven app could cause the smartphone to display a success message. The merchant is

then tricked into believing that the transaction was processed successful. The attacker then gets away with a free product.

This attack scenario covers threat **T02**.

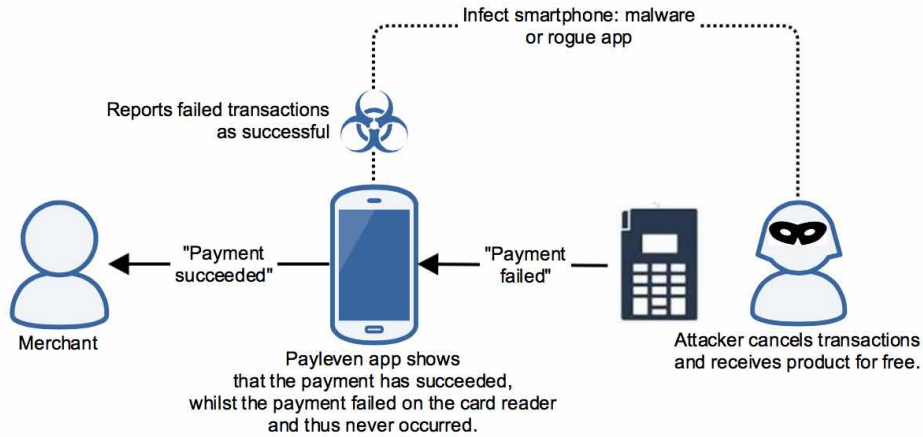


Figure 3.12: Attack scenario 4: the infected smartphone is forced to signal failed or canceled transactions as successful. As a result, the merchant is tricked into believing that failed transactions were processed successfully.

Attack scenario 5: malicious card reader

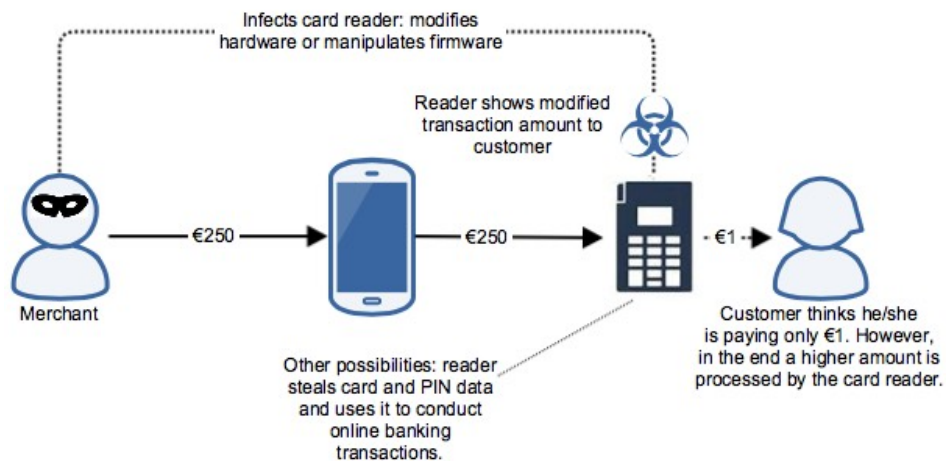


Figure 3.13: Attack scenario 5: the malicious merchant infects the card reader and steals money from customer.

Figure 3.13 shows a somewhat different attack approach. Whereas the previous attack scenarios had the merchant as victim, this scenario has not. In this

scenario the merchant is the malicious attacker. The attacker sets up an own payment platform and acts as a legitimate merchant. Moreover, instead of focusing on the Payleven app and the smartphone, the attacker targets the Payleven Chip & PIN card reader.

The card reader has been modified in such way, be it via hardware or software modifications, that the transaction amount displayed on the card reader and shown to the customer differs from the actual amount processed by the card reader and sent to the Adyen back-end. This way the malicious merchant can steal money from the customer by and processing payments with a substantial higher transaction amount while the customer

This attack implies threat **T02**. Furthermore, the attack is highly dependent on the accessibility of the card reader. The reader must cover serious vulnerabilities.

4

INTERNET NETWORK ANALYSIS

This chapter analyzes the Internet network traffic of the Payleven payment platform. First, Section 4.1 provides the reader with a global overview of the network traffic during the five use cases of the payment platform. Sections 4.2 till 4.6 then give detailed descriptions of the individual web requests per use case. Chapter 5 complements the impression of the data flow of the payment platform by examining the Bluetooth traffic.

4.1. GLOBAL OVERVIEW OF THE NETWORK TRAFFIC

This section gives the global findings based on a first view analysis of the network traffic and presents an overview of the network connections made by the Payleven payment platform. The network analysis covers the traffic under the following five use cases:

1. app login;
2. app initialization;
3. Bluetooth pairing process;
4. transaction;
5. and, refund.

The network traffic during the registration with Payleven (i.e., use case 1 in Section 2.1.1) was not investigated, since it does not form an essential part of the use cycle of the payment platform. Furthermore, it does not involve the Payleven app.

During the overall use of the payment platform, the Payleven app connected to the following four web domains:

- **apiproxy.payleven.de**, 62.138.111.121, Germany;
- **fapi.payleven.de**, 62.138.111.121, Germany;
- **ca-live.adyen.com**, 82.199.90.174, Netherlands;
- **pal-live.adyen.com**, 82.199.90.182 and 91.212.42.182, Netherlands.

Figure 4.1 gives a basic overview of the communication channels of the Payleven payment platform. The network connections to Payleven and Adyen are both TLS-protected. Furthermore, within the TLS-protected connection with Adyen lurks an additional security layer. Application data sent to Adyen is subjected to additional encryption on the application-layer. The same encrypted data is also observed in the Bluetooth connection between the smartphone and the Payleven Chip & PIN card reader. In conclusion, the application-level encryption is used to establish a secure tunnel between the Adyen payment servers and the card reader. The smartphone and the Payleven probably do not have access to the encrypted data. The smartphone primarily acts as a hatch for the encrypted data. Payleven, on the other hand, only relies on the TLS and the Bluetooth encryption; i.e., data intended for Payleven is not subjected to additional encryption.

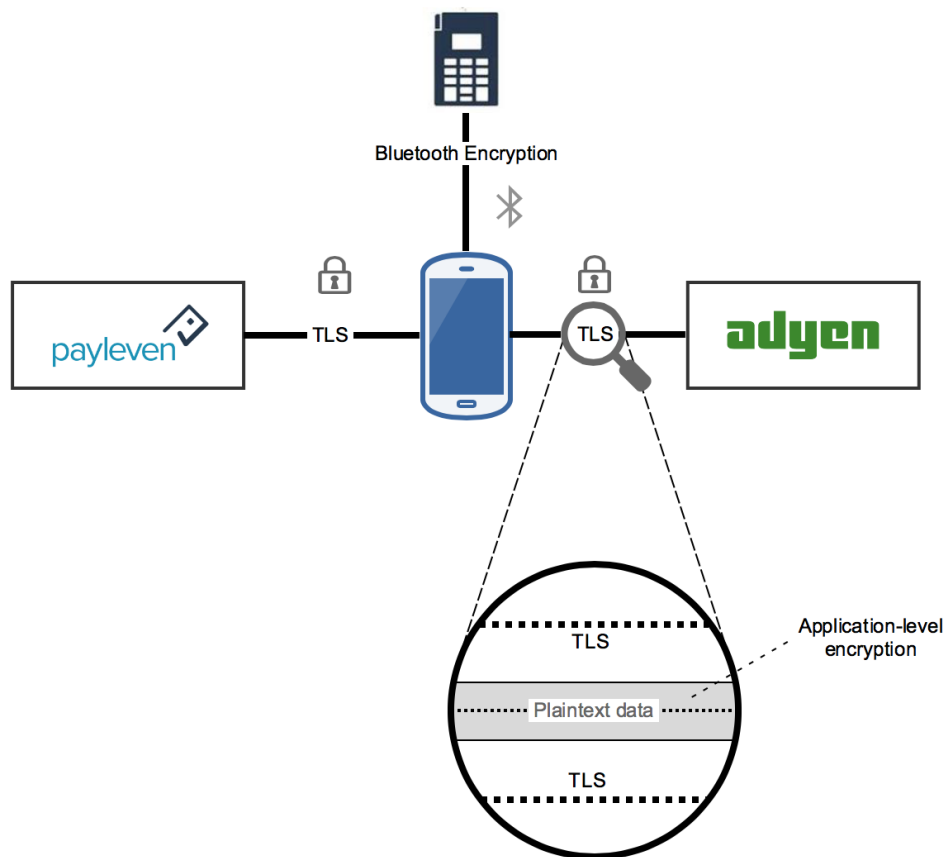


Figure 4.1: Basic overview of the network connections of the Payleven payment platform.

It should also be noted that not all application data sent to Adyen is subjected to application-level encryption, as will become clear in the remainder of this chapter. The encrypted data is mainly observed during the transaction phase. Since it is also observed on the Bluetooth channel, the encrypted data probably holds sensitive transaction and card information generated on the card reader and the banking card. Given these points, the card reader must therefore also be responsible for the application-level encryption and hold the corresponding encryption materials (i.e., encryption keys and certificates).

A first view analysis of the network traffic resulted in the outline shown in Figure 4.2. The outline summarizes the information exchange of the Payleven payment platform per web domain.

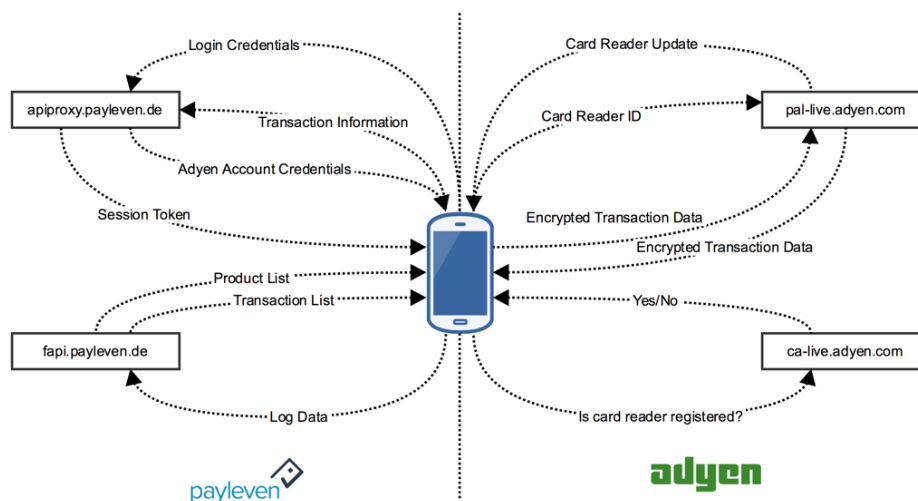


Figure 4.2: Overview of the information exchange between the Payleven app and the four web domains.

From the outline in Figure 4.2 it can be seen that each web domain serves a different purpose.

- **apiproxy.payleven.de** (apiproxy) is responsible for the authentication of the merchant during the app login. The apiproxy web server provides the Payleven app with a session token for the authentication of further messages within that same session. Also, further information on payment transactions such as transaction amount, payment method and transaction ID are communicated to Payleven via the apiproxy web server.
- **fapi.payleven.de** (fapi) is primarily used for the logging of data and app events. The fapi web server also provides the Payleven app with an up-to-date list of previous transactions and current products (product shelf).
- **ca-live.adyen.com** (ca-live) is responsible for the registration of the Payleven Chip & PIN card and the Payleven app with Adyen.

- **pal-live.adyen.com** (pal-live) is responsible for the processing of payment transactions. This is based on the observation that a large portion of network traffic with the pal-live web servers is obfuscated. In other words, the transaction data is being protected. Furthermore, communication to the pal-live server is primarily observed during (the establishment of) transactions.

Table 4.1 shows per use cases the web domains that are requested by the Payleven app. As can be seen from the table, no requests are sent towards Adyen during the app login and refund use cases.

| | Use case | Requested web domains |
|---|---------------------------|---|
| 1 | App login | <ul style="list-style-type: none"> • apiproxy.payleven • fapi.payleven |
| 2 | App initialization | <ul style="list-style-type: none"> • ca-live.adyen • fapi.payleven |
| 3 | Bluetooth pairing process | <ul style="list-style-type: none"> • apiproxy.payleven • ca-live.adyen • pal-live.adyen • fapi.payleven |
| 4 | Transaction | <ul style="list-style-type: none"> • apiproxy.payleven • ca-live.adyen • pal-live.adyen • fapi.payleven |
| 5 | Refund | <ul style="list-style-type: none"> • apiproxy.payleven • fapi.payleven |

Table 4.1: The requested web domains per use case.

Sections 4.2 to 4.6 give an analysis of the network traffic per use case. The network analysis delves deeper into the message sequence charts described in Section 2.4 and gives a more detailed view on the network traffic by outlining each individual web request.

4.2. NETWORK TRAFFIC DURING APP LOGIN

The login phase consists of two simple activities: (1) the merchant enters his or her Payleven login credentials into the Payleven app and (2) presses the 'Sign in' button. Once the 'Sign in' button is pressed in, the Payleven app connects to the following four web addresses in the following order.

| | | |
|-----|---------------|--|
| P1a | URL Method | apiproxy.payleven.de/api/user/login POST |
| P2a | URL Method | fapi.payleven.de/fapi/v1/shelf/merchantID?lastUpdate=YYYY-MM-DDTUU:MM:SS GET |
| P2b | URL Method | fapi.payleven.de/fapi/v3/transaction/list?pageSize=100&page=0 GET |
| P2c | URL Method | fapi.payleven.de/fapi/v1/logs POST |

Table 4.2: Web requests during the app login.

Table 4.3 lists all web requests initiated by the Payleven app during the app login. Figure 4.3 processes these web requests into a low-level message sequence chart (MSC).

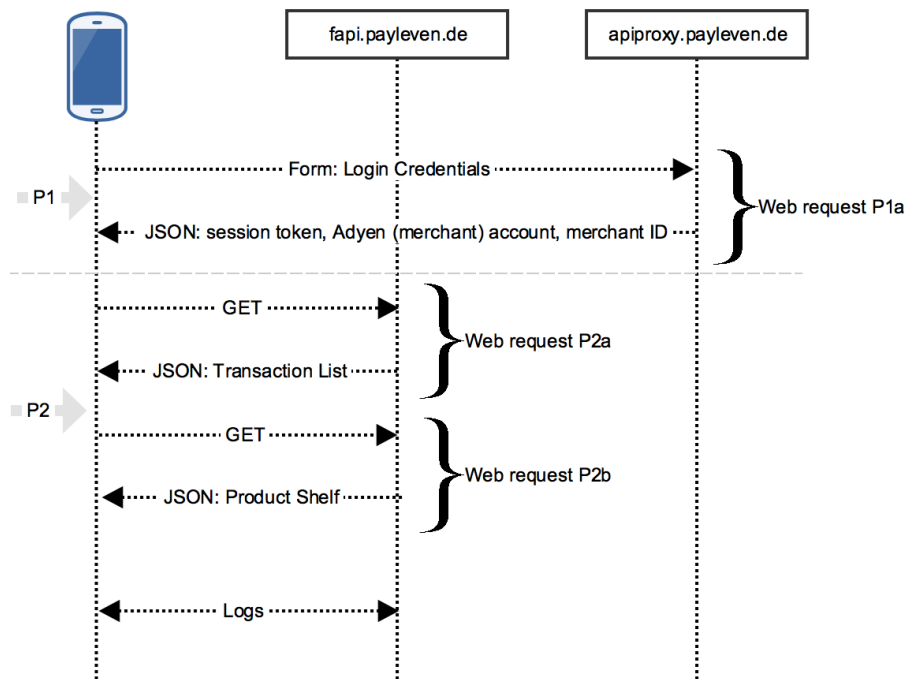


Figure 4.3: Low-level message sequence chart of the Internet network traffic during the app login.

In contrast to the high-level MSC 2.15 described in Section 2.4.1, MSC 4.3 focuses only on the network part. Furthermore, the message exchange arrows **P1** and **P2** from the high-level MSC 2.15 recur in this MSC; P1 and P2 are set out here in more detail.

4.2.1. WEB REQUEST P1A

The first web request contains a POST multipart/form-data that holds the email address used for the Payleven account, the corresponding password, and the boolean parameter `create_token`, which is set to 1 (i.e., true). The header of the request contains information on the used device, the Payleven app version and the location coordinates. Part of the HTTP header is listed in Listing 4.1.

```

3 [...]
4 X-App-Version: 2.28.0
5 X-API-Version: 89
6 Accept-Language: en
7 X-Device: samsung/Galaxy Nexus
8 X-Device-Os: Android
9 X-Device-Model: samsung/Galaxy Nexus
10 X-Resolution: 360.0x592.0
11 Accept: application/json
12 X-Coordinates: 52.10774505045265, 5.088596334680915
13 Content-Length: 585
14 Content-Type: multipart/form-data; boundary=GaBIMA_4EBs0_kOMpMx4EB-
    JupjjMASrNVb0c
15 [...]

```

Listing 4.1: Part of HTTP header of the POST request to `apiproxy.payleven.de/api/user/login`.

The location coordinates are sent to Payleven to enforce the location-restriction requirements. The platform can only be used in the country the account is registered in.

The response to the login request contains JSON-formatted data. The JSON is listed in the message body and is sent in plaintext; i.e., no encoding scheme is used.

The JSON data set consists of three parent JSON objects: `meta`, `response` and `status`. The `meta` object listed the requested session token. The session token is shown in Listing 4.2.

```

4 "session_token": {
5   "content": "LLO3wXCRtG2EDr5PAJDUGZIEVLNSMcz_3lPg_nW4WT1PJUZJ"
6 }

```

Listing 4.2: The requested session token.

The response object included primarily some trivial information on: the company, such as proprietor name, merchant name, address, account and contact details; the registered and linked card readers; and several other rule sets concerning the Payleven payment platform.

The most interesting part of the response from Payleven concerned the payment service provider (PSP). Listing 4.3 shows the contents of the object `adyen` within the JSON response.

```
4
5 "adyen": {
6     "adyen_account": "653Klu*****",
7     "adyen_company": "Company.PaylevenNL",
8     "adyen_devices": [
9         "Shuttle-016303651"
10    ],
11    "adyen_passwd": "*****",
12    "adyen_user": "928***",
13    "devices": {
14        "Shuttle-016303651": {
15            [...]

```

Listing 4.3: Adyen account information in the POST response from `apiproxy.payleven.de/api/user/login`

The `adyen` object listed the Adyen merchant account ('653Klu*****', 11 digits), the corresponding Adyen password (12 digits), the Adyen user code ('928***', 6-digit numerical code) and the card reader(s) registered under that specific Adyen account. This information regarding the Adyen account is redacted due to privacy reasons. The `adyen_company` object refers to the company for which Adyen provides their services - that is, in this case Payleven.

These account details are probably generated by Adyen and communicated to Payleven during the registration phase. The Adyen merchant account, the Adyen user code and the corresponding password are used later on by the Payleven app to authenticate to the Adyen web servers (`ca-live` and `pal-live`).

It should also be noted that these authentication credentials are communicated to the Payleven app at each and every login request. This is serious matter as these credentials are highly confidential. After all, they authenticate the merchant towards the Adyen back-end. Moreover, as will become clear later on during the analysis of the Bluetooth traffic, the merchant account is the only reference to the merchant that is communicated to the card reader. In other words, the merchant account probably determines the destination bank account.

4.2.2. WEB REQUEST P2A

The next web request concerns a GET request destined for `fapi.payleven.de`. The path name suggest that the goal of the GET request is to retrieve the latest product shelf for the merchant with ID '`merchantID`'. The merchant ID is a 6 digit numerical code and is provided by Payleven in the response to web request 1. The merchant ID is probably related to the Adyen merchant account since both start with '653'. Please note that the merchant ID (6-digit numerical code) is not the same as the Adyen merchant account.

As can be seen in Listing 4.4, the HTTP header of the GET request included the session token to authenticate to the Payleven server.

```
12 Authorization: Payleven LLO3wXCrtG2EDr5PAJDUGZIEVLNSMcz_3lPg_nW4WT1PJUzJ
```

Listing 4.4: Authorization header containing the session token.

The response to the GET request contains an empty product list, since no products were configured in the Payleven account.

4.2.3. WEB REQUEST P2B

The next GET request is aimed at fetching the latest transaction list from the Payleven web servers. The response contains information on every transaction conducted before with the specific Payleven account and was therefore fairly large.

4.2.4. LOGS 1

The last web request of the app login phase sent JSON-formatted log data to the Payleven web servers. The log data is shown in Listing 4.5.

```
17 "events": [
18   {
19     "data": {
20       "method": "GET",
21       "network_type": "WIFI",
22       "request_url": "https://fapi.payleven.de//fapi/v3/transaction
23       /list",
24       "time_interval": "0.96200"
25     },
26     "event": "TurnAroundTime",
27     "merchantId": "653***",
28     "paymentCountry": "NL",
29     "timestamp": "2015-08-07T07:38:32.032"
30   }
31 ]
```

Listing 4.5: JSON-formatted log data.

The log data contained information on the turn around time of web request P2b.

Interestingly, these log requests always come in twofold since the first request attempts always fail. The access was denied by the Payleven web server.

```
9 {
10   "code": 401,
11   "message": "Access is denied",
12   "status": "NOK"
13 }
```

Listing 4.6: Response to the failed log request.

The second request attempt is more successful as can be seen in Listing 4.7.

```
10 {
11   "code": 0,
12   "status": "OK"
13 }
```

Listing 4.7: Response to the second request attempt. Access has been granted.

