Security testing framework for a novel mobile wallet ecosystem

João Santos,
TIMWE Lab, Tortosendo,
Portugal
joao.santos@timwe.com

Marco Antunes,
TIMWE Lab, Tortosendo,
Portugal
marco.antunes@timwe.com

João Mangana,
TIMWE Lab, Tortosendo,
Portugal
joao.mangana@timwe.com

David Monteiro,
TIMWE Lab, Tortosendo,
Portugal
david.monteiro@timwe.com

Patrick Santos,
TIMWE Lab, Tortosendo,
Portugal
patrick.santos@timwe.com

João Casal,
TIMWE Lab, Tortosendo,
Portugal
joao.casal@timwe.com

Abstract – **Tasks related with payments, personal identification, and the usage of tickets and marketing items can be achieved through mobile wallets. Despite the advantages that these present, such as their ease of usage and great accessibility, some security concerns are still pointed out by consumers. On this paper are presented the main security issues existent on mobile wallet ecosystems nowadays, and the possible tests and security measures that can be adopted on those ecosystems. The tests that we describe on the paper are performed on a novel mobile wallet ecosystem named weWallet, and are segmented on three axes: mobile applications, cloud infrastructure, and physical communication technologies.**

Keywords-component: Mobile Wallet; Security; Testing Framework

I. INTRODUCTION

Nowadays, tasks related with payments, personal identification, marketing and digital tickets usage can be made through mobile applications due to the digitization of physical wallets [1]. Companies like Google [2], Apple [3], and PayPal [4] have already deployed their own mobile wallet solutions into the market. However, several studies indicate that the mobile wallet usage is lower than expected mainly due to security concerns among consumers [5]. For example, 67% of the consumers that participated on a 2015 survey about mobile financial services in the United States shown concerns regarding the security on mobile payments [6]. The most concerning aspects to those consumers involved the phone being hacked or data being intercepted during the payment process (25%), the phone being stolen or lost (13%), and the companies not providing enough security measures to protect the mobile transactions (7%). On another survey on mobile payments security, performed worldwide in 2015, only 23% of the surveyed consumers did not shown any concern about mobile payment methodologies usually adopted [7].

As it can be observed, consumers still lack confidence on mobile transactions. In fact, this mistrust is justifiable, since the payment process chain is complex and involves several actors, with each one introducing potential new security breaches on the ecosystem. Thus, on a mobile transaction chain, there can be identified, at least, the following actors [8]: *(i)* consumers and their mobile devices; *(ii)* merchants and point-of-sales (POS); *(iii)* app providers and their backend systems; *(iv)* payment service providers and their systems; and *(v)* financial institutions/banks and their systems. If ticketing, marketing and personal identification tasks are also included on a mobile wallet, other actors and related threats can be identified, increasing even more the complexity of the ecosystem. In order to protect the high-level confidentiality of the data flowing inside this ecosystem, some standards can be followed, such as the Payment Card Industry Data Security Standard (PCI-DSS) [9]. This is a proprietary information security standard created to increase controls around cardholder data and to reduce frauds on monetary transactions, which also defines objectives and guidance for assuring the security of a payment transaction and the integrity of the supporting environment. Thus, the usage of security mechanisms based on crypto-graphed data on mobile devices and on databases inside cloud infrastructures, tokenization systems, access authorization systems, firewalls, and the performing of regular security testing on the system can be essential to improve the security on mobile wallets and also to improve the acceptance of mobile wallets among consumers.

This paper presents issues and challenges related with security on mobile wallet ecosystems, and a study about the measures needed to evaluate those ecosystems. Besides that, it also describes the security measures implemented on a novel mobile wallet ecosystem. Based on these aspects, our work will depict in detail: *(i)* the threats on personal devices, centralized infrastructures, and communication technologies on mobile wallet ecosystems; *(ii)* the functional and automated tests that can be performed to mitigate the threats previously identified on mobile wallet ecosystems; *(iii)* the implementation of those tests on a novel mobile wallet ecosystem (weWallet); *(iv)* the definition of measures to improve the security on the novel mobile wallet ecosystem after conducting the tests previously mentioned.

The remainder of the paper is organized as follows. Section II presents related work about security issues on mobile wallet ecosystems. Section III briefly describes the weWallet ecosystem and the security measures there implemented. Section IV presents the security tests performed on the weWallet ecosystem, while Section V explains the results obtained with those tests. Finally, Section VI presents a discussion about the obtained results and open issues that have to be passed to improve the security on the weWallet ecosystem.

II. Related Work

This section surveys some relevant work about the security threats on mobile wallet ecosystems.

Mobile wallet ecosystems directly depend of different **physical communication technologies** in order to assure that consumers' mobile devices interact within the ecosystem. For example, the Near Field Communication (NFC) technology is widely used on mobile wallet payments. Despite that fact, there are **some threats** that have already been identified when making mobile payments through **NFC**. An example of that is given in [10], where it is mentioned a PIN authentication bypassing of the Google Wallet through a brute-force attack. Besides this attack, there are also described other kind of threats, such as the Man-in-the-Middle (MiTM) attack, which allows unauthorized users to intercept the transactions and read payment data of the consumer. Some measures that are indicated to improve the security on mobile wallets are related with the encryption of all the consumer data, including the payment details. On [11] is performed an evaluation of the advantages and the disadvantages of software card emulation in NFC-enabled smartphones, which is intended to improve the interoperability of NFC with legacy contactless smartcard systems. Despite the benefit that this approach presents over peer-to-peer mode, - devices in card emulation mode and devices in reader/writer mode communicate on an easier way - it also presents security breaches, since content on emulated cards is less secure when compared with content stored on the Secure Element (SE) of a smartphone. On [12] is described a relay attack on NFC-enabled smartphones, where a software-based relay attack is conducted on payment transactions made through the Google Wallet. In the first place, a relay software needed to be installed on the consumers' smartphone in order to directly access the private information stored in the SE. Then, all that information could be forwarded to a card emulator (or proxy-token), which is a device (smartphone, for example) able to emulate a contactless smartcard in software.

QR codes also allow to perform payments through mobile wallets. Usually, there are two ways to process a payment by using QR codes: *(i)* the customer scan a QR code on the merchants' PoS in order to finish the transaction and receive the receipt, and *(ii)* the merchant scans a QR code on the consumer smartphone in order to identify that same consumer within the system, and then process the transaction to him. However, **QR codes are prone to some security breaches**. As it is mentioned in [13], payments made through QR codes are dependent of the in-store Wi-Fi network and its underlying security. That fact improves the probability to attacks such a Denial of Service (DoS) or a MiTM to happen, which could make consumers' identities to be stolen, or payments to be redirected to another accounts by overlaying fake QR codes over existing ones. Besides these facts, there are some scientific researches that also point security issues regarding QR codes, such as the work presented on [14], which indicates situations where QR codes can be used as attack vectors when the attacker replaces some parts of the QR code, or even the whole QR code in order to direct the consumer to untrusted web destinations.

Bluetooth Low Energy (BLE) can also be used in mobile wallet application, mainly to identification or location-based marketing purposes. Most of the times, these functionalities can be achieved by using BLE beacons. However, the **BLE suffers from some security inefficiencies**. As it is described in [15], the devices' pairing mechanism used in BLE does not provide any protection against eavesdropping and MiTM attacks, resulting in a potential capture of secret keys and other private information stored in the consumer' mobile device. In [16] are mentioned the beacons' most dangerous vulnerabilities, including their implications for real world deployments. For example, spoofing attacks allow malicious individuals to capture a beacon UUID, manipulate the beacon network infrastructure by replicating UUIDs, and leverage the provided services as they intend and without the knowledge of the network administrators. DoS attacks are also possible by draining the battery of the beacon devices, or by crashing the Bluetooth stack of those same devices, which cause problems in service advertisement inside the network.