A comparison of the two attempts reveals that the second request includes an authorization line in the header containing authentication credentials in Base64, whereas the first request lacked these credentials, which thus explains why the first attempt failed. The authorization header indicates the use of the basic access authentication method and according to this method, the authorization header line contains the Base64 encoding of '[username]:[password]'. Part of the authorization header line is shown in Listing 4.11. Please note that the password is redacted.

```
16 Authorization: Basic bG9nZnJvbnRlbnQ6*****
```

Listing 4.8: Authorization header line containing authentication credentials in Base64. The Base64 encoded password is redacted.

Decoding the Base64 string reveals the authentication credentials for the Payleven log server. These credentials (username: password) are listed in Listing 4.9.

```
0 logfrontend:*****
```

Listing 4.9: Authentication credentials for `fapi.payleven.de`. The password is redacted.

Why the first attempt failed is not clear as the Payleven app was already in the possession of these authentication credentials.

4.3. NETWORK TRAFFIC DURING APP INITIALIZATION

This section describes the network traffic during the Payleven app initialization (i.e., registration for first use). This process is described in Section 2.3.2. The Payleven app requests the following web addresses in the following order:

| | | |
|-------|--------|---|
| A1a | URL | <code>https://ca-live.adyen.com/ca/servlet/soap /PosRegistration/v3</code> |
| | Method | POST |
| Log 2 | URL | <code>https://fapi.payleven.de/fapi/v1/logs</code> |
| | Method | POST |

Table 4.3: The web requests during the app initialization.

Figure 4.4 presents the MSC of the network traffic during app initialization.

4.3.1. WEB REQUEST A1A

The first web request during the app initialization phase is a POST request to Adyen and contains the SOAP message `registerApp`, which lists the app ID and the Adyen merchant account. SOAP is a protocol specification for the exchange of structured information. It uses the XML information set for the message format.

Web request A1a is the first web request destined for Adyen. The web request authenticates with the following authorization header:

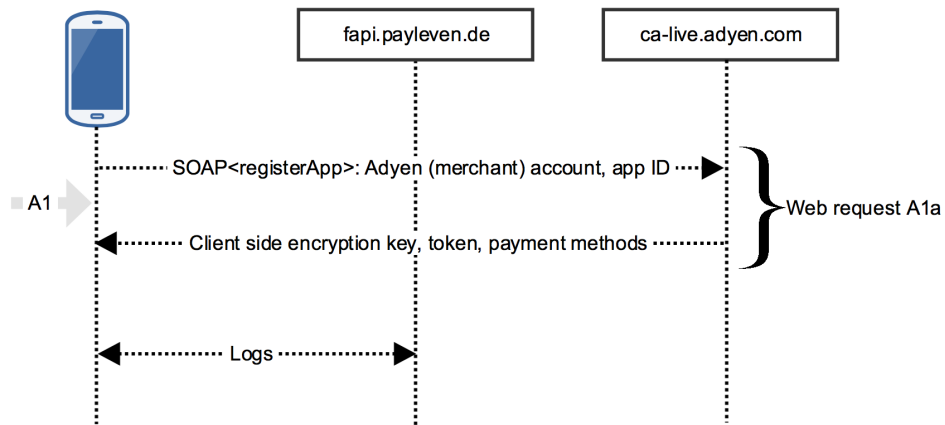


Figure 4.4: Low-level message sequence chart of the network traffic during app initialization.

```
2 authorization: Basic OTI4QENvbXBhbkuTWVY2hhbnRBY2NvdW50LjY1Mzo=
```

Listing 4.10: Authorization header line containing authentication credentials in Base64. Please note that the header line has been modified such that the password and other sensitive information is not revealed.

Decoding the authorization header line results in the following:

```
2 928***@Company.MerchantAccount.653Klu*****:*****
```

Listing 4.11: The Base64 decoded authorization header line.

A further look at the authorization header reveals some interesting information regarding the composition of the header line. The authorization line is constructed of the Adyen user code ('928***') and the Adyen merchant account ('653*****'). These details are provided by Payleven in the response to web request P1a (i.e., the login request). The merchant account is redacted due to privacy reasons. The same applies for the corresponding Adyen password, which follows after the ':'.

The contents of the SOAP message `registerApp` hold the Adyen merchant account and the app ID. The app ID is the identifier of the Payleven app and is freshly generated with any new installation of the app. Clearing the app's data via the smartphone settings menu results in the generation of a new app ID. The Payleven app needs to be initialized again after clearing the app's data.

```
13 <n0:Body>
14   <n1:registerApp xmlns:n1="http://posregistration.services.adyen.com">
15     <n1:request>
16       <n1:merchantAccount>653Klu****</n1:merchantAccount>
17       <n1:appId>658d85ff-3ee2-4b49-89d1-117ace6746f3</n1:appId>
18     </n1:request>
19   </n1:registerApp>
20 </n0:Body>
```

Listing 4.12: The SOAP message in web request A1a.

Subsequently, the response to the `registerApp` SOAP message listed:

- the associated Adyen merchant account;
- a public client side encryption key consisting of the exponent and the modulo (4096 bit);
- a list of supported currencies (only EUR) and payment methods (e.g., VISA, Mastercard etc.);
- the `registerApp` status code which is set to 'Registered';
- and a token.

As regards the client side encryption key, this is an interesting discovery. However, as will be pointed out in Section 6.1.2 this key is never used in the Payleven payment platform. It is used for the processing of Mail Order/Telephone Order (MOTO) transactions. The Payleven payment platform does not cover MOTO payment.

The other interesting finding is the token which is included at the end of the response message. The token is used by the Payleven app to authenticate to the Adyen back-end together with the authorization header line as encountered in this web request. The composition of the token is discussed in more detail in web request A2a in Section 4.4.2.

4.3.2. LOG 2

The next web requests concerns log request of two events. These two log events in Log 2 were cover the app registration. The events are listed in Table 4.4.

| | Log Event |
|---|-------------------------|
| 1 | RegisterAppWithAdyenLib |
| 2 | AdyenRegisterApp |

Table 4.4: Log events from Log request 2.

4.4. NETWORK TRAFFIC DURING BLUETOOTH PAIRING PROCESS

This section describes the network traffic during the Bluetooth pairing process. During the pairing process, the Payleven app sends the following web requests:

| | | |
|-------|---------------|---|
| P3a | URL Method | apiproxy.payleven.de/api/transaction/register POST |
| A2a | URL Method | ca-live.adyen.com/ca/servlet/soap/PosRegistration/v3 POST |
| A2b | URL Method | pal-live.adyen.com/pal/servlet/soap/PosRegistrationSync/v4 POST |
| Log 3 | URL Method | (2x)fapi.payleven.de/fapi/v1/logs POST |
| A2c | URL Method | pal-live.adyen.com/pal/adaptor/batchsync.proto POST |

Table 4.5: Web requests during the Bluetooth pairing process.

Figure 4.5 processes the above listed web requests in an MSC. The sequence chart zooms in into the P3 and A2 message exchange arrows from the high-level MSC 2.16.

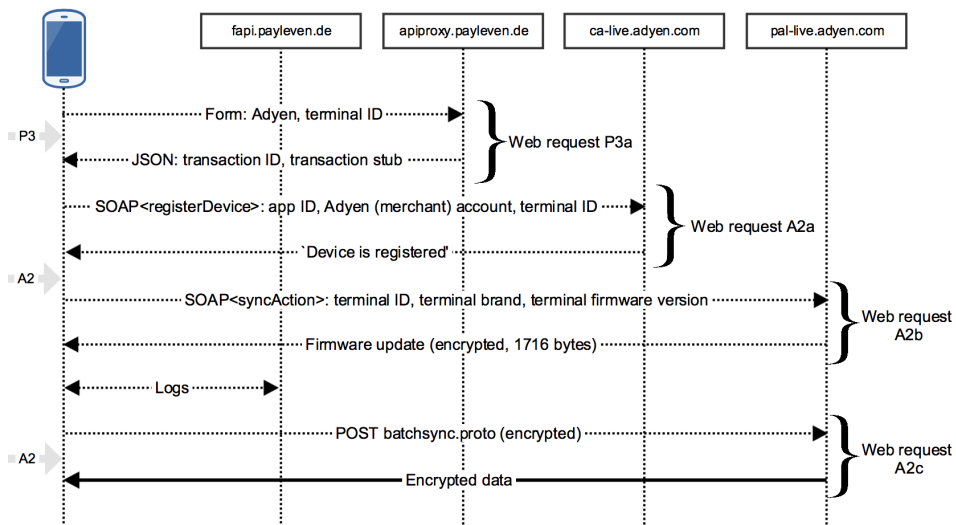


Figure 4.5: Low-level message sequence chart of the network traffic during the Bluetooth pairing process.

4.4.1. WEB REQUEST P3A

The first web request in the Bluetooth pairing process is directed to apiproxy web

server and contains a POST form that lists the payment service provider ('Adyen') and the card reader (terminal) ID ('Shuttle-016303651').

The response from Payleven contains JSON-formatted data and is structured in the same manner as previous JSON-messages. The message is composed of three parent JSON-objects: `meta`, `response` and `status`. The `meta` object holds the Payleven session token. The contents of object `response` are shown in Listing 4.13.

```

17 "response":
18 {
19   "creation_date": "2015-08-07 14:50:20",
20   "creation_date_utc": "2015-08-07T12:50:20+00:00",
21   "id_intpay_transaction": "52698677",
22   "refundable": false,
23   "transaction_stub": "8308b9bf71d57726a0a1"
24 },

```

Listing 4.13: The contents of the `response` object.

| JSON Object | Value |
|------------------------------------|----------------------|
| <code>id_intpay_transaction</code> | 52698677 |
| <code>transaction_stub</code> | 8308b9bf71d57726a0a1 |

Table 4.6: The 'intpay' transaction ID and the transaction stub.

The apiproxy web server provides the Payleven app with a transaction stub (20 bytes) and a 8-digit numerical transaction *intpay* ID.

4.4.2. WEB REQUEST A2A

The next request is destined for Adyen. The first web request attempt to `ca-live.adyen.com` fails however due to missing authentication credentials. This has been observed earlier during the log requests. Listing 4.14 and 4.15 respectively show the header of the first request attempt and its response from `ca-live.adyen.com`. The POST request clearly misses an authorization header field. As a result, the response gives an HTTP 401 status code.

```

1 POST /ca/servlet/soap/PosRegistration/v3 HTTP/1.1
2 User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.3; Galaxy Nexus Build/JWR66Y
   )
3 Host: ca-live.adyen.com
4 Connection: Keep-Alive
5 Accept-Encoding: gzip
6 Content-Type: application/x-www-form-urlencoded
7 Content-Length: 0

```

Listing 4.14: First web request attempt to `ca-live.adyen.com`.

```

1 HTTP/1.1 401 Unauthorized
2 Date: Fri, 07 Aug 2015 12:50:21 GMT
3 [...]

```

Listing 4.15: Response from Adyen to first web request attempt.

The second attempt does contain an authorization header field as can be seen in Listing 4.16. Additionally, the header also lists an authentication token (*jaas-token*). This token has been transmitted to the Payleven app in the response to web request A1a.

```

1 POST /ca/servlet/soap/PosRegistration/v3 HTTP/1.1
2 authorization: Basic OTI4QENvbXBhbkuTWVYy2hhbnRBY2NvdW50LjY1Mzo=
3 M0tsdU1lZG9QT1M6
4 jaastoken:
      NjU4ZDg1ZmYtM2VlMi00YjQ5LTg5ZDEtMTE3YWw1Njc0NmYzXk1xcUd1aFQ5bmFNcDQ2UW1j
5
6 dlaFQ5bmFNcDQ2Qmc=
7 SOAPAction: https://ca-live.adyen.com/ca/servlet/soap/PosRegistration/v3
8 content-type: text/xml
9 [...]

```

Listing 4.16: Header of request to ca-live.adyen.com including an authorization header field.

A further analysis of the jaastoken reveals that it is Base64 encoded and that it is constructed of the app ID and a 17-digit alphanumeric code. The jaastoken and thus the code are generated and provided by Adyen in the app initialization phase after the app ID has been forwarded by the Payleven app in web request A1a. Furthermore, the jaastoken is different for each new app initialization. After all, the app ID is different for each app initialization.

```

1 NjU4ZDg1ZmYtM2VlMi00YjQ5LTg5ZDEtMTE3YWw1Njc0NmYzXk1xcUd1aFQ5bmFNcDQ2UW1j
2 \\/
3 Base64 decoding
4 \\/
5 658d85ff-3ee2-4b49-89d1-117ace6746f3^MqqGuhT9naMp46Qmc

```

Listing 4.17: Base64 decoding of the jaastoken

In view of the fact that the jaastoken is generated based on the app ID and probably the merchant account and/or merchant code, it may be possible to register the app ID under a different merchant account. However, it is unknown what kind of impact this would have as it was not possible to test this without a second merchant account.

The content type header field also differentiates from the one listed in Listing 4.14: it states 'text/xml' instead of 'application/x-www-form-urlencoded'. The second request namely contains data formatted according to the SOAP protocol.

```

1 <?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
2 <n0:Envelope xmlns:n0="http://schemas.xmlsoap.org/soap/envelope/" xmlns:
   xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org
   /2001/XMLSchema-instance">
3 <n0:Body>
4 <n1:registerDevice xmlns:n1="http://posregistration.services.adyen.com">
5 <n1:request>
6 <n1:appId>658d85ff-3ee2-4b49-89d1-117ace6746f3</n1:appId>
7 <n1:merchantAccount>653Klu*****</n1:merchantAccount>
8 <n1:terminalId>Shuttle-016303651</n1:terminalId>
9 <n1:terminalSerialNumber>016-303651</n1:terminalSerialNumber>
10 <n1:terminalType>Shuttle</n1:terminalType>
11 <n1:terminalApiVersion>adyen-pl-v1_32p7</n1:terminalApiVersion>

```

```

12     </n1:request>
13 </n1:registerDevice>
14 </n0:Body>
15 </n0:Envelope>

```

Listing 4.18: Contents of web request A2a to `ca-live.adyen.com`. The contents cover a SOAP message.

Listing 4.18 shows the SOAP message directed to `ca-live.adyen.com`. The body of the message is comprised by the `Body` tags. The next tag after the `Body` tags describes goal of the SOAP message. That is, in this case, `registerDevice`. Furthermore, the SOAP message can represent a request or a response. This is indicated by the next tag after `registerDevice`. Since this SOAP message is part of a POST request, it will include `request` tags. These tags then comprise the actual information of the message. The information is composed from a set of items: (1) `appId`, (2) `merchantAccount`, (3) `terminalId`, (4) `terminalSerialNumber`, (5) `terminalType`, and (6) `terminalApiVersion`. Please note that card reader and terminal are used interchangeably throughout this research.

In the end, the SOAP request provides information regarding the Payleven app, the card reader (terminal) and the merchant account. Please note that the merchant account is the same as listed in Listing 4.3 under `adyen_account`. The terminal ID 'Shuttle-016303651' is built up from the reader type and the reader's serial number.

The SOAP response to the above-mentioned request is listed below. The envelope and body tags are omitted.

```

1 <n1:registerDeviceResponse>
2   <n1:response>
3     <n1:fleetCode>653*****POS-ALL</n1:fleetCode>
4     <n1:registerDeviceStatus>AlreadyRegistered
5     </n1:registerDeviceStatus>
6   </n1:response>
7 </n1:registerDeviceResponse>

```

Listing 4.19: Contents of the SOAP response from `ca-live.adyen.com`.

The response to `registerDevice` states that the terminal is already registered at Adyen. That is to say, Adyen probably verifies whether the card reader is registered under or linked to the merchant account stated in web request A2a.

4.4.3. WEB REQUEST A2B

The next web request is directed to `pal-live.adyen.com`. The contents of this POST request are also constructed in a SOAP message.

```

1 <n1:syncAction xmlns:n1="http://posregistersync.services.adyen.com">
2   <n1:request>
3     <n1:posRegisterId>Shuttle-016303651</n1:posRegisterId>
4     <n1:terminalId>Shuttle-016303651</n1:terminalId>
5     <n1:includeGenericConfig>ALL</n1:includeGenericConfig>
6     <n1:existingConfigItems>

```

```

7     <nl:ExistingGenericConfigItem>
8       <nl:brandModel>MiuraShuttle</nl:brandModel>
9       <nl:version>adyen-pl-v1_32p7</nl:version>
10    </nl:ExistingGenericConfigItem>
11  </nl:existingConfigItems>
12 </nl:request>
13 </nl:syncAction>

```

Listing 4.20: Web request 7 directed to pal-live.adyen.com.

The SOAP request provides information on the Chip & PIN card reader such as terminal ID and terminal API version. The same information was also presented to ca-live.adyen.com in Listing 4.18, but enclosed by other tags

The syncAction tag suggests a synchronization request. What exactly the synchronization request is requesting is not directly clear from the information provided in the SOAP message. However, the SOAP response to this web request listed in Listing 4.21 provides some clarity.

```

1 <nl:syncActionResponse>
2 <nl:response>
3   <nl:pspReference>4414389518301817</nl:pspReference>
4   <nl:updateList>
5     <nl:TerminalUpdateItem>
6       <nl:syncToTerminalData>
7         ChaONDE0Mzg5NTE4MzAxODE3EhFTaHV0dGx1LTAxNjMwMzY1MRqCA
8         gABQKLFzIpLt21GpuxhC4JO+4HCckCopH5eIDKvIT0a+wRwaQ11DC
9         bb/4LezBfzPalizPVcpXALsapxr30vp0pVUiKyAk64YWWMb3uZZ/7
10        9YRXv1Pr3VO1G50x+0oIw1RxCI9GaiQiihfZW/O8DvSyrufoHLfb
11        2fVC89vUgcLqwyJqtOqgAxEdJ/v/4hTMJrjm+F+14rd67c06Aw9vR
12        [...]
13        Xs81pwYD6JevJ35PkXLMjKFAQgJEoABYc+ksKF/xhjNjGmD8GmsaV
14        wXBggNncLFzfa73kIFCTctJbTvZHR6VuL2DQyfIdpVMGDeETRU+dv
15        CSoiCoolV9x3SPASZMqaMYdEgIm1iKXO3W1OTjYkq76MBS3dHsK1a
16        QEmU7QXXW1NT32zb7ChDCxecdal/qEZkaTaWQOF25dY6GTIwMTUtM
17        DgtMDdUMTQ6NTA6MzArMDI6MDA=
18      </nl:syncToTerminalData>

```

Listing 4.21: Response to the syncAction SOAP request.

The synchronization request involves a card reader firmware update as indicated by the TerminalUpdateItem tag. The update data is included within the syncToTerminalData tags and appears to be encoded according to Base64 (i.e., the string ends with '='). The decoding of the first line of the update data is shown below.

```

1 ChaONDE0Mzg5NTE4MzAxODE3EhFTaHV0dGx1LTAxNjMwMzY1MRqC
2   \ /
3 Base64 decoding
4   \ /
5 4414389518301817 Shuttle-016303651

```

Listing 4.22: Base64 decoding of the first line of the update data.

The rest of the decoded update data comprises random unprintable characters and a part of it is shown in Figure 4.6. The numerical code '4414389518301817' represents the PSP reference code as has already been listed in the same SOAP

message under the tag `pspReference`. The reference code is followed by the card reader (terminal) ID.

```
4414389518301817 Shuttle-016303651 @cÀl K.mFjia Nù Àr@`B~^ 2"! = ú piu &Ûÿ ðì ó=@bdö\ÿp ±
ªq)/$JUR"2 N,ae of gbyá í ú÷TéFçL-Ò OÖ B!7Fj$" Û[ó¼ ó²@çè ·ÜÜóBóÜÖ ÂéÄ"j' é 'úÿâ ì&,æø_µâ-zíÄ° o
G )DGð=U? E zãÖtú T N?AóY $ °þx"jµ OÆÐ,sæãVóÐG3ç-C7)÷ æX iÈ)¥6©½hntD¹Ul$N<ÝzÖno=þza
³Á" `clá bíF? ·a)è½&²è NÄÿÉé 0 z ôjEc >D'ã÷@ã" IQ ú{æ^ÿx~àeÇâ h 'ad7½]èCzO& %l
```

Figure 4.6: Base64 decoding of a part of the firmware update data.

The random looking data suggests that the firmware update is encrypted.

4.4.4. LOG 3

The synchronization request is then followed by two POST request directed to `fapi.payleven.de`. These requests send JSON-formatted log data to the Payleven servers. The first request holds trivial information on the previous web request to `apiproxy.payleven.de` such as time stamp, merchant ID and used network type. The second request is listed in Listing 4.23.

```
19 "events": [
20   {
21     "data": {
22       "adyenMethod": "lib.addDevice",
23       "adyenResponseTime": "15.88500",
24       "adyenTerminalOSVer": "M000-OS-V7-1",
25       "adyen_lib_version": "v1.9p22",
26       "message": "Completed successfully",
27       "network_type": "WIFI",
28       "shuttleId": "Shuttle-016303651",
29       "shuttleName": "Shuttle-016303651",
30       "shuttleVersion": "adyen-pl-v1_32p7",
31       "shuttle_version_upgrade": "",
32       "status": "OK"
33     },
34     "event": "AdyenBoardDevice",
35     "merchantId": "653***",
36     "paymentCountry": "NL",
37     "timestamp": "2015-08-07T12:50:36.036"
38   },
39   {
40     [...]
41     "event": "AdyenDeviceAdded",
42     "merchantId": "653***",
43     "paymentCountry": "NL",
44     "timestamp": "2015-08-07T12:50:37.037"
45   }
46 ]
```

Listing 4.23: First POST request to `fapi.payleven.de`.