Mobile payment ecosystems have high demands in terms of security. Sensitive data of the consumers' credit card/bank account must be stored in secure places where the access to the data is completely restricted to authorized personnel, and the likelihood of successful attacks is very reduced. So, sensitive data on mobile payment ecosystems can be stored on the internal memory of the mobile device, on the cloud, or on the SE of the mobile device [17]. However, storing sensitive data on the internal memory of the mobile device presents some issues, since only the sandboxing security mechanism is used as default to secure that data. The risk of the consumers' credit card/bank account data to be attacked when it is stored on the cloud is lower, since it is protected by stricter security mechanisms. However, some attention is needed to avoid possible spoofing or interception attacks. Storing sensitive data on the SE is seen as the most secure methodology among the three that were previously identified, since data is stored on a tamper-proof element on the mobile devices which is also isolated from data of other applications. However, this solution lacks scalability, since it is completely hardware-based, and adds dependence on mobile operators to use the SE incorporated on the SIM card. So, providers of mobile wallet solutions started to use other mechanisms to safeguard payment related sensitive data, such as the Host Card Emulation (HCE). The HCE allows NFC-based monetary transactions to be performed by mobile devices without the need to connect to the embedded SE, emulating the credit card on the device while using a virtual card number. Then, all the sensitive data related with the transaction is stored on a centralized infrastructure on the cloud, which can be controlled by the mobile wallet providers. For example, Google has adopted this mechanism for the Android Pay, as it is described in [8]. However, by trusting on a **cloud infrastructure** to store data, the mobile wallet ecosystem gets exposed to attacks that typically affect that infrastructure. As such, mobile wallet ecosystems can be exposed to [18][19][20]: *(i)* **DoS attacks**, which might affect services that require real time access by the mobile payment application to the payment services hosted in the cloud; *(ii)* **Authentication attacks**, especially when weak encryption mechanisms are used, and when two factor/step authentication methods are not used (2FA); *(iii)* **Cloud Malware Injection attacks**, which can cause a malicious individual to access private credit information about the consumer, and even eavesdrop a monetary transaction; *(iv)* **Spoofing and phishing attacks**, which can cause a monetary transaction process to be forwarded to an unreliable web location; *(v)* **Account hijacking**, which consists on the misappropriation of credentials related with credit card/bank account of a user, which then allows the processing of mobile transactions in the cloud by a malicious individual; *(vi)* **Insecure APIs and interfaces**, since these represent a direct access for interacting with the system, on which a potential vulnerability can disclosure a great portion of sensitive information. In order to better protect the sensitive data related with transactions stored on the cloud, and to minimize the impact of the for mentioned attacks, dynamic primary account numbers, limited security keys, tokenization systems, unique device profiles, firewalls, and encryption of the transmissions across public networks must be used, while real-time transaction assessment, risk management and regular security tests (such as penetration tests) should be performed [8][21]. Some of these measures are recommended by the PCI Security Standards Council (PCI-SSC) [22].

Mobile wallet applications facilitate the manner a consumer makes payments, validates tickets, and identifies himself within a system. However, these applications are exposed to some security threats that general mobile applications also are, such as [23][24][25]: *(i)* Existence of multiple **rootkit** and **mobile malwares** such as Trojan horses, Botnets and Worms that can collect sensitive data from a user; *(ii)* Use of a **non-secure Wi-Fi network**, which can instigate MiTM attacks and sniffing procedures; *(iii)* **Insecure storage mechanisms on the mobile device**, which allow data to be stored in plain text or without proper validation and encryption techniques; *(iv)* **Installation of malicious software** on mobile devices through USB ports; *(v)* **Applying reverse source code techniques** to access sensitive non-secured information declared in the program source code (key, username or passwords directly declared and accessed in the source code), and to access several mobile devices' functionalities that have the potential to use private and sensitive user information (location, calendar, camera, microphone, photos, etc).