4.4.5. WEB REQUEST A2c

The last POST request in the Bluetooth pairing phase is destined for `pal-live.adyen.com/pal/adapter/batchsync.proto`. The `.proto` extensions suggests the use of Google's protocol buffers. Protocol buffers (or `protobuf`) is a language- and platform-neutral mechanism for serializing structured data. It is comparable to XML. The contents of the POST request are shown in Figure 4.7.

```
POST /pal/adapter/batchsync.proto HTTP/1.1
jaasToken: NDIkMWM1NmEtY2QxZi00N2ZjLWlZzjctNmEzZjZhN2RiZTYwXk1xcUd1aFQ5bmfNcDQ2Qmc=
terminalId: Shuttle-016303651
Authorization: Basic OTI4NzZM2QENvbXBhbnkuTWVvYyY2hhbnRBY2NvdW50LjY1M0tsdU1ZG9QT1M6
User-Agent: Dalvik/1.6.0 (Linux; U; Android 4.3; Galaxy Nexus Build/JWR66Y)
Host: pal-live.adyen.com
Connection: Keep-Alive
Accept-Encoding: gzip
Content-Type: application/x-www-form-urlencoded
Content-Length: 580

hï õÜÖ ñ 6iaõ*! {6ao QI ù k*Õª Å¼¼
S_½ *¥8.3Y İu ðÜ'eVFBeüöEİ ;µ¼6% S B;:B 1 Ö:FÇÄbDn @ 7ä92- çç`_il' lð+Äry+b< 0 D ¼ß£aM
ýUEP¶ Sjö«4:ÄÄa: ý è cÖüc O=6y i&:g~«bš' Wwöy' 1UAL ä* x 'Lt»Y oö^Eä ?Ç æð 3
%luU°Er Ey 'Xj)K$W9 mpQe4zI8ªÈ JA`_YO %45 i3°EİñLü¶İßªEäo éÖ ÜRCñZjg½7Q 4kúTB 1r¥ Xaüè«/Oe e
_Đu· ÖSPAGqyRx*!xOe=@?A Y~»U P ääy oO*Gw :á Uä? zWXaöjc = fjyy? EäE üahZ # tÄ Uoiâ%o «'
OÄcG¹ª qi ò:xlæg} {x*³töªæy2 Æ t&l Ari²;Ü dM Lw; ñ y cæäÜE,o-W '©QäYZ£AEL%~«KluÜi «¾
W ?('Üpá ël½Uçp-NUAQ«?O j E P _VÄS_p {ö QXE¶luEÖ.h3O P
```

Figure 4.7: Header and contents of POST request to `pal-live.adyen.com/pal/adapter/batchsync.proto`.

As can be seen from the above figure, the contents of the requests are not human-readable. The data is with high probability represented in a binary wire format intended for the `protobuf` protocol. Protocol data can be decoded using the corresponding `.proto`-file. This file specifies how the information is structured. However, without the corresponding `.proto`-file decoding the `protobuf` data becomes more complicated since the `protobuf` format is ambiguous. The `protobuf` compiler offers the possibility to decode these raw binary data. The raw binary can be decoded with the following `protobuf` command:

```
1 $ protoc decode_raw < batchsync.bin
```

Unfortunately, `protobuf` fails parsing the binary data, thus suggesting a incorrect `protobuf` format. This may have several reasons:

- the data is not serialized according to `protobuf`; e.g., the content-type header line does not indicate the use of `protobuf`. However, it is very likely that `protobuf` is indeed used. The firmware of the card reader includes the `protobuf` API.
- the data is serialized according to `protobuf`, but is encrypted on top of that.

The response to web request A2c also holds unreadable data. It is assumed that the serialized application data exchanged with Adyen (especially the `pal-live` web server) is subjected to application-level encryption. That is to say, the card reader and the Adyen back-end communicate over an application-level encrypted channel.

4.5. NETWORK TRAFFIC DURING THE TRANSACTION PROCESS

The Payleven app exchanges information with the following web addresses during the transaction phase:

| | | |
|-------|---------------|--|
| P4a | URL Method | apiproxy.payleven.de/api/transaction/register-confirm POST |
| Log 4 | URL Method | fapi.payleven.de/fapi/v1/logs POST |
| A3a | URL Method | pal-live.adyen.com/pal/adapter/posmessage.proto POST |
| A4a | URL Method | pal-live.adyen.com/pal/adapter/posmessage.proto POST |
| P5a | URL Method | apiproxy.payleven.de/api/transaction/update POST |
| Log 5 | URL Method | fapi.payleven.de/fapi/v1/logs POST |
| A4b | URL Method | pal-live.adyen.com/pal/adapter/batchsync.proto POST |

Table 4.7: Approached web addresses during the transaction phase.

Figure 4.8 summarizes the above listed web requests in an MSC. The sequence chart sets out the message exchange arrows P4, P5, A3, A4 from the high-level MSC 2.17 into individual web request. Please recall that ‘A’ stands for the communication with Adyen and ‘P’ stands for the communication with Payleven. Moreover, A3 covers the data traffic after the banking card has been entered in the card reader. A4 covers the data traffic after the PIN has been entered.

4.5.1. WEB REQUEST P4A

The first web request in the transaction phase is directed to Payleven’s apiproxy web server. The request body consists of multi-part form data and lists 6 parameters that hold information on the transaction amount, terminal version and ID, type, the payment provider and the currency. These parameters and its values are listed in Table 4.8.

The parameter values ‘1.00’ and ‘EUR’ indicate a transaction value of €1. It is unknown what the parameter `type` and the value ‘UN’ refer to. The parameter `type` probably refers to the terminal type.

The response to the POST request holds a JSON-message and has a content length of 1066. Some interesting object names and values from the JSON-message are listed in Table 4.9.

The response body also lists the URL to the corresponding transaction receipt.

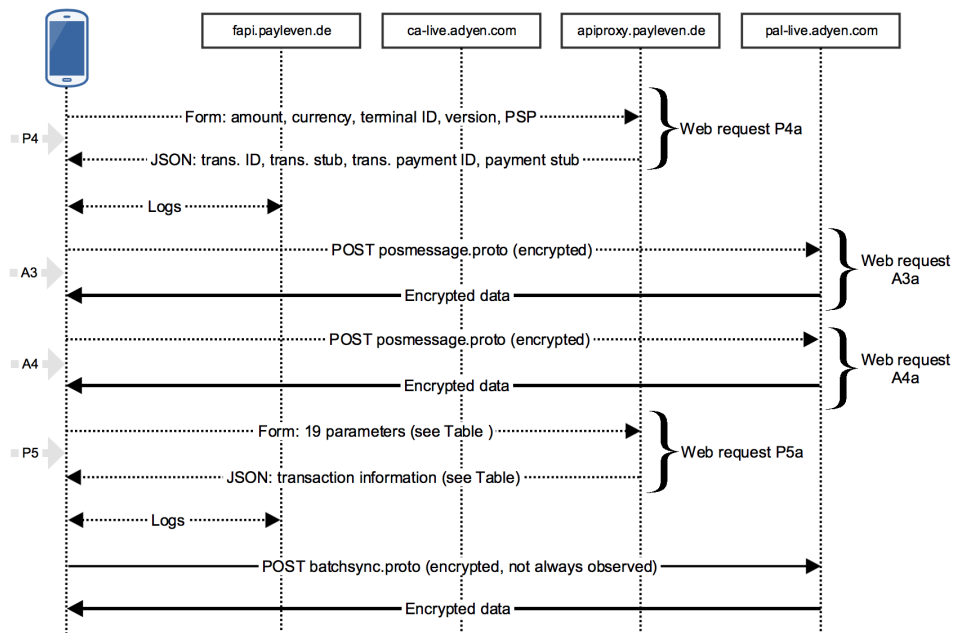


Figure 4.8: The network traffic during the transaction phase. A3 represents the data exchange after the banking card has been entered in the card reader. A4 represents the data exchange after the PIN has been entered.

| # | Parameter | Value |
|---|------------------|-------------------|
| 1 | amount | 1.00 |
| 2 | terminal_version | adyen-pl-v1_32p7 |
| 3 | terminal_id | Shuttle-016303651 |
| 4 | type | UN |
| 5 | psp | Adyen |
| 6 | currency | EUR |

Table 4.8: Multi-part form data parameters of web request 10.

The generic URL to a receipt is listed in Listing 4.24.

```
1 https://service.payleven.com/nl/receipt?payment_stub=*****&hash
   =*****
```

Listing 4.24: Generic URL to an online transaction receipt

The query of the URL is constructed from the payment stub (see Table 4.9) and a 40 digit hash value. Note that the above listed URL links to the receipt *prior* to the actual transaction. The receipt indicates that the transaction has not been completed yet (Dutch: transactie nog niet voltooid) and lists the merchant ID and the value of the key `id_intpay_transaction` (Dutch: betalingsbewijs ID). The receipt is updated after the transaction has been completed.

As can be seen in Table 4.9, the response provides the Payleven app with the merchant ID, a transaction ID and transaction stub, and a transaction payment ID and payment stub. The merchant ID has already been mentioned in the re-

| JSON Object | Value |
|-------------------------------|----------------------|
| fk_marketplace_merchant | 653*** |
| id_intpay_transaction | 52792027 |
| transaction_stub | d1f61a6ff3426e6df851 |
| id_intpay_transaction_payment | 47168944 |
| payment_stub | 04fda3a8dc638424b17e |

Table 4.9: Some significant JSON objects and values from the response to web request 10.

response to web request P1a. The transaction ID and transaction stub have already been listed in the response to web request A1a. This is interesting since no transaction has been initiated during web request A1a. This also means that the transaction ID and transaction stub are not directly attached to a particular transaction. However, the transaction *payment* ID and *payment* stub are first mentioned here in the response to web request P4a. The object `payment_stub` holds a 20 digit hexadecimal code. The name of the object suggests that it denotes some kind of receipt or record - payment stub is a different denomination for payment check. The payment stub thus probably forms the link between the transaction and the corresponding receipt. This fact is also reflected in the URL of the receipt; the payment stub is used in the query.

4.5.2. LOG 4

Log request 4 lists JSON-formatted information on several events in the application prior to the transaction. These log events are listed in Table 4.10. The payment and transaction stubs are also included in the logs under the event `AdyenTransactionStarted`. The payment and transaction stubs are listed respectively as `paymentID` and `transactionID` in the logs, as can be seen in Listing 4.25. The merchant ID is also listed in the logs.

| | |
|---|-----------------------------|
| 1 | PurchaseAttempt |
| 2 | PaymentMethodSelected |
| 3 | ConnectingToBluetoothDevice |
| 4 | SetDefaultDevice |
| 5 | TurnAroundTime |
| 6 | AdyenTransactionStarted |

Table 4.10: Log events from Log request 4.

```

92 "amount": 100,
93 "currency": "EUR",
94 "data": {
95   "adyenTerminalOSVer": "M000-OS-V7-1",
96   "adyen_lib_version": "v1.9p22",
97   "completed_status": "DEFAULT",
98   "cvm": "UNINITIALIZED",
99   "network_type": "WIFI",

```

```

100     "psp": "Adyen",
101     "shuttleId": "20:14:09:08:2F:01",
102     "shuttleName": "payleven016-303651",
103     "shuttleVersion": "adyen-pl-v1_32p7"
104 },
105 "event": "AdyenTransactionStarted",
106 "merchantId": "653****",
107 "paymentCountry": "NL",
108 "paymentId": "04fda3a8dc638424b17e",
109 "timestamp": "2015-08-08T08:04:12.012",
110 "transactionId": "d1f61a6ff3426e6df851"

```

Listing 4.25: Part of the logs showing information on the event `AdyenTransactionStarted`.

4.5.3. WEB REQUEST A3A & A4A (ENCRYPTED)

During the transaction process, that is, inserting the bank card and entering the PIN code, the card reader exchanges transaction information with Adyen. This sensitive information is with high probability included in the A3a and A4a POST requests transmitted to `adyen.com/pal/adapter/posmessage.proto` during the transaction process.

The header of the POST request is equal to the header of web request A2c (`adyen.com/pal/adapter/batchsync.proto`). It includes the authorization header line, the `jaastoken` and the card reader (terminal) ID. The contents of web request A3a and A4a are also encrypted.

Please recall the high-level MSC 2.17. Web request A3a occurs after the banking card has been inserted into the Chip & PIN card reader. Web request A4a occurs after the PIN has been entered on the card reader.

4.5.4. WEB REQUEST P5A

The next POST request is directed to `apiproxy.payleven.de` and contains 19 multi-part/form data parameters. These parameters are set out in Table 4.11.

The first parameter is the payment stub which was first included in the response to web request P4a. The next parameter (`psp_payment_id`), on the other hand, does not appear in the previous messages. It seems that the ID is generated by the payment service provider or card reader, and sent to the Payleven app after the transaction. The app then forwards the PSP payment ID to the Payleven back-end.

The next parameters describe the method and application used to process the transaction. The payment method is Chip & PIN (or EMV) according to the parameter `method`. The AID is an acronym for Application Identifier: the code 'A0000000043060' refers to Maestro (Debit). This is also reflected in the parameters `application_label` and `application_preferred_name`.

The parameter `iin` holds the issuer identification number (IIN). Information on the issuer can be retrieved based on the IIN. The issuer information listed in 4.12 corresponds to the IIN '673703'.

| # | Parameter | Value |
|----|----------------------------|----------------------|
| 1 | payment_stub | 04fda3a8dc638424b17e |
| 2 | psp_payment_id | 4814390210829861 |
| 3 | method | chippin |
| 4 | aid | A0000000043060 |
| 5 | application_label | MAESTRO |
| 6 | application_preferred_name | MAESTRO |
| 7 | iin | 673703 |
| 8 | expiry | 1902 |
| 9 | issue_date | 1402 |
| 10 | icc | ICC |
| 11 | card_scheme | mastercard_maestro |
| 12 | pay_code | DD.CP |
| 13 | pi_hash | C133084544881715 |
| 14 | terminal_id | Shuttle-016303651 |
| 15 | terminal_version | adyen-pl-v1_32p7 |
| 16 | auth_code | 85I2P1 |
| 17 | method | chippin |
| 18 | total_amount | 1.00 |
| 19 | desc | <i>empty</i> |

Table 4.11: Multi-part/form data parameters of web request 14 (apiproxy.payleven.de).

Parameters 8 to 11 provide information on the card, such as the expiration date and date of issue. The parameter `icc` stands for integrated circuit card (ICC) and is also set to 'ICC'. This parameter indicates that the bank card is in possession of a chip.

| | |
|--------------------|---------------|
| Card Brand | Maestro |
| Issuing Bank | ING Bank N.V. |
| Card Type | Debit |
| Card Level | Standard |
| Iso Country Name | Netherlands |
| Iso Country A2 | NL |
| Iso Country A3 | NLD |
| Iso Country Number | 528 |

Table 4.12: Information on issuer based on the IIN 673703.

The parameter `pay_code` holds the value 'DD.CP'. It is not clear what the code directly means; however some information can be obtained from the source code of the Payleven app. The string 'pay_code' is found in the JAVA-file `DefaultCommand.java`.

```

389 if (carddata.isDebit())
390 {
391     return "DD";

```

```

392 } else
393 {
394     return "CC";
395 }

```

Listing 4.26: DefaultCommand.java

The code snippet in Listing 4.26 shows that the first half of the pay code stands for the card type: debit card (DD) or credit card (CC). Also, it appears that the last half of the pay code ('CP') is only appended to 'DD' upon approved payment. 'CP' is presumably an acronym for 'Completed Payment'. However, the pay code for a canceled transaction also ends with 'CP'. The pay code for an unauthorized payment can also be retrieved from `DefaultCommand.java`; that is 'UN.DC'.

From the remaining parameters, only `pi_hash` and `auth_code` are noteworthy. The parameter `pi_hash` contains a 16 digit string value which represents a card token.

```

265 if (purchase.getCardToken().isPresent())
266 {
267     addEntityPart("pi_hash", (String)purchase.getCardToken().get());
268 }

```

Listing 4.27: The parameter `pi_hash` in `DefaultCommand.java`

The parameter `auth_code` contains a 6 digit alphanumeric code and represents a certain authorization code. The code is different for each transaction. The amount of the transaction is covered in the parameter `total_amount`.

The response to web request P5a contains the same values as listed in Table 4.11. The only extra information included in the response concerns two numerical codes:

| # | Parameter | Value |
|---|---|----------|
| 1 | <code>fk_intpay_transaction</code> | 52234204 |
| 2 | <code>fk_intpay_transaction_payment_tracking</code> | 45040574 |

Table 4.13: Part of the response to web request P5a.

4.5.5. LOG 5

Log request 5 lists JSON-formatted information on several events in the application after the transaction. These log events are listed in Table 4.14.

| | |
|---|-----------------------------|
| 1 | AdyenTransactionSuccess |
| 2 | ApiCallRequest |
| 3 | ReceiptViewViewed |
| 4 | ReceiptSkipped |
| 5 | ShownPaymentStatusFinalView |
| 6 | NewPayment |

Table 4.14: Log events from log request 5.

```

92 {
93   "amount": 100,
94   "currency": "EUR",
95   "data": {
96     "adyenTerminalOSVer": "M000-OS-V7-1",
97     "adyen_appinfo_model": "Galaxy Nexus",
98     "adyen_appinfo_name": "Payleven",
99     "adyen_appinfo_os": "4.3",
100    "adyen_code": "0",
101    "adyen_lib_version": "1.9p22",
102    "adyen_message": "Transaction approved",
103    "completed_status": "APPROVED",
104    "cvm": "PIN_OFFLINE_ENCIPHERED",
105    "entry_mode": "ICC",
106    "network_type": "WIFI",
107    "psp": "Adyen",
108    "shuttleId": "20:14:09:08:2F:01",
109    "shuttleName": "payleven016-303651",
110    "shuttleVersion": "adyen-pl-v1_32p7",
111    "status": "APPROVED"
112  },
113   "event": "AdyenTransactionSuccess",
114   "merchantId": "653***",
115   "paymentCountry": "NL",
116   "paymentId": "04fda3a8dc638424b17e",
117   "timestamp": "2015-08-08T08:04:49.049",
118   "transactionId": "d1f61a6ff3426e6df851"
119 },

```

Listing 4.28: Information contained in the log event AdyenTransactionStarted.

From this log file it becomes clear that 'PIN_OFFLINE_ENCIPHERED' (i.e., offline PIN) is used as the cardholder verification method (CVM) for this transaction. This implies that the PIN is transmitted enciphered to the card and is then verified by the card.

4.5.6. WEB REQUEST A4B (ENCRYPTED)

The last request in the transaction phase concerns the *batchsync.proto* web request. This request has been observed previously in web request A2c. It should be noted that this sync-request is not observed during each transaction. The URL-path of the web request suggest that the message exchange with Adyen is aimed at some sort of batch synchronization. The fact that the request is not always observed strengthens this belief. The batch probably holds a set of data from current and possibly previous transactions which need to be communicated to Adyen at one time. Unfortunately, it is not possible to determine what specific information is exchanged in the batch since the batch is subjected to application-level encryption.

Moreover, the fact that 'PIN_OFFLINE_ENCIPHERED' is used fits into this definition as a all transactions verified with an offline PIN need in the end to be verified by Adyen and the bank. Hence, these batches may contain such transaction data and may be synced with the back-end after a certain period of time.

4.6. NETWORK TRAFFIC DURING THE REFUND PROCESS

The refund process consists of only one web request. A5 refers to the message exchange arrow in the high-level MSC 2.18.

| | | |
|-----|--------|---|
| A5a | URL | https://apiproxy.payleven.de/fapi/v3/transaction/ refund |
| | Method | POST |

Table 4.15: The web requests during the refund process.

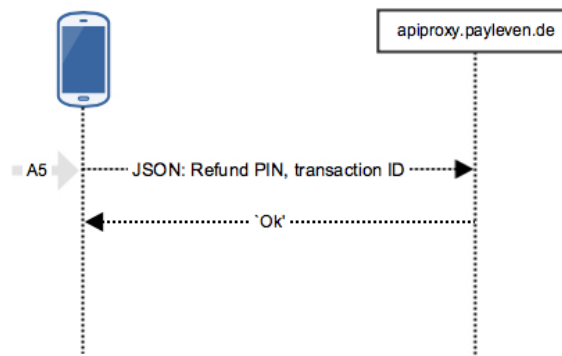


Figure 4.9: The network traffic during the refund process.

4.6.1. WEB REQUEST A5A

Section 2.3.5 describes the refund process. Please recall that in order to initiate

a refund process, the merchant has to authorize the refund with the self-set 4-digit refund authorization PIN code. This PIN is therefore transmitted together with the associated transaction ID to the Payleven back-end. Listing 4.29 lists the whole contents of the refund request.

```
92 POST /fapi/v3/transaction/refund HTTP/1.1
93 User-Agent: Payleven 2.23.0, samsung/Galaxy Nexus, Android 4.3, US
94 Accept-Language: en
95 X-Device: samsung/Galaxy Nexus
96 X-Device-Os: Android
97 X-Device-Model: samsung/Galaxy Nexus
98 X-App-Version: 2.23.0
99 X-API-Version: 89
100 Accept: application/json
101 Authorization: Payleven yiQH8hnp9pWd3pCNwCP9Ts8walyVHmRgjpW0eEBO-ffulBQG
102 X-Coordinates: 52.107904851436615, 5.088488711044192
103 Content-Type: application/json; charset=UTF-8
104 Content-Length: 53
105 Host: fapi.payleven.de
106 Connection: Keep-Alive
107 Accept-Encoding: gzip
108
109 {"pin":"1337","transactionId":"dlf61a6ff3426e6df851"}
```

Listing 4.29: The contents of web request A5a; i.e., the refund request. The refund PIN and the transaction ID are located in the body of the request.

The Payleven back-end thus approves the refunds request if and only if the refund PIN corresponding to that Payleven account is correct. The Payleven account is linked to the session token included in the authorization header line.

5

BLUETOOTH TRAFFIC ANALYSIS

This chapter sets out the data traffic sent over the Bluetooth channel. Section 5.1 describes the data traffic during the Bluetooth pairing process and Section 5.2 focuses on the data traffic during the transaction phase.