As it can be identified, there are multiple threats originated by different sources that can affect mobile wallets and other mobile money related applications. To avoid or minimize those threats, a great effort is needed to develop and maintain a mobile wallet ecosystem. As such, we will later present a set of tests and tools that can enable the development and maintenance of a mobile wallet ecosystem in a secure way, which will also aim for the protection of all the sensitive data available in that same ecosystem.

III. CASE STUDY

To prove the feasibility of the studies presented on this paper, we will perform security tests on a holistic mobile wallet ecosystem named weWallet [26]. This ecosystem, which has its architecture depicted in Figure 1, aims for the digitization of payment, identification, marketing and ticketing experiences into a single mobile wallet application, abstract to the operating systems and the communication technologies being used.

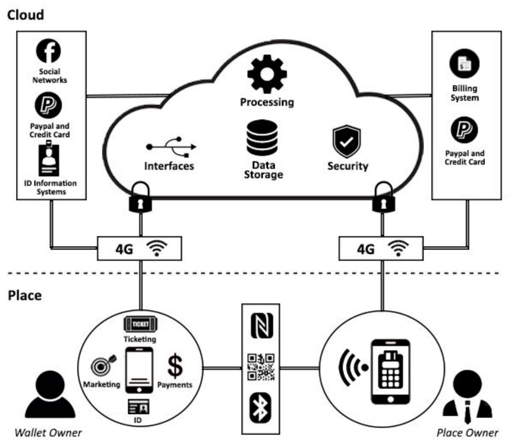


Figure 1. weWallet functional architecture.

In order to be scalable and to support different business sizes, the ecosystem considers two main personas: *(i) Wallet Owner*, which is the consumer that has the mobile wallet application installed on his smartphone; *(ii) Place Owner*, which is the person responsible for the business and manages the whole system performance. As a mobile wallet ecosystem, the weWallet has several security mechanisms implemented to assure that private information is correctly protected, and to grant secure communication between the wallet owner and the place owner, and also between the wallet/place owner and the cloud infrastructure. The following security mechanisms and tools are implemented on the weWallet ecosystem: *(i)* Role Based Access Control

(RBAC) systems; *(ii)* Use of SHA-512 hashing function to encrypt credentials of the wallet owners; *(iii)* Plain text passwords are not stored on databases at the cloud infrastructure, only the hash of those passwords is; *(iv)* Only one login per session to avoid multiple devices to be logged to the system at the same time; *(v)* Remote deactivation/disabling of the weWallet account to avoid unauthorized access to that account when a smartphone is lost or stolen; *(vi)* Sensitive information is stored at the cloud and split through multiple databases; *(vii)* Generation of dynamic QR codes with expiration timers to identify a user or a transaction, in order to diminish the probability of an attacker using that same QR code to access information about the user or about the transaction.

Despite using the for mentioned security mechanisms, the weWallet ecosystem was tested against some prominent threats that usually affect mobile wallet ecosystems. The performed tests, and the results will be presented in sections to come.

IV. Research Methodology

This section overviews the tests that can be performed to evaluate the security on the main components of the weWallet ecosystem. In order to assure that the weWallet ecosystem followed the best security practices existent nowadays, we studied the guidelines defined by the PCI-DSS and Open Web Application Security Project (OWASP) [27] compliances. As such: *(i)* strong security mechanisms should be applied on servers in order to protect all the sensitive data stored there, including the configuration of a firewall; *(ii)* access control systems must be implemented on backend systems; *(iii)* sensitive data entering, leaving, or being stored on mobile devices must be encrypted and prevented from being intercepted; *(iv)* mobile wallet applications must allow remote disablement; *(v)* mobile devices must be protected against unauthorized applications, malwares, unauthorized attachments, and other known vulnerabilities; *(vi)* regular security tests must be performed on the whole mobile wallet ecosystem (including penetration tests); *(vii)* third-party systems must be aware of the security measures that have to be implemented.