5.1. BLUETOOTH TRAFFIC DURING THE PAIRING PROCESS

Figure 5.1 shows the Bluetooth message sequence chart (MSC) between the smartphone (or more precisely the Payleven app) and the Payleven Chip & PIN card reader during the Bluetooth pairing process. The charts in this chapter only cover the RFCOMM messages¹ since these hold application data. Other Bluetooth protocols are out of scope. For example, the actual establishment of the Bluetooth connection between the smartphone hardware and the card reader is carried out on a different Bluetooth protocol. This protocol follows strict Bluetooth standards and does not include any application data, and therefore, it is not included in this study. The focus of this chapter primarily lies on the application data exchanged over the Bluetooth connection.

Each message in the MSCs included in this chapter is labeled with a number. The label numbers also indicate the transmission order of the messages. However, please note that some messages are retransmitted during the protocol run.

Message 1 to 7 in MSC 5.1 cover the message exchange during the card reader registration process. This process starts once the card reader is successfully paired with the smartphone hardware (i.e., step 6 and Figure 2.8c from Section 2.3.3). During this stage of the protocol run, the card reader and the Payleven app exchange basic information.

¹<https://developer.bluetooth.org/TechnologyOverview/Pages/RFCOMM.aspx>

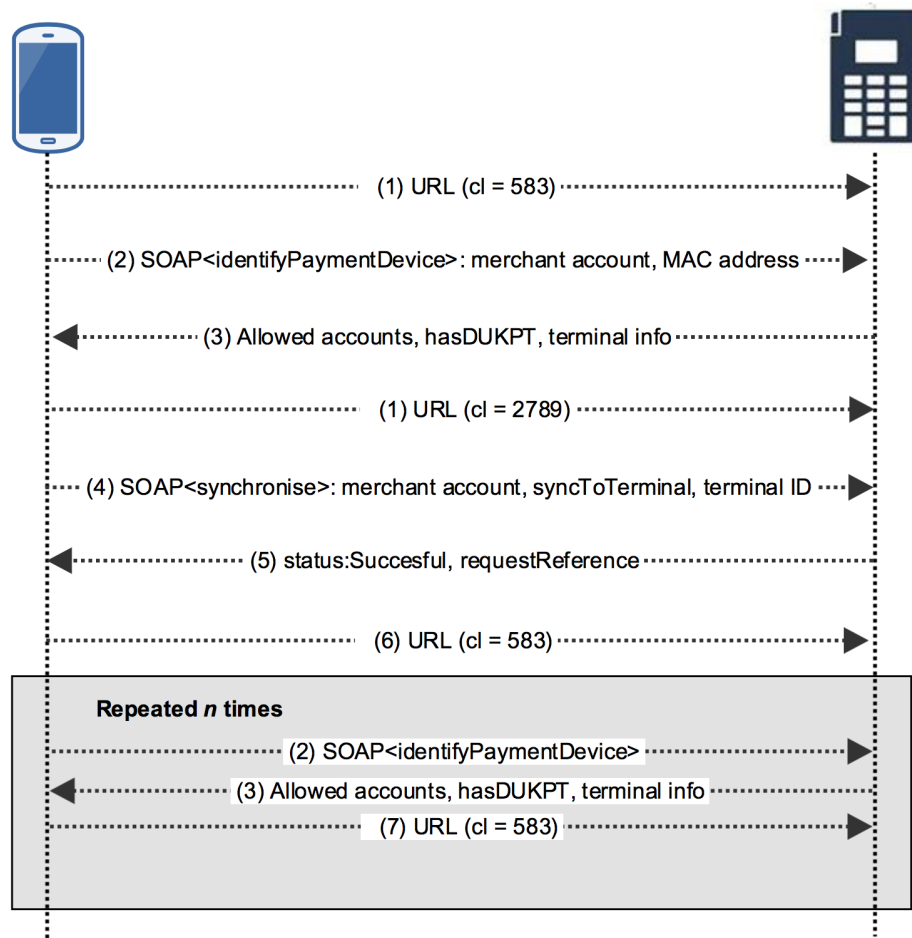


Figure 5.1: Message sequence chart of the Bluetooth traffic during the card reader registration phase.

5.1.1. BLUETOOTH MESSAGE 1

The first message originates from the Payleven app and contains a URL. The URL is listed below:

```
1 url=http://bluetooth-remote/posregister/services/PosRegister/v10&#cl=583
```

Listing 5.1: Contents of message 1.

The URL is constructed in `RunSoapRequest.java`. The JAVA-file is part of the Payleven app source code.

```

57 private static final String BLUETOOTH_REMOTE_URL = "http://bluetooth-
    remote/posregister/services/PosRegister/%s";
58 public static final String DEFAULT_SOAP_VERSION = "v10";

419 if (Text.isEmptyOrNull(s3)
420 {
421     s3 = "v10";
422 }
423 if (deviceinfo.getConnectionType() == 2)
424 {
425     devicepreferences.setMacAddress(deviceinfo.getDeviceId());
  
```

```
426     s3 = String.format("http://bluetooth-remote/posregister/services/  
    PosRegister/%s", new Object[] {  
427         s3  
428     });
```

The string 'v10' in the path name of the URL refers to the SOAP version and is concatenated at the end of the URL in line 426. The fragment `c1=583` denotes the content-length of the upcoming message(s).

It is not clear how the card reader handles the URL sent in message 1. However, it can be assumed that the URL triggers the set-up for the card reader registration process. What exactly happens next in the card reader is unknown.

5.1.2. BLUETOOTH MESSAGE 2

The second message also originates from the Payleven app and contains a SOAP request according to the SOAP protocol specification. SOAP is also used by the Payleven app to exchange information with the Adyen back-end.

```
1 <?xml version='1.0' encoding='UTF-8' standalone='yes' ?>  
2 <n0:Envelope xmlns:n0="http://schemas.xmlsoap.org/soap/envelope/" xmlns:  
    xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org  
    /2001/XMLSchema-instance">  
3   <n0:Body>  
4     <n1:identifyPaymentDevice xmlns:n1="http://posregister.services.adyen.  
        com">  
5       <n1:request>  
6         <n1:merchantAccount>653Klu*****</n1:merchantAccount>  
7         <n1:posRegisterMacAddress>20:14:09:08:2F:01</n1:posRegisterMacAddress>  
8         <n1:posRegisterConfiguredName>Shuttle-016303651</n1:  
        posRegisterConfiguredName>  
9       </n1:request>  
10      </n1:identifyPaymentDevice>  
11    </n0:Body>  
12  </n0:Envelope>
```

Listing 5.2: Contents of message 2

The body of the SOAP request holds the following three elements:

- `merchantAccount` - the Adyen merchant account coupled to the logged-in Payleven account;
- `posRegisterMacAddress` - the Bluetooth MAC address of the Chip & PIN card reader registered to the merchant account. POS stands for point-of-sale and is another term to refer to the card reader;
- and `posRegisterConfiguredName` - the card reader type and serial number.

The SOAP request is constructed in `IdentifyPaymentDeviceRequest.java`. The aim of a SOAP request is often stated in the first tag after the body-tag; in this case, the aim is to identify the used payment device.

5.1.3. BLUETOOTH MESSAGE 3

Message 3 holds the response to the previous SOAP request. The SOAP response comprises 13 SOAP elements. Additionally, the first element, `additionalData`, includes the following 7 entries: `DeviceLocale`, `SupportedCurrencies`, `DeviceAllowedMerchantAccounts`, `SoapVersion`, `hasDUKPT`, `SupportedPaymentMethods` and `UnconfirmedBatches`. The key `DeviceAllowedMerchantAccounts` suggests that one single card reader can be used with multiple accounts. This is true to a certain extent. It is possible to create multiple sub-accounts (with their own e-mail address and password) under the main account. Each sub account is then permitted to pair with the concerning card reader. In message 3, `DeviceAllowedMerchantAccounts` is set to the main account.

Another interesting entry is `hasDUKPT`, which is set to 'true'. This entry indicates the use of the DUKPT (Derived Unique Key Per Transaction) key management scheme. It can be stated with certainty that the card reader is in possession of a secret master key that is used to derive unique transaction keys. The payload of the HTTP requests to `pal-live.adyen.com` already designated the use of application-level encryption. This presumption is only strengthened in this chapter: the information exchange between the Payleven Chip & PIN card reader and the Payleven app is obfuscated at certain stages in the protocol run. For example, message 4 contains a firmware update that is obfuscated.

The remaining SOAP elements provide information regarding the card reader, such as brand and type, hardware version, serial number et cetera.

The next message after message 3 is identical to message 1; only the last fragment of the URL (`c1=2789`) differs. This suggest that the upcoming message has a content-length of 2789 bytes. According to Wireshark, the data sent in message 4 has a length of 2807 bytes. The `c1` fragment gives a good indication on the content-length of the upcoming message, but it is unclear what part of the message the `c1` fragment actually covers.

5.1.4. BLUETOOTH MESSAGE 4

Message 4 is in comparison to the other Bluetooth messages fairly large. This can be explained by the contents of the SOAP request. The body consists of three elements: `merchantAccount`, `terminalId` and `syncToTerminal`², where the latter holds 2250 bytes of random-looking data (see Listing 5.3).

```

1 <n1:syncToTerminal>
2   ChA0NDE0Mzg5NTE4MzAxODE3EhFTaHV0dGx1LTAxNjMwMzY1MRqCA
3   gABQKLfzIPLt21GpuxhC4JO+4HCckCoph5eIDKvIT0a+wRwaQ11DC
4   bb/4LezBfzPaliZPVcpXALsapxr30vp0pVUiKyAk64YWWMb3uZZ/7
5   9YRXv1Pr3VO1G50x+0oIw1RxCIT9GaiQiihfZW/O8DvSyrufOHLfb
6   2fVC89vUgcLqwyJqtOqgAxEdJ/v/4hTMJrjm+F+14rd67cO6Aw9vR
7   [...]
8   Xs81pwYD6JevJ35PkXLMjKFAQgJEoABYc+ksKF/xhjNjGmD8GmsaV
9   wXBgqNncLFZfa73kIFCTctJbTvZHR6VuL2DQyfiDpVMGDeETRU+dv
10  CSoiCoolV9x3SPASZMqaMYdEgIm1iKXO3W1OTjYkq76MBs3dHsK1a
11  QEmU7QXXW1NT32zb7ChDCxecdal/qEZkaTaWQOF25dY6GTIwMTUtM

```

²The term 'terminal' is used here to refer to the Chip & PIN card reader.

12 DgtMDdUMTQ6NTA6MzArMDI6MDA=

Listing 5.3: Contents of the `syncToTerminal` entry

The `syncToTerminal` tag has been detected previously in the network analysis in Section 4.4.3. The SOAP response from Adyen to web request A2b contains the `syncToTerminalData` entry (see Listing 4.21) which holds the exact same data as the `syncToTerminal` entry found in Bluetooth message 4. As already stated in Section 4.4.3, the contents of the `syncToTerminalData` entry concern a firmware update. Therefore, it can be concluded that Bluetooth message 4 pushes the firmware update to the card reader via the Bluetooth connection.

The update data in the `syncToTerminal` tag is encoded according to the Base64 encoding scheme. The decoding of the `syncToTerminal` data in UTF-8 characters is shown in Figure 5.2. Only a part of first line is human-readable. The card reader ID ('Shuttle-016303651') is listed together with a 16 digit numerical code ('1314374629683409'). The numerical code represents the PSP reference code (also listed in Listing 4.21). The rest of the decoded data is encrypted.

```
1314374629683409 Shuttle-016303651 b'6 kpf|eE3 2Ù} /
Ø ÓÓ~jZ/Ø {xwdIG-ø½M le÷ VóÖÉ ØøjT Óo 6CXO67$Ø' w
¶JJVæ$" Q8 ),î (N3P iÁY2y. î|G øð,( íá»2¨· ýú1üA¶
æf÷ÁëäT .æs 9{| N d [«u
ÏÜYC 9WrO øt-ç`Ööó þ´ 4|sRÆ5É £ Ø O7 y/â ÉA, ræ !ßéá?ß£ ñD
â·x
« S ýp@Ó à ìFMReh É , pÿ v: OÁ= <w-|ô %7 vp?ø'LæiâW ->0X CÄ@-¾áj|>
ÄD Ø ìYA Dæ £|
" . . . . .
```

Figure 5.2: Base64 decoding of the `syncToTerminal` data

Message 4 is probably the most interesting among the Bluetooth messages. The reason for this is that any modifications of the update data made on the network part are directly reflected in the contents of the `syncToTerminal` entry. This opens interesting possibilities for malicious attackers. This allows attackers to inject tampered data into the card reader where it was otherwise nearly impossible. That is to say, an attacker may manipulate the update data in such way that the card reader crashes or is rendered unusable. The attacker may also execute malicious code on the card reader by injecting the code first via the update item.

5.1.5. BLUETOOTH MESSAGE 5

Subsequently, the card reader replies to message 4 with the status of the synchronization request and a `requestReference`. In this case, the update has been received successfully by the card reader, hence that the `status` element contains the string 'Successful'. The `requestReference` elements holds an 18-digit alphanumeric code: '16PK71437462956063'.

5.1.6. BLUETOOTH MESSAGE 6

Listing 5.4 shows the contents of message 6. The message contains a URL that is similar to the one sent in message 1. The only difference is that an authentication token (`jaastoken`) is inserted to the message.

```

1 url=http://bluetooth-remote/posregister/services/PosRegister/v10&jaastoken
  =
  NjU4ZDg1ZmYtM2VlMi00YjQ5LTg5ZDEtMTE3YWwNINjc0NmYzXk1xcUd1aFQ5bmFncDQ2UW1j
  =&#cl=583

```

Listing 5.4: Contents of Bluetooth message 6

Decoding the *jaastoken* according to Base64 results in:

```

1 658d85ff-3ee2-4b49-89d1-117ace6746f3^MqqGuhT9naMp46Qmc

```

Listing 5.5: Base64 decoding of the *jaastoken*

The *jaastoken* appears for the first time in the SOAP response to web request A1a and is generated by Adyen during app initialization phase (see Section 4.3). As can be seen in Listing 5.5, the *jaastoken* is constructed from two parts: the app ID ('658d85ff-3ee2-4b49-89d1-117ace6746f3'), which is generated by the Payleven app and sent to Adyen in web request A1a; and, a 17-digit alphanumeric code ('MqqGuhT9naMp46Qmc') generated by Adyen. These two parts form an authentication token for web requests destined for Adyen.

5.1.7. BLUETOOTH MESSAGE 7

The Payleven app then retransmits message 2 (*identifyPaymentDevice*) and the card reader then replies with message 3 (*identifyPaymentDeviceResponse*). Message 7 is identical to message 6. The URL is now preceded by an authorization header and authorization code. The header indicates the use of the basic access authentication method. According to this method, the authorization code is the Base64 encoding of '[username]:[password]'.

```

1 @authorization=Basic
  OTI4KioqKkBDdb21wYW55Lk1lcmNoYW50QWNjb3VudC42NTNLbHUqKioqKioqOg==&#url=
  http://bluetooth-remote/posregister/services/PosRegister/v10&jaastoken
  =
  NjU4ZDg1ZmYtM2VlMi00YjQ5LTg5ZDEtMTE3YWwNINjc0NmYzXk1xcUd1aFQ5bmFncDQ2UW1j
  =&#cl=583G

```

Listing 5.6: Contents of message 7

However, decoding the authorization code results only in a username.

```

1 OTI4KioqKkBDdb21wYW55Lk1lcmNoYW50QWNjb3VudC42NTNLbHUqKioqKioqOg==
2 \ /
3 Base64 decoding
4 \ /
5 928****@Company.MerchantAccount.653Klu*****:

```

Listing 5.7: Base64 decoding of the authorization code from message 7

As can be seen in Listing 5.7, the authorization code is built up from the Adyen merchant account ('653*****') and the Adyen username ('928***'). Both the merchant account and the Adyen user code are communicated to the Payleven app during the app login phase (namely, in the response to web request P1a).

The messages 2, 3 and 7 are retransmitted as long as the Bluetooth connection is active. The retransmission is ceased when a monetary transaction is started in the Payleven app.

5.2. BLUETOOTH TRAFFIC DURING THE TRANSACTION

Figure 5.3 presents the MSC of the Bluetooth communication between the smart-phone and the card reader during the transaction phase. The transaction phase is split up over three separate MSC diagrams; MSC 5.3 is the first of three.

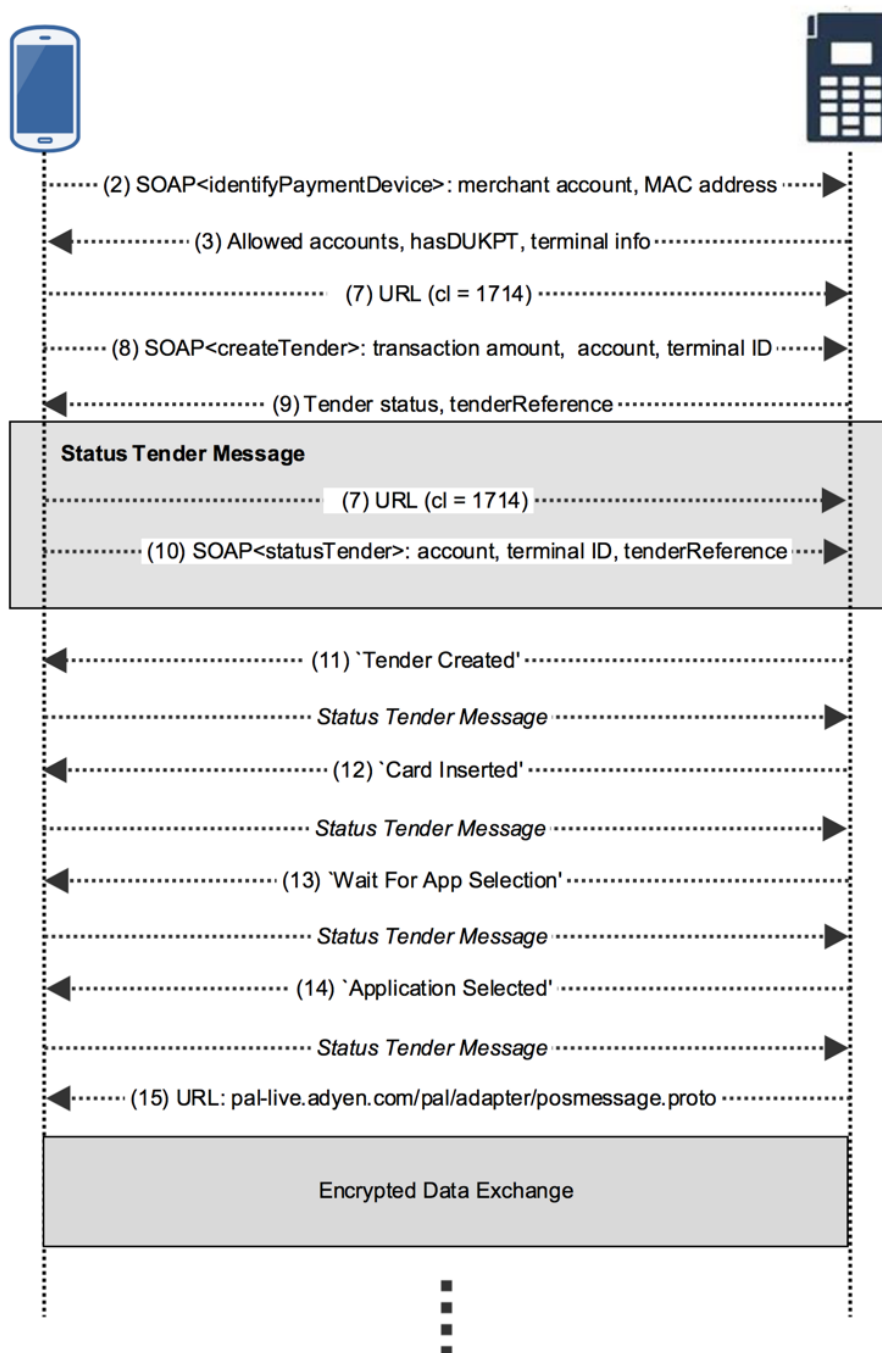


Figure 5.3: Message sequence chart of the Bluetooth traffic during the transaction phase. Part 1 of 3.

5.2.1. BLUETOOTH MESSAGE 8

Message 8 indicates the start of the transaction phase. The Payleven app sends message 8 (`createTender`) to the Chip & PIN card reader, requesting the terminal to create a *tender* (a tender is an offer to buy an asset at a stated fixed price - synonyms are: bid, offer, quote). The SOAP request in message 8 includes standard information regarding the merchant and the card reader, and also provides the reader with the transaction amount (see line 7 in Listing 5.8: 100 represents €1.00). The reference tag holds the payment stub (see Section 4.5.1).

```

1 <n1:merchantAccount>653Klu*****</n1:merchantAccount>
2 <n1:terminalId>Shuttle-016303651</n1:terminalId>
3 <n1:reference>04fda3a8dc638424b17e</n1:reference>
4 <n1:transactionType>GOODS_SERVICES</n1:transactionType>
5 <n1:amount>
6 <n2:currency xmlns:n2="http://common.services.adyen.com">EUR</n2:currency
  >
7 <n3:value xmlns:n3="http://common.services.adyen.com">100</n3:value></n1:
  amount>
8 <n1:options>
9 (* Tender Options *)

```

Listing 5.8: First part of the SOAP request in message 8 (`createTender`)

Furthermore, the SOAP request provides the card reader with several tender options: `ResetCurrentTender`, `AttendantActionHandler`, `GetAdditionalData` and `ReceiptHandler`. The `additionalData` element in message 8 consists of 10 entries, such as the time stamp, longitude and latitude, and information on the Android device and application. The following information is extracted from the application:

- `appinfo.posregisterconfiguredname` - 453009d98289fdd
- `appinfo.lib` - 1.9p22
- `appinfo.appid` - 658d85ff-3ee2-4b49-89d1-117ace6746f30
- `appinfo.os` - 4.3
- `appinfo.model` - Galaxy Nexus
- `appinfo.appname` - Payleven

The `appid` is used to construct the `jaastoken` from Listing 5.4 and 5.5 (see Section 4.4.2).