Table I indicates threats on the three main components of the weWallet ecosystem: mobile wallet applications (wallet owner and place owner apps), cloud infrastructure, and communication technologies. A total of fifteen security threats were identified as eligible to be tested. As we will indicate in the next section, some security testing tools were used to perform the necessary tests. Besides these tools, some functional tests were also conducted.

TABLE I. Mobile Wallet Ecosystem Testing Framework

Threats	***Components of the Ecosystem Involved***
1. Inexistence of mechanisms to validate sensitive fields on the mobile application.	-Mobile application
2. Reverse code engineering to access sensitive information hardcoded in the application.	-Mobile application
3. Directly access sensitive information stored on the mobile device.	-Mobile application
4.Intercept sensitive information leaving or arriving at the mobile device.	-Mobile application; -Cloud Infrastructure
5. Use the application with invalid credentials (SQL Injection).	-Mobile application; -Cloud Infrastructure
6. Incapability to remotely disable the mobile wallet account.	-Mobile application (backoffice)
7. Log in with the same account in multiple devices at the same time.	-Mobile application
8. Lack of two factor/step authentication/confirmation.	-Mobile application
9. Unauthorized access to sensitive data stored on the cloud.	-Cloud Infrastructure
10. System overloaded with external requests from clients.	-Cloud Infrastructure
11. Access sensitive information exposed by APIs.	-Cloud Infrastructure
12. Replace BLE beacons UUID.	-Communication Technologies
13.Intercept communication between BLE beacons and the consumers' mobile devices.	-Communication Technologies; -Mobile application
14. Intercept NFC communication in order to gather credit card information of the consumer.	-Communication Technologies; -Mobile application
15. Read a malicious QR code.	-Communication Technologies

V. Results

This section presents the results of the security tests performed on the weWallet ecosystem.

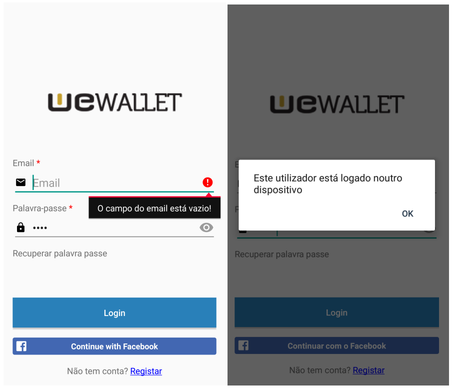
The first eight performed tests were mainly related with the mobile wallet application. Those tests were performed on root and non-rooted devices that had the Android OS installed. MobiSec 2.0 framework [28] was used to perform some of those security tests. This framework has several security testing tools, divided into categories such as forensics tools, penetration testing tools, and reverse engineering tools. The other tests were functional, and implied that some application functionalities were directly tested on smartphones with the weWallet applications installed. Table II depicts the type of tests, tools, and respective results of each security test conducted on the weWallet mobile wallet application (on rooted and non-rooted devices).

TABLE II. weWallet Mobile Applications Security Testing Results

Results of the security tests performed on the mobile wallet applications			
Threat Number	***Type of tests / Testing tools***	***Result***	
		Non-Rooted device	***Rooted device***
1	Functional tests	Passed	Passed
2	APK Studio [29],	Passed	Passed

	Mobisec (APK Tool [30], Dex2jar [31], JAD [32]) Adb commands		
3	Adb commands	Passed	Failed
4	Mobisec (Burp [33])	Failed	Failed
5	Functional tests	Passed	Passed
6	Functional tests	Passed	Passed
7	Functional tests	Passed	Passed
8	Functional tests	Passed	Passed

So, for the **first threat**, we accessed the mobile wallet application on different smartphones and tested every application flow that involved open fields that needed to be filled (Login, payments, among others). In every tested flow, the fields were validated (Figure 2, on the left), avoiding the introduction of unexpected values that could affect the system security (SQL injection attacks, for example).



O campo do email está vazio!

Este utilizador está logado noutro dispositivo

OK

Figure 2. weWallet email validation on Login (on the Left); Login with the same account on multiple devices failed (on the Right).