5.2.2. BLUETOOTH MESSAGE 9

Message 9 (`createTenderResponse`) contains the SOAP response of the card reader to message 8. The response is shown in Listing 5.9.

```

1 [...]
2 <n1:response>
3 <n1:createStatus>Created</n1:createStatus>
4 <n1:nextTenderStatusPollSeconds>0</n1:nextTenderStatusPollSeconds>

```

```

5 <n1:tenderReference>16PK71437462986069</n1:tenderReference>
6 </n1:response>
7 [...]

```

Listing 5.9: SOAP response in message 9 (createTenderResponse)

The response includes information on the status of the `createTender` request and provides the Payleven app with a reference code. The `nextTenderPollSeconds` element is set to 0.

5.2.3. BLUETOOTH MESSAGE 10

Hereafter, message 7 is retransmitted. The content-length fragment `c1` of the URL is set to 524, which indicates the length of message 10 (`statusTender`). The contents of message 10 are shown in Listing 5.10. The SOAP request in message 10 provides the card reader with information on the merchant and the corresponding card reader (terminal ID) accompanied with the tender reference. This reference is the same as the one sent by the card reader in message 9 (Listing 5.9).

```

1 [...]
2 <n1:request>
3 <n1:merchantAccount>653Klu*****</n1:merchantAccount>
4 <n1:terminalId>Shuttle-016303651</n1:terminalId>
5 <n1:tenderReference>16PK71437462986069</n1:tenderReference>
6 </n1:request>
7 [...]

```

Listing 5.10: Contents of message 10 (`statusTender`)

Message 7 (`c1=524`) and message 10 (`statusTender`) are retransmitted multiple times during the rest of the protocol run. These two messages are bundled together into ‘Status Tender Message’ (see MSC 5.3). The card reader then replies with a `statusTenderResponse` in the messages 11, 12, 13, 14, 16, 19, 20, 21, 22, 24 and 25.

5.2.4. BLUETOOTH MESSAGES 11 TO 14

The first four `statusTenderResponse` messages (11 to 14) have the same structure. The messages contain information on the state of the tender; in the case of message 11, the state is set to `TENDER_CREATED` (see Listing 5.11). In addition to the tender reference, the SOAP response also provides the Payleven app with a request reference. The `nextTenderStatusPollSeconds` element is now set to 1.

```

1 [...]
2 <n1:response>
3 <n1:nextTenderStatusPollSeconds>1</n1:nextTenderStatusPollSeconds>
4 <n1:requestReference>16PK71437462993070</n1:requestReference>
5 <n1:state>TENDER_CREATED</n1:state>
6 <n1:tenderReference>16PK71437462986069</n1:tenderReference>
7 </n1:response>

```

Listing 5.11: Contents of message 11 (`statusTenderResponse`)

Table 5.1 shows the contents of the the elements `state` and `requestReference` for messages 11, 12, 13 and 14. The contents of the remaining elements did not change. The states clearly outline the progress of the transaction process. In message 12, the card reader notifies the Payleven app that a card has been inserted. Then, depending on the payment card, the corresponding payment application (e.g., Maestro) is selected. The card reader keeps the Payleven app up to date regarding the selection of the payment application with messages 13 and 14.

The request reference number varies for each message and serves probably as a sequence number. The last digit is increased in each subsequent `statusTenderResponse` message. The 15th digit of the reference number is in a few cases also incremented (see underlined digit in Table 5.1).

| | | |
|------------|---|--|
| Message 11 | <code>state</code> <code>requestReference</code> | TENDER_CREATED 16PK71437462993070 |
| Message 12 | <code>state</code> <code>requestReference</code> | CARD_INSERTED 16PK7143746299 <u>4</u> 071 |
| Message 13 | <code>state</code> <code>requestReference</code> | WAIT_FOR_APP_SELECTION 16PK71437462997072 |
| Message 14 | <code>state</code> <code>requestReference</code> | APPLICATION_SELECTED 16PK71437462997073 |

Table 5.1: The contents of elements `state` and `requestReference` for messages 11, 12, 13 and 14.

5.2.5. BLUETOOTH MESSAGE 15

Message 15 does not contain a response to the `statusTender` request. Its content is similar to the contents of message 7. The message is composed of the same `jaasToken` and authentication code as listed respectively in Listing 5.5 and Listing 5.7. Furthermore, the message contains the terminal ID of the card reader and provides the Payleven app with the URL: `'https://pal-live.adyen.com/pal/adapter/posmessage.proto'`. This URL has been observed earlier during the Internet network analysis in Section 4.5.3. What stood out about the `'posmessage.proto'` URL is that all that sent to it was encrypted. And indeed, the message(s) that follow(s) message 15 contain encrypted data.

Also noteworthy is the fact that the content-length fragment `cl` is set to `-1`. This suggest an upcoming message of length `-1`, which is an unrealistic attribute.

```
1 jaasToken=
  NjU4ZDg1ZmYtM2VlMi00YjQ5LTg5ZDEtMTE3YWNIjE3YWNINjc0NmYzXk1xcUd1aFQ5bmFNcDQ2UW1j
  =&terminalId=Shuttle-016303651&Authorization=Basic
  OTI4KioqKkBDdb21wYW55Lk1lcmNoYW50QWNjb3VudC42NTNLbHUqKioqKioqOg&#cl
  =-1&#url=https://pal-live.adyen.com/pal/adapter/posmessage.proto
```

Listing 5.12: Contents of message 15

5.2.6. ENCRYPTED DATA EXCHANGE (1)

After message 15 follow two messages that contain application-level encrypted data. This data is partially listed in Listing 5.13. This encrypted message exchange only covers two messages: the card reader sends a message with 1222 bytes of encrypted application data and the smartphone then replies with 101 bytes of encrypted data.

The encrypted data sent here to the Payleven app is ultimately forwarded to the Adyen back-end in web request A3a (see Section 4.5).

```

1 02 0c 20 e7 03 e3 03 43 00 09 ef bc 07 05 6c 82 .. ....C.....l.
2 10 00 00 01 01 00 26 0a 15 d8 6f 40 08 aa 79 81 .....&...o@..y.
3 0e 6e bf cc ea c6 ec 8f e2 1b d4 ac 75 4f 77 e4 .n.....uOw.
4 17 f7 9f f7 9c 66 77 74 73 db 58 c8 a4 32 7b 88 .....fwts.X..2{.
5 10 0e dd a5 bc 41 cd cd c8 53 f7 4a ee 10 b2 27 .....A...S.J...'.
6 27 da 7a 63 eb e5 62 15 08 6d 70 a9 8b d8 3a 84 '.zc..b..mp...:.
7 30 de e7 b6 52 bf 42 fb 49 cc 6b 70 52 e0 5b f8 0...R.B.I.kpR.[.
8 0d 87 9c 77 fc 30 d1 e0 fa 83 cd 82 d4 c4 e9 b5 ...w.0.....
9 48 45 37 ed 93 8b 80 c1 c2 45 40 17 ab 85 42 ad HE7.....E@...B.
10 f9 d0 d0 24 5c fc b5 a9 27 2e 33 6e 95 59 5a 4d ...$\...'.3n.YZM

```

Listing 5.13: Encrypted application data

5.2.7. BLUETOOTH MESSAGE 16

Figure 5.4 shows the second MSC diagram of the three-part. It continues where MSC 5.3 left off. The next message (message 16) is human-readable and contains a response to the `statusTender` request transmitted just prior to message 15. The response provides information on the state of the tender and the request reference. This information is presented in the table below.

| | | |
|------------|---------------------------|---|
| Message 16 | state requestReference | ADDITIONAL_DATA_AVAILABLE 16PK71437462998074 |
|------------|---------------------------|---|

Table 5.2: The contents of the elements `state` and `requestReference` for message 16

The state `ADDITIONAL_DATA_AVAILABLE` refers to an extra element `additionalData`, which holds 24 key entries. These key entries are listed in Table 5.3. The entries hold information on the inserted payment card, such as expiry date, issue number and country, card type and scheme. The card holder name is not listed, but an alias is provided instead. The card summary key holds the last four digits of the bank account number. Furthermore, the remaining keys hold information on the transaction, such as the transaction amount (listed twice), start date and time of the transaction, and the merchant reference.

5.2.8. BLUETOOTH MESSAGE 17 & 18

The Payleven app then reacts with message 17. The message contains a request to update the tender. The request is shown in Listing 5.14.

```

1 <nl:request>
2 <nl:merchantAccount>653Klu*****</nl:merchantAccount>

```

```

3 <n1:terminalId>Shuttle-016303651</n1:terminalId>
4 <n1:tenderReference>16PK71437462986069</n1:tenderReference>
5 <n1:modifyState>JustProcess</n1:modifyState>
6 <n1:process>>false</n1:process>
7 </n1:request>
    
```

Listing 5.14: SOAP request in message 17 (updateTender)

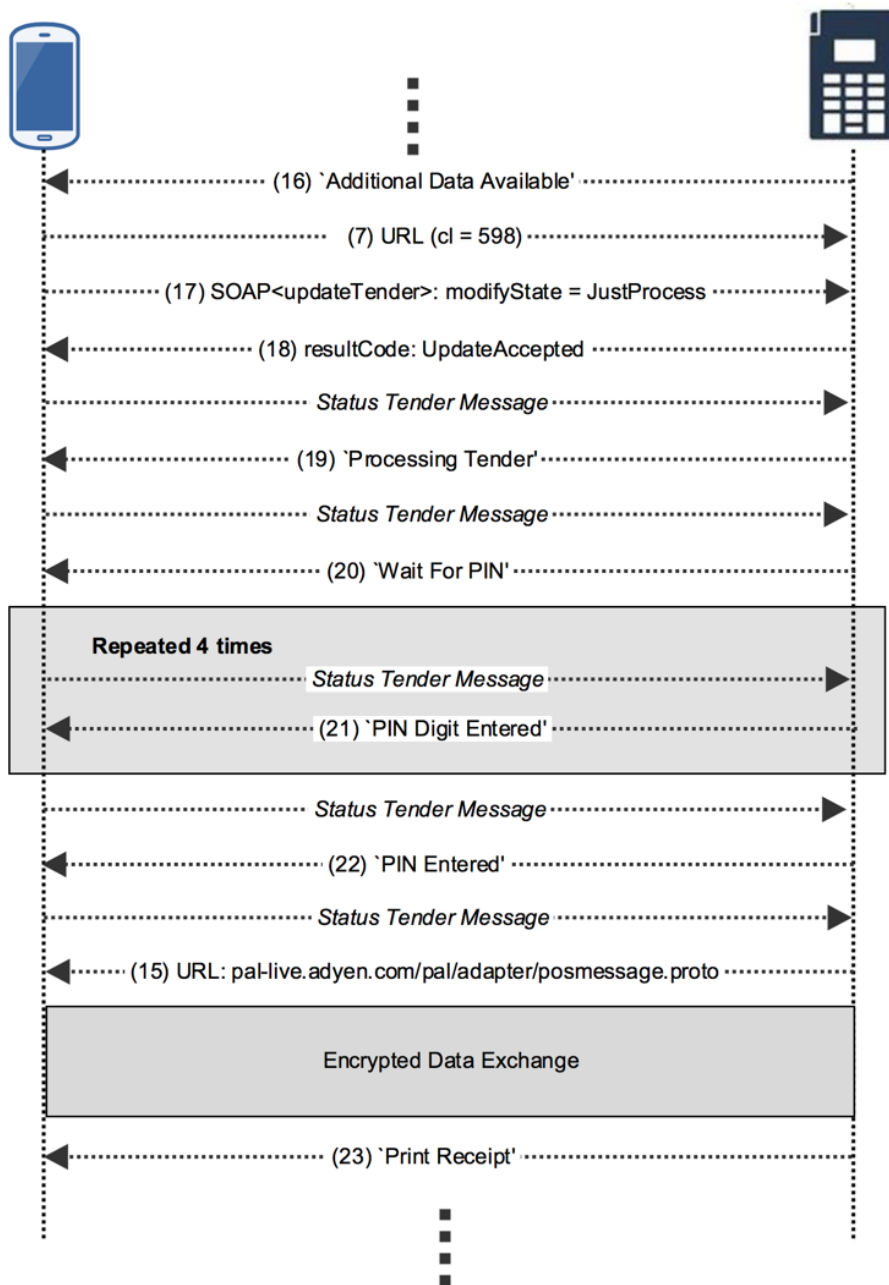


Figure 5.4: Message sequence chart of the Bluetooth traffic during the transaction phase. Part 2 of 3.

The goal of the SOAP request is probably to trigger the processing of the trans-

| | |
|--------------------------|----------------------|
| expiryYear | 2019 |
| cardIssueNumber | 02 |
| posAuthAmountValue | 100 |
| cardHolderName | empty |
| alias | C133084544881715 |
| cardSummary | 0537 |
| cardIssuerCountryId | 528 |
| merchantReference | de82fb6b2237210dbf17 |
| posAuthAmountCurrency | EUR |
| cardType | MAESTRO |
| cardScheme | maestro |
| txtime | 9:16:27 |
| applicationPreferredName | MAESTRO |
| transactionType | GOODS_SERVICES |
| startYear | 2014 |
| expiryMonth | 02 |
| AID | A0000000043060 |
| applicationLabel | MAESTRO |
| tid | 16303651 |
| posEntryMode | ICC |
| startMonth | 02 |
| txdate | 21-7-2015 |
| posOriginalAmountValue | 100 |
| cardBin | 673703 |

Table 5.3: Key entries listed in the `additionalData` element from message 16.

action by the card reader. This can be deduced from the element `modifyState`, which is set to the value `JustProcess`. However, the element `process` is set to 'false'. Therefore, the purpose of the update request not completely clear.

The card reader then replies with message 18 (see Listing 5.15) which contains the SOAP response. The response signifies that the tender update has been accepted.

```

1 <nl:requestReference>16PK71437463001075</nl:requestReference>
2 <nl:resultCode>UpdateAccepted</nl:resultCode>

```

Listing 5.15: SOAP response from message 18 (`updateTenderResponse`)

5.2.9. BLUETOOTH MESSAGE 19, 20, 21 & 22

Subsequent to message 17 and message 18, the Payleven app starts retransmitting 'Status Tender' messages to the card reader. The reader then replies with the messages 19, 20, 21 and 22. The `state` element of the messages, as shown in Table 5.4, gives a clear indication of the transaction state applicable at that time.

Message 19 indicates that the processing of the tender request (i.e., the transaction) has been initiated by the card reader. The reader then requests the card

| | | |
|------------|---------------------------|---|
| Message 19 | state requestReference | PROCESSING_TENDER 16PK71437463002076 |
| Message 20 | state requestReference | WAIT_FOR_PIN 16PK71437463002077 |
| Message 21 | state requestReference | PIN_DIGIT_ENTERED 16PK71437463003078 16PK71437463008079 16PK71437463008080 16PK71437463008081 |
| Message 22 | state requestReference | PIN_ENTERED 16PK71437463009082 |

Table 5.4: The contents of elements `state` and `requestReference` for messages 19, 20, 21, 22.

holder to enter the PIN in order to continue the process and authenticate the transaction. This is pointed out in message 20. For each PIN digit entered on the PIN terminal, message 21 is generated and sent to the Payleven app. Hence that the request reference is listed four times for message 21 in Table 5.4. This is also reflected in MSC 5.4. Message 22 is transmitted once the PIN is entered and confirmed.

5.2.10. ENCRYPTED DATA EXCHANGE (2)

Subsequently, the Payleven app sends a ‘Status Tender’ message to the card reader. The card reader then replies with message 15. What then follows is an exchange of encrypted data as has already been observed earlier in the protocol run in Encrypted Data Exchange - 1. This time, the card readers sends 1914 bytes of encrypted application data. The Payleven app then replies with 742 bytes of encrypted data. The data sent here to the Payleven app is ultimately forwarded to Adyen in web request A4a (see Section 4.5).

This is probably the most important message exchange of the whole protocol run since it occurs directly after the PIN was entered by the customer. The message exchange probably holds interesting and sensitive information regarding the handling of the transaction.

Also, it should be noted that each message 15 triggers an encrypted data exchange between the Payleven app and the Chip & PIN card reader.

5.2.11. BLUETOOTH MESSAGE 23

Message 23 holds the `statusTenderResponse` on the latest ‘Status Tender’ request sent prior to the encrypted data exchange. Table 5.6 shows the values for the `state` and `requestReference` elements of message 23.

| | | |
|------------|---------------------------|-------------------------------------|
| Message 23 | state requestReference | PRINT_RECEIPT 16PK71437463010083 |
|------------|---------------------------|-------------------------------------|

Table 5.5: The contents of elements `state` and `requestReference` for messages 23

Additionally, message 23 also lists the elements `additionalData`, `authCode`, `cardHolderReceipt` and `merchantReceipt`. The `additionalData` element holds the same key entries as listed in Table 5.3 and was supplemented with four extra key entries:

- `transactionReferenceNumber`;
- `signature.merchant.name1` and `signature.merchant.name2`, which hold the merchant company name and location;
- and, `paymentverificationdata`.

The key entry `paymentverificationdata` has been sent earlier in message 8 and was holding the term 'populate' then. In message 23, `paymentverificationdata` is populated with the following data:

```

1 <entry><key xsi:type="xsd:string">paymentverificationdata</key>
2 <value xsi:type="xsd:string">
3 BQABAQCF6Aopma3OeaKR0evqqNls+x7b+xzpOsmM+2fkyu9Ewr4Z1GE17Z+O2bl/
4 6njgQWVpPsPYppK5bTM3bxrGs00f90jiCfdPr5ppsiU40r8Gn9b01f++NsxSPrH
5 eoa+Jkv7QXs2PB0PYcQkUmm+MbqzbB0o37EMiN/jeqj4nWmziPzNAxCFqaEZbuF
6 Yy4+1phhK7ULTOT5AVAk8o9nq44ZpNUnPvjxpbFGjSJlvKo0alsuupTKo1JoHZh
7 wZBVHdl2x1cv0aCtik0e8XVffOBi2hoffwX/2MUvARa0sBvullutfm8fbzfBtFD
8 IPy4jJknCU7ROgZtbZ2F1LFIPeyT+709EPb6SeDrCuNuUpeateUmNvMAADYodiJ
9 P8RCf0K+obdOgPSGH+13MM/kNWDxa9Z0q/e4NLI3na5tS5ZJ15nRZZGJEodk7Id
10 jL6BhhgMhASFBBEbiUf0LuKe2oHzdB1SA5rjly+jbyhXeml6nQbo4j1ysBRGGsw
11 7rDwWjjw+GmdVknFE92WSeYL1ldZ25vAaqJsdecWkoKEN3UqaApX+3AcnPtf8Ug
12 </value>
13 </entry>

```

Listing 5.16: The value of `paymentverificationdata` in message 23.

Unfortunately, decoding the payment verification data according to Base64 does not reveal any human-readable information. It is suspected that the payment verification data is also subjected to application-level encryption.

The element `authCode` contains a 6-digit alphanumeric authorization code: in this case 26M9M2. The authorization code is different for each transaction.

The remaining elements, `cardHolderReceipt` and `merchantReceipt`, represented the receipt information. The receipt information is generated by the Payleven Chip & PIN card reader upon transaction completion and is structured according to the following syntax:

```

1 <n1:cardHolderReceipt>
2 <n1:content>
3   <n1:ReceiptLine>
4     <n1:format>Normal</n1:format>
5     <n1:mustPrint>>false</n1:mustPrint>
6     <n1:name>Datum</n1:name>
7     <n1:position>1</n1:position>
8     <n1:value>21-7-2015</n1:value>
9   </n1:ReceiptLine>

```

Listing 5.17: Structure of `cardHolderReceipt` in message 23

Each line on the receipt is defined by the elements enclosed in `ReceiptLine`. The `position` element indicates the position of the receipt line. The first line on the receipt for the card holder holds the date of the transaction. The elements `format` and `mustPrint` are self-explanatory.

Figure 5.5 holds the last part of the MSC representing the Bluetooth message exchange during the transaction phase.

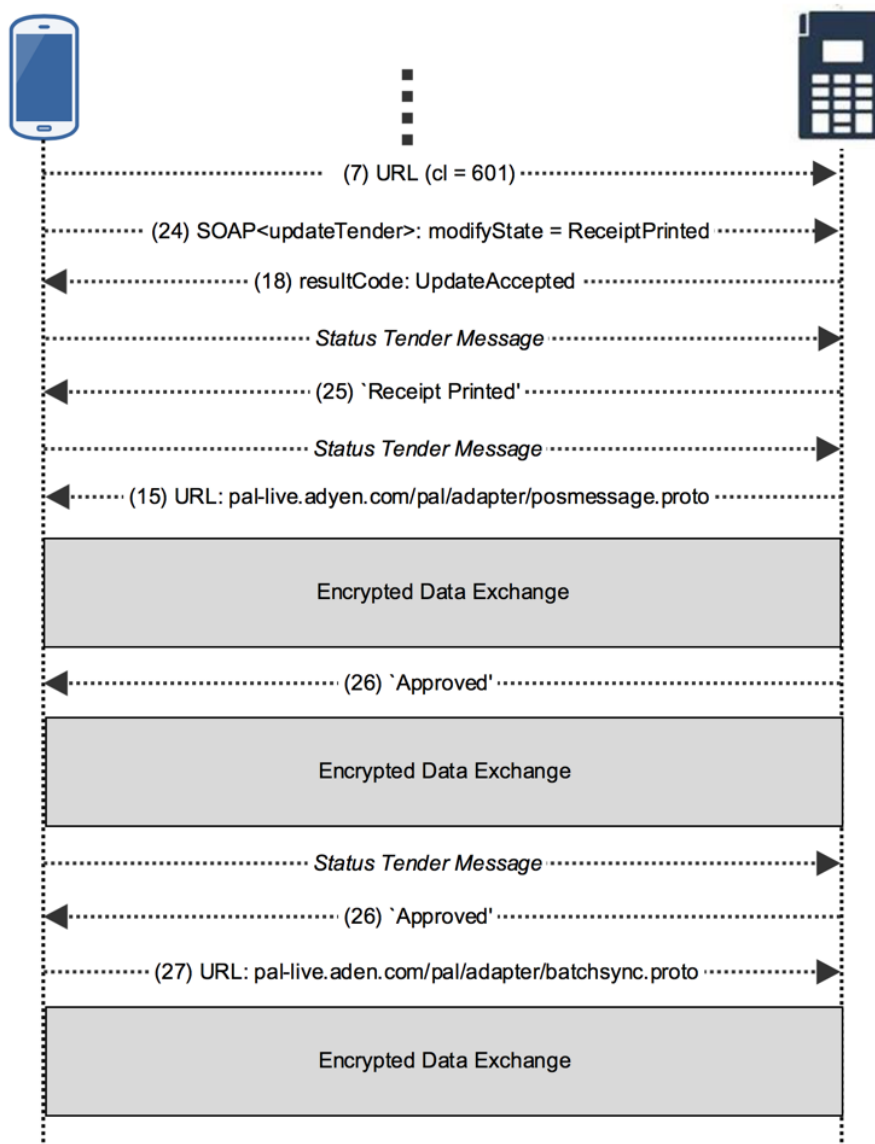


Figure 5.5: Message sequence chart of the Bluetooth traffic during the transaction phase. Part 3 of 3.

5.2.12. BLUETOOTH MESSAGE 24, 25 & ENCRYPTED DATA EXCHANGE (3)

The Payleven app then proceeds the protocol run with transmitting an `updateTender` request in message 24. The `modifyState` parameter was set to `receiptPrinted`:

```
1 <n1:modifyState>ReceiptPrinted</n1:modifyState>
```

Listing 5.18: `modifyState` in message 24 (`updateTender`)

The message probably serves as a confirmation to the card reader that the receipt has been received. The reader then replies with message 18 as described earlier in Listing 5.15.

Message 25 contains a response on the ‘Status Tender’ request. The authentication code (`authCode`) is added to the standard layout of a `statusTenderResponse`. With this message, the card reader probably acknowledges the fact that the receipt has been received by the Payleven app.

| | | |
|------------|---------------------------------------|---|
| Message 25 | state requestReference authCode | RECEIPT_PRINTED 16PK71437463016085 26M9M2 |
| Message 26 | state requestReference authCode | APPROVED 16PK71437463017086 26M9M2 |

Table 5.6: The contents of elements `state`, `requestReference` and `authCode` for messages 25 and 26.

Then again, message 15 is retransmitted to the Payleven app, followed by 13kB of encrypted application data.

5.2.13. BLUETOOTH MESSAGE 26, 27 & ENCRYPTED DATA EXCHANGE (4, 5)

The next message after the encrypted data exchange is message 26, which holds a `statusTenderResponse`. The `additionalData` element in message 26 holds the same key entries as message 23 (except for `signature.merchant.name`). The contents of the payment verification data entry did not change.

Message 26 is then followed by another encrypted message exchange, which in turn is followed by another message 26. The last human-readable message is message 27 and it is similar to message 15; only the last part of the message, the URL, is different:

```
1 jaasToken=
  NDikMWM1NmEtY2QxZi00N2ZjLWlZjctNmEzZjZhN2RiZiYwXk1xcUd1aFQ5bmFNcDQ2Qmc
  %3D&terminalId=Shuttle-016303651&Authorization=Basic+
  OTI4NzM2QENvbXBhbncuTWVvY2hhbnRBY2NvdW50LjY1M0tsdU1lZG9QT1M6&#c1=-1&#
  url=https://2Fpal-live.adyen.com/pal/adapter/batchsync.proto
```

Listing 5.19: Contents of message 27

The protocol run is then concluded with another encrypted message exchange. The last part of the URL-path (`batchsync.proto`) suggests that the last encrypted messages must contain batch data. What data exactly is synchronized during the batch synchronization, is unknown. However, the batch must hold a set of requests from current (and maybe previous) transactions which need to be executed or communicated to another party all at once.

6

SOFTWARE & HARDWARE ANALYSIS

The first part of this chapter focuses on the Payleven app; the source code is examined and findings are presented. The second part is aimed at the Payleven Chip & PIN card reader; the hardware is tested for its temper resistance and the reader's firmware is extracted and scrutinized.

6.1. ANALYZING THE APP'S SOURCE CODE

The Payleven app forms an essential part of the Payleven payment platform. It handles the communication between the Chip & PIN card reader and the Payleven and Adyen back-end servers. The app also provides the user interface to the payment platform. In essence, the Payleven app, and the smartphone, function as some sort of 'trusted' intermediary for sensitive transaction data. The app has to be trusted up to a certain extent as it passes on account information to the card reader on which, in the end, the transactions are based.¹ Therefore, the Payleven app and its source code had to be examined.

Unfortunately, it soon turned out that the Payleven app was too big to be investigated thoroughly within a short period of time. The Payleven app comprised an astonishing 5000 JAVA classes distributed over roughly 200 packages. It was not possible to explore every aspect of the app and to set out its workings in more detail. It simply comprised too many classes, which in terms of functionality also seemed to overlap. This and the fact that multiple parties are involved made it hard assert any interesting and valuable claims as regards security and operations.

The initial aim of this software analysis was (1) to uncover security flaws and privacy issues, and (2) to provide useful insights regarding the structure of the Payleven app. However, the focus of the analysis had to be narrowed. In the end,

¹note that the app is not trusted with the actual transactions data as it has not access. On the other hand, the card reader is trusted.

the software analysis of the source code primarily provided insights at a global level. That is to say, the source code of the Payleven app was consulted when necessary in order to fill in knowledge gaps which could not be explained by the information provided by the network and Bluetooth traffic.

This does not mean that the Payleven app has not been addressed. Some observations regarding the program contents are made in the next subsection.

6.1.1. PROGRAM UNDERSTANDING

The large number of classes and packages made it very difficult to identify the logic behind the Payleven app. Based on a first glance of the source code, it can be stated that a great number of the classes did not directly contribute to the main core of the program. The main core is defined as the security relevant parts responsible for the initialization and processing of transactions and other sensitive information, and the establishment and handling of the Bluetooth and network connection.

Eventually three packages were identified that could hold relevant classes covering some core activities.

- **adyen** - the 'adyen' package forms probably the vital part of the Payleven app. It contains the Adyen API which encompasses roughly 400 classes. Moreover, the API comprises classes aimed at the card reader and app registration, the communication towards the card reader and the payment server, SOAP message transport and handling, and the transaction API.
- **evopay** - Evopay is the former name of Payleven. The corresponding package contains folders and classes that seem to cover various basic functionalities of the Payleven app. That is, 'evopay' holds source code concerning the user interface, communication handling (e.g., contains HTTP client), tokens, payment stubs, tracking and also comprises the logic behind the handling of different types of card readers, banking cards applications and transactions. Large part of the 'evopay' package seems to concern legacy source code.
- **payleven** - the 'payleven' package is relatively small and contains classes that seem to add Chip & PIN functionalities to the Payleven app. The possibility exists that this package has been added later on to support new functionalities. The earlier versions of the Payleven app seem to only support mag-stripe functionalities

Other interesting observation that emerged from the analysis is that functionalities seem to overlap among different packages. For example, the JAVA-file *TransactionData.java* appears both in 'evopay' and 'adyen'. Which class is then utilized to represent transaction data? Both packages seem to overlap in certain functionalities, as indicated here; however, put against each other, the packages seem to act complementary. The Adyen package is aimed at the Adyen related functionalities of the Payleven app; for example, the registration of the Chip & PIN card reader and the Payleven app, whereas the other two packages provide functionalities that relate to the responsibilities of Payleven.

6.1.2. CLIENT-SIDE ENCRYPTION KEY

During the app registration process (see Section 4.3.1), the Payleven app receives a *client side encryption key* via a SOAP response from Adyen. It was not possible to deduce from the SOAP message for which purposes the encryption key is used. Of course, the possibility existed that the encryption key could be used for the application-level encrypted data exchange (see Section 4.1). If that would be the case, the application-level encryption would be easily circumvented. The next logical step was to search all files in the 'adyen' directory for the keyword 'clientsideencryptionkey' in the hope that the aim would then be apparent from the source code. The keyword was found in 6 JAVA-files:

- LibraryReal.java in 'library/real/';
- AppRegistrationTask.java in 'library/real/tasks/';
- Preferences.java in 'adyenpos/generic/';
- RegisterAppResponse.java in 'services/posregistration/';
- RegisterAppResponse.java in 'services/posregistrationsync/';
- and, RunMotoPayment.java in 'adyenpos/transactionapi/emv/processing/'.

In the method *createMotoPaymentRequest()* in RunMotoPayment.java the client side encryption key is used as follows:

```
1 context.setEncryptedCardData((new AdyenEncrypter(preferences.  
    getClientSideEncryptionKey()).encrypt(jsonObject.toString())));
```

The encryption key is loaded in the AdyenEncrypter class (see above code snippet). From *AdyenEncrypter.java* ('library/clientencryption') it became clear that the client side encryption key represents a public RSA key. The `encrypt()` method in that class subsequently generates a 256-bit AES key and uses it to encrypt the method's input (in this case, a JSON-object). The RSA key is used to encrypt the freshly generated AES key. The output of the `encrypt()` method then consists of an AES-encrypted JSON-object and a RSA-encrypted AES key.

This part of the Payleven app would be interesting to attack. By making the AES and RSA known fixed values, it would be possible to decrypt data that is encrypted with the above-mentioned encryption method. However, this encryption method is only invoked in RunMotoPayment.java and, unfortunately, this class is never put into operation by the Payleven app. The acronym MOTO stands for 'Mail Order/Telephone Order' and, thus, a MOTO payment is a payment in response to an order made via the telephone or mail (so called virtual POS)[24]. In conclusion, the client side encryption key sent during the app registration phase is used to encrypt an AES key which, in turn, was used to encrypt the transaction and card details within a MOTO-payment. These payments are never invoked in the Payleven payment platform.

6.2. ANALYZING THE CHIP & PIN CARD READER

The security evaluation of the Chip & PIN card reader consisted of two parts. The first part was aimed at the hardware of the card reader; a closer look was taken at the internals. The second part focused on the extraction of the firmware from the reader and its evaluation. The goal of the hardware evaluation was to investigate how tamper-resistant the card reader actually is. As slightly pointed out during the methodology in Chapter 3, there exists the threat that the reader is tampered with. For example, a small device could be planted that registers and stores keystrokes, or the reader's display is manipulated. Therefore, the hardware was taken apart and the possibilities for tampering were investigated. As regards the firmware, it was first necessary to extract all files located on the card reader. This seemed struggling as the card reader did not appear as a local drive while plugged-in on a USB port. Eventually, it was possible to extract all files through the methods described in Section 3.3.4. Manual investigation was then used to sift through all detected files, seeking for striking or sensitive information and files. All file were subjected to examination.

The next section deals with the evaluation of the hardware, whereas the subsections thereafter discuss the findings of the evaluation on the file system and firmware found on the card reader.

6.2.1. DISSEMBLING THE CARD READER

Taking apart the card reader was far from easy. It was clearly built to never be opened again. No bolts were used to fasten the plastic casings, thus force was required to loosen the casing. In the end, with some patience and a lot of force, the backside became detached.



(a) The backside of the card reader without plastic casing. (b) The frontside of the card reader without plastic casing.

Figure 6.1: The Payleven Chip & PIN card reader with the plastic casing removed.

Figure 6.1a shows the backside of the card reader without the plastic casing. The battery (silver) and the smart card slot are exposed. The exposed frontside is shown in Figure 6.1b. The keypad and the display can clearly be seen. The top of the card reader comprises a magnetic strip reader.

Subsequently, the card reader was further taken apart. The remaining plastic was removed and the battery was detached from the backside. The battery was welded on two points on the backside and needed to be cut loose. This caused an electrical short cut which rendered the device useless. Figure 6.2a showcases the card reader without the battery. The integrated circuit of the card reader is clearly visible, as are the processor, memory, and other chipsets. However, no JTAG port was detected on the main board.



(a) The internals of the card reader demonstrating the integrated circuit. (b) The magnetic strip reader is loosened from the card reader.

Figure 6.2: The card reader is further torn apart. The battery and magnetic strip reader are removed.



Figure 6.3: The message on the display indicates that the card reader has been tampered with.

Prior to removing the battery, the card reader was alive; however, it was still useless. The Chip & PIN card reader immediately noticed that the plastic casing was removed. Once the casing was detached, the display on the card reader showed the message 'xx SYSTEM TAMPERED xx', thus indicating that it has detected an attempt to compromise security. The tamper attempt has probably been detected by detection points on the keypad. The plastic front case namely has small pins on the inside that exert pressure on specific points on the main board. By removing the front case, the pressure disappears and the card reader shuts down and ends up in an infinite error state. That is, it is not possible for the card reader to exit the error state. Restarting the device is useless.

In conclusion, opening the Payleven Chip & PIN card reader carelessly will render the device useless. This makes it unfeasible for an attacker to apply modifications on the internals of the device. Moreover, the size and volume of the card reader leaves little room for the emergence of other attack possibilities. The only weakness of the hardware as became apparent in this research was the fact that it allowed for the extraction of the firmware.

6.2.2. EXTRACTING THE FILE SYSTEM

The card reader's file system was extracted and copied into an ISO-file (approx. 33.6MB) using the methodology as described in Section 3.3.4. Subsequently, 'Binwalk' was then used to identify and extract separate individual files from the ISO-file. The extraction ended up with some folders, a few shell scripts, binaries, certificates, configuration files, .txt-files and JAVA .properties-files. A selection of these findings is listed below:

- .txt-files:
 - **cardbin.txt**: lists range of allowed bank identification numbers (BIN) together with card type and issuing country code;
 - **cardReferral.txt**: lists three short Base64 encoded strings.;
 - **capkeys.txt**: lists the CAPKI and RID. The CAPKI stands for Certification Authority Public Key Index. It identifies the certificate authorities public key in combination with the RID. The RID is the Registered Application Provider Identifier which is allocated to each card scheme to identify the corresponding EMV application. It seems that this text file stores the public keys for each RID in order to support Offline Enciphered PIN [25][26].
- Certificates:
 - **main-user.crt**: signed by Adyen and issued by Miura Systems (i.e., card reader manufacturer);
 - **terminal.crt**;
 - **prod-sign.crt**.

- Shell scripts:
 - **start**: defines the start procedure during boot time;
 - **restore**: defines restore procedure;
 - **install**: defines the install procedure. Probably the more interesting script as it demonstrates what steps are invoked to be completed or executed before the system is considered to be ready. For example, during the install procedure several certificates are verified on the device.
- JAVA `.properties` files:
 - **terminal.properties**: lists the supported terminal capabilities of the card reader, such as manual key entry and online/offline enciphered PIN handling.
 - **global.properties**: lists the global properties of the firmware. It lists, for example, the local time, location, used currency. It also defines the `posmessage.proto` and `batchsync.proto` URLs as encountered during the network analysis in Section 4.4 and 4.5. The most interesting part of this file concerned four PIN codes held in plaintext. The respective piece from the `.properties`-file is shown below in Listing 6.1. These PIN codes are presumably used to access an admin menu or setup on the card reader. This is indeed the case as will be demonstrated later on in Section 6.2.4.

Moreover, the four admin PIN codes are, as it seems, sorted based on access rights. The ‘adyen’ clearance level is probably the highest (i.e., level 1) and the ‘operator’ level is the lowest (i.e., level 4). Indeed, this is true as the ‘operator’ PIN code will result in less options in the admin menu than when accessed with the ‘adyen’ PIN code. Section 6.2.4 describes the hidden admin menu on the card reader.

```
0  setup.clearance.adyen.pin = ***** (11 digits)
1  setup.clearance.adminplus.pin = ***** (12 digits)
2  setup.clearance.admin.pin = **** (4 digits)
3  setup.clearance.operator.pin = **** (4 digits)
4
```

Listing 6.1: The discovered admin PIN codes in ‘global.properties’. The PIN codes are removed for security reasons.

- Folders:
 - **icons**: comprises the icons that are displayed on the card reader screen, such as the Payleven logo;
 - **fonts**: lists several fonts in BDF format;
 - **adyen**: lists several JAR-files. Obviously, this is the most interesting folder as it contains the application. The JAR-archives ‘miura-sup’ and ‘miurajni’ hold the firmware of the card reader. The archive

'protobuf-java-2.4.1.jar' was also included in the folder. This confirms that Google's Protocol Buffers are used within the Payleven payment platform.

- remaining folders: **bin**, **fs**, **META-INF** and **receipts**.

Although a clear file system including firmware could be extracted from the card reader, a large part of the extracted ISO-file was as it seemed obfuscated (entropy approached the value 1.0). Binwalk was not able to identify or extract any useful files from that portion of the ISO-file. This could be an indication of encryption. The extracted file system did not contain any secret keys, hence it might be the case that these keys were situated within the obfuscated part. This stresses the fact that some kind of security measurement was implemented on the card reader to protect sensitive data. Further research is required to investigate this finding.

6.2.3. COMMENTS REGARDING THE FIRMWARE

As mentioned in the previous section, the two JAR-archives 'miurasup' and 'miurajni' found in the folder 'adyen' represented the firmware of the card reader. A quick analysis of the source code of the firmware identified several functionalities; for example, the firmware comprised source code for the Bluetooth functionality, the (SOAP) message and transport handling, the card reader registration and the payment handling. Basically, similar functionalities as encountered in the analysis of the Payleven app. However, the firmware did also include source code which covered EMV and cryptography functionalities. The firmware clearly covers the 'core' functionalities of the payment system, which is obvious as the card reader plays a central role within the system.

The examination of the firmware did further not reveal any security-related issues, however it should be noted that the examination was far from complete and comprehensive. The firmware did comprise many files and was definitely too complex to be understood within a short period of time. Moreover, far too little time was spend into examining the firmware.

Nevertheless, some notions have been made on basis of a first view analysis on the firmware:

- A significant part of the firmware is provided or co-developed by Adyen. To underline this observation, no occurrences of the string 'Payleven' were found in the firmware. This stresses the fact that Payleven was not involved in the development of the payment system. More importantly, it points out the significance of Adyen to the Payleven payment platform.
- The firmware clearly pointed out the implementation of the DUKPT key management scheme. Also, it was observed in the source code that several important (secret) keys are loaded from the card reader's file system. As slightly noted at the end of the previous section, a major part of the data

extracted from the card reader was obfuscated; that is to say, no useful information could be identified or extracted. Hence, it may be concluded that this part of the file system that contained these keys was obfuscated in order to ensure that sensitive keys are confiscated.

6.2.4. ACCESSING THE HIDDEN ADMIN MENU

As pointed out in the previous section, the card reader holds a 'global.properties' file which lists four admin PIN codes in plaintext. At first it was not clear how and where these PIN codes could be used, however, very soon the suspicion arose that a hidden admin menu had to exist on the Payleven Chip & PIN card reader. Therefore, the remaining question is how to evoke the menu on the card reader. As this is obviously not mentioned in the instructions guide of the Payleven payment platform, it had to be sought elsewhere. The firmware formed an obvious spot.

Finally, the firmware was searched for occurrences of the string 'setup.clearance'. Eventually, the search ended up in *MainActivity.java* and *SetupAuthActivity.java*:

```
516 private boolean key9()
517 {
518     this.console.setBacklight(true);
519     new SetupAuthActivity().waitDone();
520     return true;
521 }
```

Listing 6.2: The `key9()` method in *MainActivity.java*

```
20 public SetupAuthActivity()
21 {
22     this.dpy = new ScreenTextInput(getConsole(), new PinInputFormatter
23     (), I18NUtil.getI18NString("enter.admin.pin"), true);
24     this.pins = getClearanceLevels();
25     this.dpy.setInput("");
26     this.dpy.setValidator(new AdminPinValidator(null));
27     this.dpy.display();
28 }
29 [...]
30
31 private Map<String, String> getClearanceLevels()
32 {
33     String levels = Configuration.getString("setup.clearance.levels");
34     [...]
```

Listing 6.3: The `SetupAuthActivity()` method in *SetupAuthActivity.java*. The string 'setup.clearance' was found in this JAVA-file (line 33).

Method `key9()` suggests that key '9' is involved in the handling of the detected admin PIN codes. Indeed, holding the '9' on the reader's keypad evokes the admin menu as shown in Figure 6.4. Subsequently, the card reader prompts for the admin PIN code. The four admin PIN codes cover different clearance levels as noted previously. Hence, the setup of the admin menu differed slightly for each PIN code. Figure 6.4b shows the menu for the lowest clearance level.



Figure 6.4: The admin menu. (a) The card reader prompts for the admin PIN. (b) The upper menu items (4 items).

The admin menu consisted of the following menu items:

1. Bluetooth (1, 2, 3, 4)
2. Show report (1, 3, 4)
3. Synchroniseren (1, 2, 3)
4. Show debug info (1)
5. Keypad test (1)
6. Export log (1, 2, 3)
7. MSD Update (1, 2, 3)
8. Delete Settings (1, 2, 3)
9. Device Info (1, 2, 3)
10. Set the Clock (1, 2, 3, 4)
11. Key Load (1, 2, 3, 4)
12. Leave Setup (1, 2, 3, 4)

The numbers inside the parentheses indicate the clearance level required to access the menu item in question. The highest clearance level is '1', the lowest '4'.