For the **second threat**, we used the reverse engineering tools available on the Mobisec framework. First, developer mode had to be enabled on the smartphone on which we wanted to perform this test. Then, with the APK tool (available on Mobisec), we remotely accessed the path folder of the weWallet application and extracted the .apk of the application to our environment. Then, we converted the .apk file into .dex files by unziping it. Those .dex files were then converted into .jar files by using the dex2jar tool available on Mobisec. Finally, we decompiled the .jar files by using the JAD tool, originating Java classes of the original source code. However, the Java source code that is obtained is not exactly equal to the original code on the Android project, since Java code that is compliled and decompiled in the process is in short form and interpretable by the Java Virtual Machine. So, no sensitive information hardcoded into the source code of the application was obtained. Regarding the **third threat**, Adb commands were used to access sensitive information stored on the mobile devices. Through the command line, we executed the "adb shell" command to acess the file system of the non-rooted smartphone connected to the computer. However, by accessing the weWallet application folder, we discovered that we did not have enough privileges to access some files there available (databases, etc). On a rooted device, the procedure is a little bit different: we executed the "adb shell" command on the command line, and then executed the command "su" to access the rooted device file system as a super user. Finally we executed the "run-as com.xxx.yyyyy" (where x and y correspond to the package name of the mobile wallet application) command to directly access the private files of the application. With this procedure, we were able to access files (such as shared preferences) and databases of the application. For example, we have discovered the access PIN to unblock the application screen on the mobile device, since it was stored in plain text on a shared preferences file. The **fourth threat** was related with the interception of sensitive data flowing between the mobile wallet application and the centralized (cloud) infrastructure. This was possible by using the Burp Suite tool, which allowed us to intercept HTTP requests between the client and the server (Registration, Login, Payments, etc). However, the sensitive data fields were encrypted, and an attacker could not easily discover the values of each data field (ex: password field). Despite these facts, the communication was not protected via SSL/TLS, which is highly insecure for mobile payment systems. To test the SQL injection attack (**fifth threat**), we tried to introduce short-circuit evaluation expressions *(' or '1'='1'*--, for example) to bypass the validation mechanisms that were supposed to be used at the backend. However, these expressions did not allowed us to perform the desired operation (Login, for example), which was due to the mechanisms used at the backend to validate the arriving information. Then, the system is protected against SQL injection attacks. The possibility to remotely disable the mobile wallet account (**sixth threat**) was successfully tested in the backoffice application. To do that, we had to introduce valid credentials into the backoffice and click on "Cancel the device" option. We've tried to bypass the user credentials, but we did not succeeded, which allowed us to understand that the remote disablement of the mobile wallet account was secure. The **seventh threat** we previously identified is concerned with the use of the same weWallet account on multiple devices at the same time. We tried to make login into another smartphone with an account that was already logged on another device, but we were unsuccessful. We concluded that the system was protected against the utilization of the same account on multiple devices at the same time (Figure 2, on the right). Regarding the two-factor/two-step authentication mechanisms (**threat number eight**), these were implemented by allowing the user to define a PIN after login into the application. Then, everytime the application comes from the background, or everytime a user logs in into the application, or makes a payment, the PIN code must be introduced to allow the user to continue using the application. This mechanism introduces an aditional level of security into the mobile wallet application.

On the next table we present the results of the security tests exclusively performed on the cloud infrastructure.