The 'Key Load' option is an interesting one as it suggests that it enables users to load a custom key on the card reader. However, this is not the case. If selected, the card reader tries to set up an encrypted channel with the Adyen back-end servers to download the new key. Therefore, the 'Key Load' option requires that the card reader is connected to a smartphone with a network connection.

By selecting the 'MSD Update' option, the card reader showed up as a USB drive (no files are shown). The objective was then to copy an 'update.dat' file into the USB drive window. The card reader will then try to update its firmware

based on the loaded `.dat`-file. It is assumed that `'update.dat'` needs to be signed in order to be accepted, hence uploading malicious `.dat`-files would not be possible.

The remaining menu items were not very interesting except for `'Export Log'`. The `'Export Log'` option generated a zip-file containing several log-files. The card reader then showed up again as a USB drive. This time, the window included the zip-file `'logs-Shuttle-016303651.zip'`. Extracting the zip-file resulted in the following log-files:

```
zip-file
├─ adyen.properties
├─ aeskey
├─ alog
├─ alog.1
├─ alog.2
├─ App-version.txt
├─ bootstrap-files.log
├─ cpuinfo
├─ dmesg
├─ events.log
├─ global.properties
├─ main.log
├─ receipt.properties
├─ terminal.properties
```

Figure 6.5: The log-files contained in the `'Export Log'` zip-file.

Two observations stood out regarding the log-files: (1) the contents of all the log-files were encrypted, and (2) an AES key was included. The possibility exists that the provided AES key was used to encrypt the log-files. However, no attempt has been made yet to decrypt the log-files with the AES key. A quick test revealed that the IV or passphrase were missing. Moreover, the `.properties`-files listed above were also present in the ISO file that was extracted from the card reader.

7

RESULTS

This chapter presents the results of the security analysis by discussing three possible attack scenarios. These scenarios are based on various security issues identified within the Payleven payment platform. Additionally, this chapter also highlights and addresses these issues. Section 7.1 describes a theoretical attack in which payments are diverted by modifying the merchant account. Section 7.2 describes an attack in which payments are faked towards the Payleven back-end. Lastly, Section 7.3 describes an attack in which illegitimate refunds are triggered by brute-forcing the refund authorization PIN code.

7.1. SCENARIO 1: ALTERING THE ADYEN MERCHANT ACCOUNT

This section describes a theoretical attack scenario on the Payleven payment platform. The attack is aimed at diverting monetary transactions by substituting the genuine merchant account by the merchant account belonging to the attacker. In the end, this leads to monetary loss for the merchant and monetary gain for the attacker. Firstly, Section 7.1.1 will discuss the issue concerning the Adyen merchant account that led to this attack scenario. The section thereafter will elaborate on the attack scenario and its impact. The remaining sections will address the feasibility and traceability of the attack.

7.1.1. THE ISSUE CONCERNING THE ADYEN MERCHANT ACCOUNT

The Adyen merchant account plays a key role within the Payleven payment platform. In fact, it can be stated that the whole payment system revolves around the merchant account. It determines, in the end, at whose bank account the money

will be credited. Hence, it is to be expected that the merchant account is properly preserved against malicious practice.

Unfortunately, this is not the case. The corresponding merchant account is communicated by Payleven to the Payleven app during the app login phase based on the logged-in Payleven user account (see Section 4.2). This suggests that the Payleven user account and the Adyen merchant account are intrinsically linked. After all, the Payleven account is used to retrieve the corresponding merchant account, after which the merchant account is used as the merchant's main identifier throughout the payment process. However, this link is non-existent throughout the Payleven payment platform; only the Payleven back-end is aware of this link between the merchant account and the Payleven account. That is to say, the Payleven app cannot verify whether the merchant account actually corresponds to the Payleven account logged in at that moment. This is simply not possible. These two accounts, or instances, exist independently from each other within the payment platform. This means that one instance can be modified without affecting the other one. So, accordingly, what will happen if the merchant account is modified?

This is a concerning thought. However, the lack of verification measures on the Chip & PIN card reader is of greater concern. The card reader has no conception of the existence of a Payleven user account. The only identifier it receives from the app is the Adyen merchant account (see Section), which is, in turn, used by the card reader to process transactions with Adyen. This means that the card reader should be accepting any merchant account as it cannot verify whether the received merchant account is also the correct one (i.e., corresponds with the logged-in Payleven account). In other words, it is theoretically possible to change merchant account, and thus divert transaction away from the merchant, without the card reader or the app noticing it. After all, they cannot detect the modification since the link between the merchant account and the Payleven account is non-existent throughout the platform.

Indeed, the card reader will accept any merchant account fed to it, as long as the reader is not set up with an another merchant account; i.e., the card reader needs to be factory reseted. This has been observed in the Bluetooth traffic. However, it is expected that the modification will be detected and the transaction will be canceled due to the permanent linkage between the card reader and the merchant account. The card reader will eventually communicate its identification number (i.e., terminal ID) and the modified merchant account to Adyen. Consequently, Adyen will detect the mismatch as the terminal ID is not linked to that specific merchant account. It is linked to the original, pre-modification, merchant account.

Although this may be true, it does not make an attack based on the above-mentioned findings completely impossible. It does however make the attack more difficult and more improbable to be put into practice. The next section will describe a theoretical attack scenario in which transactions are diverted successfully by replacing the genuine merchant account with the account belonging to the attacker (so-called malicious merchant account), and by replacing the genuine

card reader with the reader whose terminal ID is linked to the malicious merchant account (so-called malicious card reader).

7.1.2. ATTACK SCENARIO: DIVERTING TRANSACTIONS

Based on the notions discussed in Section 7.1.1, it is thus possible to conduct a serious attack on the Payleven payment platform. The fact that (1) the authenticity and integrity of the Adyen merchant account cannot be assured throughout the payment platform, and (2) the fact that the Chip & PIN card reader is not capable of verifying whether the received Adyen merchant account corresponds to the logged-in Payleven account, facilitate an attack on the payment system. A successful attack would consist of three steps:

1. Firstly, the attacker would need to substitute the genuine merchant account with the malicious one. This could be accomplished in several ways: (1) via an MITM-attack on the network channel as applied in this research, (2) by substituting the Payleven app with a rogue variant, (3) or through mobile malware. In the end, the essence of this step is to ensure that the card reader is fed with the malicious merchant account.
2. Secondly, the attacker would need to swap out the genuine card reader. As pointed out in the previous section, the card reader will accept any merchant account, as long as the reader has been factory reset. However, eventually the Adyen back-end will detect the mismatch between the malicious merchant account and the genuine card reader as it is expected that the card reader will forward all the information it has to Adyen. Therefore, Adyen will cancel any attempted transaction based on the mismatched combination of terminal ID and merchant account. Evidently, this linkage between the card reader and merchant account is a good thing. It is clearly intended to prevent abuse, and indeed, it makes the attack a lot less feasible and practical. The attacker is forced to physically replace the merchant's genuine card reader with a different card reader whose terminal ID is linked to the malicious merchant account, and all this without the merchant noticing. The feasibility of this is discussed in 7.1.4.
3. Lastly, all outbound communication towards the Payleven back-end needs to be altered in such way that neither the malicious merchant account nor the malicious card reader are referred to in the logs. Instead, the original merchant account and the original terminal ID should be used. This way both Payleven and the merchant are fooled to believe that the transactions were executed flawlessly.

Interestingly, the merchant account is never sent towards the Payleven back-end, except for once during the app initialization phase for logging purposes. All the other times the *marketplace merchant ID* is used to refer to the merchant. However, it is unknown whether this ID is generated by Adyen or by Payleven. It is in any case not transmitted to the card reader.

In summary, all information destined for Payleven must refer to the original merchant account and card reader, whereas information destined for Adyen must refer to the malicious merchant account and the malicious card reader. This can be achieved in the same way the merchant account is altered.

In brief, an attacker would need to (1) replace the original merchant account with the malicious account, (2) replace the original card reader with the malicious reader, and (3) modify all outbound communication towards Payleven to go unnoticed. As a result, a transaction conducted on the payment system will be diverted to the attacker's bank account. This all happens under the surface, whereby the merchant only has a visibility of this surface. Neither Payleven nor the merchant will be aware of anything since all information towards Payleven and the Payleven dashboard is manipulated. The merchant will only notice after a few days as there is no money credited. Even Adyen would not notice anything erroneous as it receives seemingly legitimate data from the card reader. This clearly points out the issue here caused by the lack of cooperation between the two instances within the Payleven payment platform.

7.1.3. TRACEABILITY OF THE ATTACK

As has been slightly addressed in the previous sections, Payleven would not notice the attack. Payleven would not notice that the payment transactions were diverted to a merchant account other than the one associated with the logged-in Payleven account. At least, not in the first days; payments are credited on the merchant's bank account after two or three business days. Only after these days have passed - maybe a week - the merchant would notice that no money is coming in. The merchant would then contact Payleven and raise the problem. Payleven, in turn, would contact Adyen, since they are responsible for the payment processing. As regards the communication between Payleven and Adyen, its contents are unknown and each assumption about the content would be based on pure speculations. In that case, it is important to determine whether the attack can be traced based on the information that is sent to the Chip & PIN card reader, as the card reader probably forwards all received information to Adyen.

All traffic going back and forth between the card reader and the Payleven app, and the Payleven app and the Internet was set out in the Chapters 4 and 5. Any information that was sent to the card reader, but originated from the Payleven back-end will raise suspicion, because this information could, in turn, be sent to Adyen via the encrypted channel. This would then link the transaction known to Payleven to the malicious transaction known to Adyen. A call to Adyen would then immediately result in finding the attacker.

Indeed, such information is exchanged. Payleven generates and sends a payment stub (20 bytes) as a response to web request P4a (first web request in the transaction process, Section 4.5). This payment stub is then forwarded in Bluetooth message 8 as a tender reference (first message to card reader in transaction process, Section 5.2). Thereafter, the payment stub is sent as a merchant ref-

reference code back to the Payleven app in Bluetooth message 16 (see Table 5.3). Finally, the payment stub is forwarded to the Payleven back-end together with other information regarding the completed transaction in web request P5a. In the transaction overview on the Payleven dashboard, the payment stub is displayed as the payment ID (Dutch: 'Betaling ID'). Figure 7.1 visualizes the exchange of the payment stub in the Payleven payment platform.

Therefore, the tender reference code sent to the card reader should be made different from the payment stub received by the Payleven app. Only this way it can be precluded that Adyen ever receives information that would link the transaction back to Payleven. Without this link, neither Payleven nor Adyen cannot ascertain to what merchant account the money was diverted to.

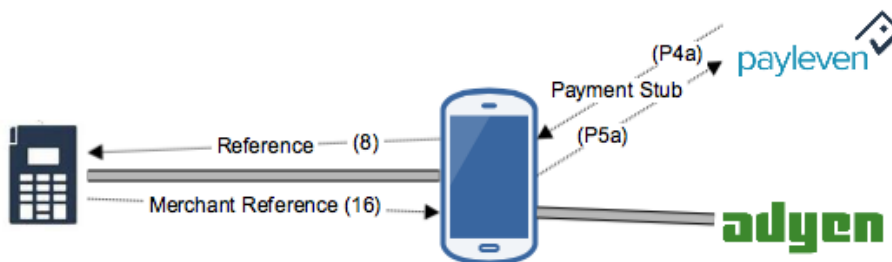


Figure 7.1: The exchange of the payment stub. First, the stub is transmitted to the Payleven app in the response to web request P4a. The stub is then included as a tender reference in Bluetooth message 8 and sent back as a merchant reference in Bluetooth message 16. Finally, the payment stub is included together with other transaction information in web request P5a. The changes are high that the payment stub is also communicated to Adyen via the encrypted channel.

It is further unknown whether Payleven communicates the payment stub to Adyen in some other way. However, this seems unlikely since the Payleven app never sends information to the Payleven back-end that may link the Adyen transaction to the payment stub.

Another issue that it has not been investigated is whether Adyen will accept the manipulated tender reference code. It could be possible that Adyen has a list of all payment stubs generated by Payleven. Then the manipulated tender reference would not be accepted that easily.

7.1.4. FEASIBILITY OF THE ATTACK

The feasibility of an attack on the merchant account is dependent on the feasibility of the three steps discussed previously in the attack scenario. Firstly, it depends on the extent to which the attacker is capable of altering the merchant account. Several methods to accomplish this have been referred to such as: (1) an MITM-attack on the network channel, (2) installing a rogue Payleven app on the merchant's smartphone, or (3) malware on the Android device (e.g., by means of Android Hooking). Secondly, it depends on how difficult it is to switch the original Chip & PIN card reader without the merchant noticing it.

The first method has been applied successfully in this research to intercept, gather and alter information sent back and forth over the network channel. However, a number of issues can be identified concerning this method. The Payleven app makes use of SSL certificate pinning, hence making a conventional MITM-attack impossible. In order to remove this security feature, a tool was installed on the Android smartphone that bypasses the SSL certificate pinning. The only downside: the Android smartphone needs to be rooted. As long as the merchant's Android smartphone is not rooted, the attack is unrealizable. And on top of that, an MITM-attack is very limited; it does not enable complete control over the app. For example, Bluetooth messages sent from the Payleven app cannot be modified via an MITM-attack. So, in the end, this method promises a quick and easy attack setup, but is otherwise limited and unpractical in real-life situations.

The second method is more practical and enables full control over the Payleven app. Obviously, replacing the original Payleven app by a rogue variant provides full access and control. The rogue variant behaves and acts like the original Payleven app: it exchanges information with Payleven and Adyen and pairs with the Chip & PIN card reader. The only difference with the original Payleven app is that the rogue variant manipulates specific outgoing information (i.e., from the smartphone) in such way that is it beneficial for the attacker. It implements all the features necessary for a successful and untraceable attack.

The question then arises how to get the rogue variant on the merchant's Android smartphone. There exists a number of ways to accomplish this. The rogue variant could be placed in the Google Play Store; however, it is then still questionable whether the merchant will download the rogue variant. Moreover, the rogue Payleven app would be probably detected immediately, either by Google or by Payleven. Then, the best options for an attacker to make the merchant install the rogue variant would be via malware or phishing. Through phishing the merchant could be urged to install the newest update of the Payleven app via an update link in the mail, which in the end would replace the genuine app by the rogue variant. Malware could have the same objective. But at the same time, once malware is installed on the target's smartphone, its rogue functions could easily be extended to directly attack specific information exchange of the genuine Payleven app. Therefore, the last method is related to malware targeting Android.

All things considered, there are plausible ways for an attacker to alter the merchant account code. The only measure left is the substitution of the genuine Chip & PIN card reader. The attack requires the genuine card reader to be replaced by an alternate reader that is linked to the malicious merchant account. Given the fact that the card reader is mobile and wireless, it is not hard to imagine that substituting the card reader is very easy. The only issue is that, once the card reader is replaced, the merchant would need to redo the Bluetooth pairing process. However, this would not raise any suspicion since this operation also had to be repeated every once in a while in an honest setup. The only possible way to verify whether the genuine card reader is used, is by checking the serial number on the back of the reader. However, it can be assumed that no merchant will memorize the serial number of the card reader.

7.1.5. THE IDEAL APPROACH

Reflecting on Section 7.1.4, it can be argued that switching the card reader in sight of the merchant is not necessarily needed. The same effect can namely be achieved by corrupting payment system from the start; that is, (in)directly selling the merchant the malicious card reader. This corrupted Payleven payment kit would be identical to the official genuine kit, including box and manual. Moreover, the corrupted kit could also contain a leaflet that lists the download link for the rogue Payleven app and the instructions to enter the link directly on the Android smartphone (instead using the Play Store). This attack scenario tackles both the issues described in the previous section regarding the feasibility of the attack; namely, how to get a rogue variant of the Payleven app on the merchant's Android smartphone, and how to switch the card reader without the merchant noticing it. Furthermore, in this case, the attacker does not need to figure out which merchant has downloaded the rogue app and does not need to physically go there to replace the card reader. The ideal scenario would thus consist of selling corrupt Payleven payment kits to merchants.

The only issue is that the merchant must eventually register the card reader during the registration phase. This would result in an error as the card reader is already linked to the attacker's merchant account. Therefore, the attacker must purchase two card readers, wherein one reader is linked to the attacker and the other is left unlinked. Subsequently, the attacker swaps out the stickers on the back of the card readers stating the serial number. This would lead the merchant to register the serial number of the unlinked card reader, but in the end, use the malicious card reader to process transactions.

7.1.6. EXAMINATION OF THE ATTACK

The possibilities of this attack have been investigated. Unfortunately, due to the lack of a second valid merchant account, it was not possible to fully validate the existence of this attack. It was not possible to test the attack scenario without a second account. However, a few things did become clear. For example, the merchant account was modified while in transit to the Payleven app and the app accepted the merchant account without any errors. Subsequently, the app used the modified merchant account during the whole login session. This also meant that the modified merchant account was forwarded to the Chip & PIN card reader. This has been observed in the Bluetooth traffic. It only went wrong on the part of the card reader. The reader was factory reseted, however since an non-existent merchant account was used - the account was simply altered - the card reader could easily detect the anomaly. This is because during the Bluetooth pairing process the reader receives an Adyen token (i.e., so called 'jaastoken') which is generated by the Adyen back-end. Since all communication towards the Adyen back-end was based on the original merchant account, the card reader could easily spot the discrepancy by comparing the jaastoken, which was based on the original

merchant account, to the altered merchant account. The card reader did thus reject the non-existent merchant account. However, if it concerned a valid account it would be accepted by the card reader as it does not receive any information from the app that could prove that the merchant account is modified.

7.2. SCENARIO 2: ALTERING TRANSACTION INFORMATION

The Payleven app feeds information regarding the transactions to the Payleven back-end servers (both `apiproxy` and `fapi`). This information is then included in the Payleven dashboard. As elaborated in Chapter 2, the dashboard lists all successful and unsuccessful transactions including the transaction amount, time stamp, both payment and transaction ID, and even the location of the transaction. Furthermore, the dashboard also provides the merchant the option to process refunds on individual transactions. All this information included in the dashboard, originates from the Payleven app. But what happens if all this information transmitted from the Payleven app to the Payleven back-end is modified while en route?

7.2.1. ALTERING THE TRANSACTION AMOUNT

As described in Section 3.3.2, it was possible to apply a successful MITM-attack between the Payleven app and the Payleven back-end. This attack gave full ‘read-and-write’ access to the information exchanged with the Payleven back-end. Therefore, it was very easy to modify certain data fields.

The first data field that was modified was the transaction amount value. During a legitimate transaction, all occurrences of the transaction amount were altered and replaced by a different amount. The transaction amount is communicated to Payleven in web requests P4a, log 4 and P5a (see Section 4.5).

All transactions were conducted with a transaction amount of €1 (i.e., the minimum starting value). The transaction amounts communicated to Payleven were subsequently altered and replaced by an amount of €10. It was expected that this illegitimate transaction amount would then be reflected in the transaction overview on the Payleven dashboard. And indeed, the transaction amount of the affected transactions included in the transaction overview comprised a value of €10 instead of the actual value of €1.

7.2.2. REFUNDING ALTERED TRANSACTIONS

It was then attempted to initiate a refund on the altered transaction. It was hoped that the €10 would be refunded. Unfortunately, and also slightly expected, each attempt to start a refund resulted in an unknown error message. The refund request sent to Payleven is probably forwarded to Adyen. Because Adyen receives genuine transaction information directly from the card reader via the encrypted channel, disparities in the transaction information received from Payleven will be

detected instantly. The payment ID's would match, however the corresponding transaction amounts would differ. This results in a failed refund request. Obviously, the €10 is also never credited to the merchant's bank account as Adyen is responsible for the processing of the transactions. In conclusion, in order to gain money, an attacker would need to focus on Adyen or, to be more precise, on the transaction information sent to the card reader and the data transmitted by the Payleven app to Adyen. Payleven clearly focuses on the administrative part of the payment platform.

7.2.3. FAKING TRANSACTIONS TOWARDS PAYLEVEN

More interesting is the fact that the web requests sent to the Payleven back-end during the transaction process (i.e., P4a and P5a) can be replayed. Replaying these two requests will ensure that a fake transaction is added to the transaction overview on the Payleven dashboard. Payleven is tricked to believe that a real transaction has taken place. The only thing that needs to be considered when performing this replay attack is to include the payment stub (i.e., payment ID) received from the Payleven back-end (response to web request P4a), in request P5a. Furthermore, the PSP identifier (`psp_payment_id`) included in request P5a should not be used before; changing one byte is often enough.

Just as with the refund attempt on an altered transaction, a refund request on a fake transaction also failed for obvious reasons. However, in comparison with the previous refund attempt, this attempt immediately prompted a fail message ('Internal server error'), whereas a refund attempt on an altered transaction did not result in an error message until after a few minutes. The error message was also different. In either case, an altered or fake transaction on the side of Payleven will not lead to monetary gain.

7.2.4. CASH BACKS WITH FAKE PAYMENT RECEIPTS

More serious is the issue regarding the payment receipts of fake transactions. These fake receipts are identical to legitimate receipts and can be abused by malicious attackers as they form a proof of payment. An evil customer could fake a transaction via an MITM-attack and use the corresponding fake receipts to claim a refund at the merchant. The refund would then have to be processed outside the Payleven payment system as refunds via the platform would immediately result in a fail message. Fortunately, it is possible to initiate refunds partially outside the payment platform. That is to say, payment transactions handled via the Payleven payment platform *in cash* can be refunded without the involvement of Adyen. The evil customer would then only need to fake a payment transaction in cash and use the corresponding fake receipt to claim a refund in cash (so called cash backs).

This attack would definitely be successful, but depends highly on the assumption that the attacker is already in possession of the fake receipt. This is however

not self-evident. The evil customer could not be in possession of the receipt, since if the transaction never took place, the merchant would never had an opportunity to hand over the receipt to the customer.

Luckily, Payleven always includes the web URL of the payment receipt directly in the response to the transaction request (i.e., request P4a, see Section 4.5). The web request for a payment transaction in cash is slightly different from request P4a, which is aimed at card payments. The main difference is that the PSP is set to 'Cash'. However, the response to this request also contained the payment stub, the transaction stub and the URL of the receipt.

In summary, an evil customer is capable to carry out this cash back attack with fake receipts by replaying just one web request (i.e., a variant of P4a). The response to this request would then lead the attacker directly to the fake payment receipt.

It should also be noted that Payleven does warn merchants not to give cash backs on payments conducted via the Payleven payment platform. This would suggest that Payleven is aware of the possible threats regarding fake receipts and fake transactions¹. However, this does not retain uninformed merchants from doing so.

As a side note, the attacks discussed in this and the previous subsection do not necessarily require a MITM-setup in order to be executable. These attacks can be carried out by replaying or re-using specific web request to Payleven. The only requisite is a valid Payleven session token, which is included in the authorization header of each web request sent to Payleven. Otherwise, it is not possible to authenticate to the Payleven back-end. The session token will be elaborated in the next section in more detail as it exhibits some undesirable characteristics. Also, some ways to retrieve Payleven login credentials will be touched upon in the next section since it gives access to a valid session token.

7.2.5. BYPASSING LOCATION RESTRICTIONS

As been pointed out in Section 2.2.3, the Payleven payment platform can, due to security reasons, only be used in the country the merchant has registered in. To enforce this, geo location details are included in almost every web request sent towards Payleven. However, it should now be clear that these location details can be altered very easily. For example, transaction have been conducted in Berlin (Germany) according to the receipt shown partially in Figure 7.2, while actually the payment platform was located in the Netherlands.

¹<http://help.payleven.co.uk/what-are-rules-using-payleven>

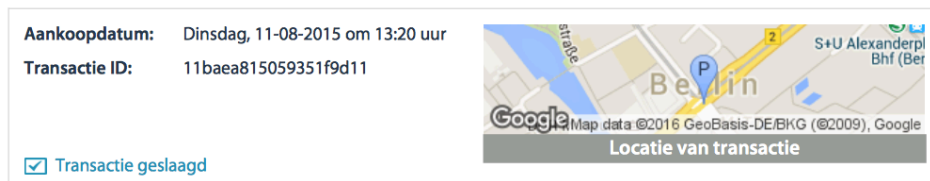


Figure 7.2: The receipt clearly indicates that the transaction was conducted in Berlin, Germany.

This finding gives the merchants the opportunity to fake their location and enables them to operate outside their permitted region. The only necessity is a proxy setup as described in this research.

7.3. SCENARIO 3: TRIGGERING ILLEGITIMATE REFUNDS

The Payleven payment platform makes it possible to carry out refunds on payment transactions. In order to use the possibility to initiate refunds, a 4-digit PIN code needs to be set up by the merchant. This PIN is then used to authorize refunds. The refund process is described in more detail in Section 2.3.5.

Section 4.6 describes the web request that is sent by the Payleven app to the Payleven back-end to initiate the refund process. As can be seen in Listing 4.29, the body of the web request holds a short JSON-message that lists the PIN and the transaction ID of the transaction in question. The header of the web request also includes the Payleven session token in the authorization header line. The session token is supplied to the Payleven app in the response to the app login request (see Section 4.2).

To sum up, the following three elements are required in order to carry out a refund successfully:

1. the Payleven session token (48 bytes);
2. a 4-digit refund authorization PIN code (set up by the merchant);
3. and the transaction ID.

7.3.1. BRUTE-FORCING THE REFUND AUTHORIZATION PIN

In essence, an attacker (in this case a malicious customer) could be capable of initiating illegitimate refunds: the attacker only needs to be in possession of the three above-mentioned elements. The transaction ID is easily retrieved since it is listed on the customer receipt. What remains is the PIN code and the session token. The PIN consists of a 4-digit numeric code, which means that there are 10,000 (10^4) different PIN combinations. This may be sufficient to ward off manual brute force attacks (i.e., entering the PIN on the smartphone manually); however, 10,000 PIN combinations are not sufficient in an automated environment.

Burp offers such an automated environment.² It features an automated tool that targets a specific payload value and performs brute-force attack against that value. In the case of a 4-digit PIN code, Burp will generate requests for each PIN combination and send these requests to the Payleven web servers. A total of 10,000 requests is sent to the back-end. This is equivalent to 10,000 PIN entry retries.

However, web applications often do implement measurements to counter these types of brute-force attacks. Counter measurements would be for example: (1) disabling the account after few retries (but sensitive to DOS) or (2) introducing random time delays between unsuccessful entries. Further countermeasures exist [27] [28] [29]. Strikingly, the Payleven back-end implemented none of these counter measurements. Payleven did not avert the 10,000 requests generated by Burp. This means that an attacker can easily retrieve the refund authorization PIN code through brute forcing.

7.3.2. IMPROPER SESSION EXPIRATION MANAGEMENT

What is left is the session token. The attacker needs a valid session token to authenticate against the Payleven back-end. The session token is generated server side once a login request containing the Payleven authentication credentials is submitted. It is needless to say that, if in possession of these authentication credentials, an attacker can easily retrieve a valid session token. Obviously, the attacker could mainly focus on retrieving the Payleven credentials, however the attacker could also try to hijack the login session through session sniffing (via an MITM-attack) or cross-site scripting.

An attack on the session token itself seems improbable: a statistical analysis of 2000 tokens have shown that the quality of randomness within the token is estimated to be very high. The effective entropy is estimated to be 221 bits, at a significance level of 1%.³ Predicting the session token is thus not doable.

Nonetheless, an unexpected and interesting finding was observed during the collection of these 2000 tokens. Prior gathering 2000 tokens, 2000 login requests with the same login credentials were sent to the Payleven back-end. An interesting observation was the fact that it was not necessary to log out a existing login session prior it was possible to proceed with the next login request. However, the observation that immediately stood out was the fact that a new login request did not automatically invalidate the previous session. This means that multiple active sessions from the same Payleven account could exist at the same time. In this case, 2000 concurrent login sessions were active.

Furthermore, the session token lacks proper expiration management. Even after weeks of inactivity, the session tokens remain valid. This increases the payment platform's exposure to attacks [30] [31].

It is important to realize that this Payleven session token is not utilized by the Payleven web application. Therefore, cross-site scripting attacks to steal the token

²<https://portswigger.net/burp/intruder.html>

³The statistical analysis is performed with the Burp Sequencer module.

are not applicable here.

7.3.3. RETRIEVING THE LOGIN CREDENTIALS

An attacker may obtain the Payleven login credentials through several methods such as social engineering, phishing (e.g., via mail, malware or rogue website), MITM or dictionary attacks on the password. This section will highlight the latter two. As regards phishing, the next subsection will elaborate on the lack of Extended Validated (EV) SSL certificates and its implications for phishing attacks.

Dictionary attacks can be successful since no counter measurement are implemented to prevent such types of attacks, as indicated previously with the refund authorization PIN. The same applies for the login credentials. However, it should be noted that without the corresponding email address such attacks are not feasible. Fortunately, email addresses are public and often easily guessed or retrieved, especially corporate email addresses. For example, the email address may have the form 'info@corporate.com' or 'merchantname@corporate.com'. Merchants also often have a business card which lists their email address. The same address is then in all probability used for the Payleven payment platform since using multiple email addresses for different platforms will form a nuisance for most people.

Another way of obtaining the authentication credentials is through an MITM-attack. An attacker can perform MITM-attacks on two positions in the payment platform: between the Payleven app and the Payleven back-end, and between the merchant's web browser and the Payleven back-end (i.e., the Payleven dashboard). Performing a successful MITM-attack on the secure connection from the Payleven app to the Payleven back-end is not that feasible. That is because the Payleven app uses SSL certificate pinning to validate the server certificate. In order to perform a successful MITM attack, the attacker would then need to: (1) bypass SSL pinning on the merchant's smartphone with Android-SSL-Killer; (2) install Cydia Substrate since it is required by Android-SSL-Killer - which, in addition, requires a rooted device; (3) setup a malicious WiFi hotspot or proxy to which the smartphone connects to; and (4) install the proxy's malicious certificate on the smartphone. An MITM-attack on the merchant's browser does not require the above-mentioned steps and is relatively easier to be carried out.

Also, it is worth mentioning that Payleven does implement HTTP Strict Transport Security (HSTS). Therefore, downgrading the HTTPS connection will not work.

7.3.4. THE LACK OF EXTENDED VALIDATED SSL CERTIFICATES

The Payleven website lacks Extended Validated (EV) SSL certificate. Instead they use Domain Validated (DV) certificates. DV certificates provide the lowest level of validation available from commercial certificate authorities and are extremely easy to obtain. These types of certificates can be issued to anyone who is listed as the domain admin contact in the WHOIS record of a domain name. There is no

connection with a legal entity.

The lack of EV certificates make websites such as Payleven prone to phishing attacks. Attackers can set up a phishing site with the misspelled name of the legitimate domain (e.g., '*.pay1even.com') and request a cheap SSL certificate for that domain (i.e., not EV) [32]. Visitors may then be tricked into visiting the phishing website and entering their authentication credentials. Because the phishing site has a legitimate certificate, it will display the padlock in the browser's address bar. In effect, visitor are unlikely to have a proper reason to check whether the URL is correct. After all, the padlock indicates a secure connection.

EV certificates could have prevented this as they show a distinctive green lock in the address bar. Often the company's name is also presented in green in the address bar. By making visitors aware of this, Payleven can prevent phishing attack which involve misspelled domain names, because certificate authorities would never issue EV certificates for misspelled domain names since the validation is far more extensive.

8

CONCLUSION

Obviously, security is important for a functional smartphone-based card payment system. Therefore, this research set out to evaluate the security of such system: the Payleven payment platform. The payment platform consists of a smartphone payment app, a web-based personal dashboard, and a wireless Chip & PIN card reader (see Chapter 2). The card reader forms a separate mobile device that passes on sensitive transaction and card data to the corresponding smartphone app via Bluetooth. Particular the wireless and smartphone-based nature of the payment platform raised suspicion as to whether the information security of the system is adequately safeguarded.

This research has identified a number of security issues within the Payleven payment platform, which may make the platform susceptible to various attacks. Surprisingly, these issues did not concern the wireless nature of the payment system, but concern other aspects of the system. That is, the issues concerned the overall information-flow design of the platform and the implementation of server-side security measures. Moreover, the fact that the payment system uses a smartphone that may be compromised by an attacker only facilitates the exploitation of these issues.

8.1. MAIN FINDINGS

1. One of the more interesting findings is that Payleven is not responsible for the actual processing of payments as thought first. In fact, Payleven is only responsible for the administrative part of the payment platform; i.e., the handling of login sessions and the personal dashboard. From the point of view of the merchant (i.e., the customer of Payleven), it appears as Payleven is the only party involved. However, it soon became apparent that a second party is involved in the Payleven payment platform. Indeed, the payments

are processed by Adyen, a payment service provider (PSP) based in the Amsterdam, the Netherlands. The whole payment platform probably originates from Adyen, as supporting evidence for this have been found in the source code of both the Payleven app and the firmware on the Chip & PIN card reader (see Section 6.1 & 6.2.2). In conclusion, Adyen is a very important actor of the Payleven payment platform.

2. Another key point and also the biggest issue within the Payleven payment platform is the interconnection between the Payleven account and the Adyen merchant account. The merchant account is the main identifier of the merchant towards the card reader and towards Adyen. It determines on whose bank account the money is credited. The whole payment platform revolves around this identifier. The merchant account is communicated by Payleven to the Payleven app during each and every app login based on the logged-in Payleven account (see Section 4.3). It is this loose link between the Adyen merchant account and the Payleven account that forms a weak point. The Payleven app is assumed to trust all information received from the Payleven back-end regarding the merchant account, however the integrity and authenticity of this data throughout the payment platform cannot be assured, which poses a significant monetary threat for the merchant. The Payleven app does not have the possibility to check whether the received merchant account is indeed the correct one¹ and is not tampered with. The same applies to the card reader. The reader will accept any merchant account, and thus any destination bank account, that is fed to it and process transactions based on that account. This might make it possible to divert payments from the merchant to an adversary. A theoretical attack is discussed in Section 7.1.

The remaining issues identified in the Payleven payment platform are due to improper security implementations:

3. It is possible to brute-force the 4-digit refund authorization PIN code. No brute-force counter measurements are implemented by Payleven. By brute-forcing this PIN, it would be possible for an attacker to trigger illegitimate refunds (see Section 7.3). This further only requires a valid Payleven session token which can be, for example, retrieved via an MITM-attack or indirectly via phishing attacks. Furthermore, Payleven username and password are also susceptible to brute-force attacks.
4. As regards the Payleven session token, the Payleven payment platform lacks proper session expiration management (see Section 7.3.2). That is, multiple concurrent Payleven sessions can exist. Moreover, a new session does not invalidate the previous one and sessions only expire after a considerable large time frame (i.e., few weeks). This increases the exposure to attacks.

¹i.e., associated with the logged-in Payleven account

5. It is possible to fake transactions towards the Payleven back-end. Refunds on fake transaction will not work; however, it is possible to fake cash transactions together with the corresponding invoice. With that fake invoice, an attacker could claim a cash refund for a transaction that never happened (see Section 7.2.4).
6. Additionally, it is possible to fake location details towards the Payleven back-end. As to the location restrictions, it is thus possible to bypass these by changing the location coordinates to the country the Payleven account has been registered in (see Section 7.2.5).
7. The lack of Extended Validated (EV) SSL certificates on the side of Payleven is a cause of concern, especially since Payleven is registered as an official payment institution. The lack of EV certificates makes the Payleven website more susceptible to phishing attacks (see Section 7.3.4). Payleven should take the extra step of using EV certificates to protect the merchant's Payleven login credentials as these can be abused to fake payments or trigger illegitimate refunds as pointed out above.
8. As regards the card reader, it is possible to evoke a hidden admin menu on the card reader by holding the '9' key. The card reader then prompts for an admin PIN code to unlock the menu (see Section 6.2.4). Strikingly, these PIN codes were easily retrieved by extracting the firmware from the card reader, which were stored in plain text. Moreover, it is relatively easy to extract the reader's firmware and file system as no real protection is embedded to prevent this from happening. However, a large part of the extracted file system was obfuscated - probably holding the secret keys.

8.2. CONCLUDING REMARKS & RECOMMENDATIONS

In the final analysis, it can be concluded that the issues (2), (5) and (6) primarily exist due to the fact that two parties with different responsibilities and priorities are involved in the system. Adyen provides the core of the payment system, whereas Payleven provides everything around the core; that is, the payment dashboard, the login functionality, and the availability of the app. It may be the case that these two systems were developed separately, without the inference of the other, hence introducing weaknesses in the overall system which could easily be prevented if the other was involved more in the process.

In comparison to the implementation related issues, the issue concerning the Adyen merchant account is less easily resolved. It is inherent to the overall design of the system. Therefore, some fundamental changes need to be applied. In the end, the fundamental issue lies in the fact that Payleven has access to the crucial Adyen merchant account. Payleven never should have provided the app with the merchant account in the first place. This responsibility should be delegated to Adyen. In fact, the Payleven app itself never should have had any access to the merchant account. Its functionality does not depend on the account. Hence, a

possible solution would be to forward the merchant account directly to the card reader via the application-level encryption channel set up by Adyen. After all, the merchant account should be treated just as other sensitive banking data.

8.3. FUTURE WORK

With respect to security, Adyen did a good job by setting up an application-level encryption channel between the card reader and the Adyen back-end. How this channel is set up, what keys are used and how the key management is carried out, is unknown. Further research is required to determine the security of the key management infrastructure underpinning the payment system. It would be very interesting to see whether these keys could be extracted from the card reader.

As regards the Payleven app, investigations into the possibilities of a rogue app are necessary in order to determine the feasibility of the theoretical attack on the merchant account as outlined in Section 7.1. Also, further experimental investigations are needed to estimate the effectiveness of this attack.

Another possible area of future research would be to investigate the possibilities and effectiveness of Android malware targeting the Payleven app or other comparable payment apps. This would be a very fruitful area for further work since it could disrupt the market for smartphone-based payment applications.

Future research could also investigate the possibilities of a rogue card reader, especially with regards to online banking. The card reader could, for example, act like it is handling a Payleven payment, but ‘under the hood’ use the card details and the customer’s PIN to start an online banking session.

More broadly, future research could also compare the Payleven payment platform to similar smartphone-based payment systems from other vendors and manufacturers. It is for example known that also other vendors make use of the Miura Shuttle (i.e., the Chip & PIN card reader used by Payleven). Future research might explore the similarities and differences in the firmware of those devices.

BIBLIOGRAPHY

- [1] W. Frisby, B. Moench, B. Recht, and T. Ristenpart, *Security analysis of smartphone point-of-sale systems*. in *WOOT* (2012) pp. 22–33.
- [2] [Youtube] *Black Hat 2014 - Mobile: Mission mPOSsible*, <https://www.youtube.com/watch?v=PqdFtYCRa2g>, accessed: 2016-01-20.
- [3] 'Smart credit card' terminals can be hacked too, <http://money.cnn.com/2014/08/08/technology/security/hack-credit-card-terminal/>, accessed: 2016-01-20.
- [4] *EMVCo: EMV 4.3 specifications*, <https://www.emvco.com/specifications.aspx?id=2>, accessed: 2016-01-20.
- [5] M. Bond, O. Choudary, S. Murdoch, S. Skorobogatov, and R. Anderson, *Chip and skim: Cloning emv cards with the pre-play attack*, in *Security and Privacy (SP), 2014 IEEE Symposium on* (2014) pp. 49–64.
- [6] S. Murdoch, S. Drimer, R. Anderson, and M. Bond, *Chip and pin is broken*, in *Security and Privacy (SP), 2010 IEEE Symposium on* (2010) pp. 433–446.
- [7] B. Adida, M. Bond, J. Clulow, A. Lin, S. Murdoch, R. Anderson, and R. Rivest, *Phish and chips*, in *Security Protocols* (Springer, 2006) pp. 40–48.
- [8] J.-S. Coron, D. Naccache, and M. Tibouchi, *Fault attacks against emv signatures*, in *Topics in Cryptology-CT-RSA 2010* (Springer, 2010) pp. 208–220.
- [9] *PIN Transaction Security (PTS) Modular Security Requirements*, PCI Security Standard Council (2015).
- [10] *PCI DSS Version 3.1 Requirements and Security Assessment Procedures*, PCI Security Standard Council (2015).
- [11] *Miura Shuttle datasheet*, <http://www.miurasystems.com/hubfs/MiuraUi/downloads/Shuttle-DataSheet.pdf?t=1440608034550>, accessed: 2015-03-03.
- [12] *Ubertooth*, <http://ubertooth.sourceforge.net/hardware/zero/>, accessed: 2016-01-20.
- [13] *Github - ubertooth*, <https://github.com/greatscottgadgets/ubertooth/wiki/Ubertooth-One>, accessed: 2016-01-20.
- [14] *Wireshark homepage*, <https://www.wireshark.org>, accessed: 2016-01-20.

- [15] Sourceforge - ubertooth mailing 1, <http://sourceforge.net/p/ubertooth/mailman/message/31233507/> (), accessed: 2016-01-20.
- [16] Sourceforge - ubertooth mailing 2, <http://sourceforge.net/p/ubertooth/mailman/message/32503427/> (), accessed: 2016-01-20.
- [17] Google playstore - hci logger, https://play.google.com/store/apps/details?id=com.android_rsap.logger&hl=nl, accessed: 2016-01-20.
- [18] Portswigger burp proxy, <https://portswigger.net/burp/>, accessed: 2016-01-20.
- [19] Wikipedia - dd (unix tool), [https://en.wikipedia.org/wiki/Dd_\(Unix\)](https://en.wikipedia.org/wiki/Dd_(Unix)), accessed: 2016-01-20.
- [20] Binwalk - firmware analysis tool, <http://binwalk.org>, accessed: 2016-01-20.
- [21] Jtag 101 - ieee 1149.x and software debug, <http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/jtag-101-ieee-1149x-paper.pdf>, accessed: 2016-01-20.
- [22] J. Huncker and C. W. Probst, *Insiders and insider threats—an overview of definitions and mitigation techniques*, Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications **2**, 4 (2011).
- [23] F. Swiderski and W. Snyder, *Threat Modeling*, Microsoft Press (2004).
- [24] Mail Order/Telephone Order Definition, <https://www.cardpaymentoptions.com/glossary/mail-ordertelephone-order-moto-definition/>, accessed: 2016-01-20.
- [25] Emvlab.org - capki, <http://www.emvlab.org/emvtags/show/t9F22/>, accessed: 2016-01-20.
- [26] Emv glossary, <https://www.level2kernel.com/emv-glossary.html>, accessed: 2016-01-20.
- [27] Blocking brute force attacks, https://www.cs.virginia.edu/~csadmin/gen_support/brute_force.php, accessed: 2016-01-20.
- [28] I. Ristic, *ModSecurity Handbook* (Feisty Duck, United Kingdom, 2010).
- [29] M. Burnett, *Hacking the code: auditor's guide to writing secure code for the web* (Elsevier Science, Burlington, MA, 2004).
- [30] Cwe-613: Insufficient session expiration, <https://cwe.mitre.org/data/definitions/613.html>, accessed: 2016-01-20.
- [31] R. C. Barnett, *Preventing Web Attacks with Apache* (Addison-Wesley Professional, 2005).

- [32] *The dangers of domain validated ssl certificates*, <http://www.symantec.com/connect/blogs/dangers-domain-validated-ssl-certificates>, accessed: 2016-01-20.