TABLE III. WEWALLET CLOUD INFRASTRUCTURE SECURITY TESTING RESULTS

Results of the security tests performed on the cloud infrastructure		
Threat Number	***Type of tests / Testing tools***	***Result***
9	Functional Test	Passed
10	SoapUI [34]	Passed
11	Automated Test/SoapUI	Failed

The **ninth threat** concerned the direct access to the databases of the ecosystem on the cloud infrastructure. However, only authenticated users can access those databases, and we have not been able to bypass the credentials needed to access them. For the **tenth security threat**, related with the overload of the cloud infrastrucuture, we used the SoapUI [34] tool, which is an open-source testing application for representational state transfer (REST) and service-oriented architectures (SOA). The SoapUI application includes functional, load balance, and security testing tools. So, by using the load balance tool of the SoapUI, we were able to test several APIs in order to evaluate the performance of the system under a heavier than expected load. We configured the test environment to support up to 25 simultaneous virtual users, and detect failures every time the response time was higher than 3500 milliseconds. Figure 3 depicts the results of these tests on two APIs: Get Cities API (on top), which uses the HTTP GET mehod, and the Login API (below) which uses the HTTP POST method. It can be seen that in five minutes, on both situations, no failures or timeouts have occured. The results that we obtained with other APIs that we have tested were similar. This means that the servers always gave timely responses (under 3500 ms) to the users' requests. However, this fact does not mean that the given response was always positive (200 OK). Additionally, more tests with more virtual users could be performed to evaluate the system performance on an even heavier load.

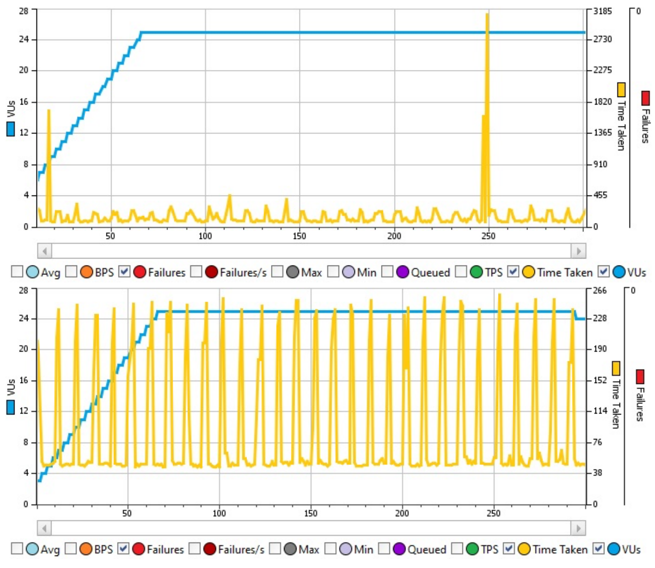


Figure 3. Peak load test of Get Cities API (on top) and Login API (below).

The **eleventh security threat** was also tested with the SoapUI. But this time, we used the security testing tool, which allowed us to perform a Cross Site Scripting scan, a Fuzzing Scan, SQL Injection attacks, XPath Injection attacks, Weak Authentication scans, among others. All the APIs of the weWallet project were tested in order to find potential security breaches. Since the communication between the client mobile device and the servers is not protected by SSL/TLS, all the senstive data regarding the consumers and the payment process should be encrypted, which is not true on this case. For instance, by testing APIs related with Login and Payments, it was possible to intercept some fields with values in plain text, such as the device description and the payment value. Intercepting the plain text device description in the Login API was not too harmful, since it only gives a textual description of the consumer mobile device ("SamsungS6_device", for example), and does not directly affect the system performance (neither the security). On the other hand, intercepting a payment value in plain text is very dangerous, since a malicious person could repalce that value for another without the consumer and the place owner noticing it. For example, this issue can make a wallet owner to pay more money for a product that it was supposed.

On Table IV are presented the results of the security tests related with the communication technologies used on the weWallet ecosystem.

TABLE IV. WEWALLET COMMUNICATION TECHNOLOGIES SECURITY TESTING RESULTS

Results of the security tests related with the communication technologies performed on a mobile wallet ecosystem		
Threat Number	***Type of tests / Testing tools***	***Result***
12	Functional Test	Passed
13	Functional Test/Wireshark	Passed
14	Functional Test	Passed
15	Functional Test	Passed

For the **twelfth threat**, we used three Estimote [35] beacons, and installed the Estimote application on an Android smartphone. Estimote beacons communicate by BLE, and each one is defined by a MAC Address and a universally unique identifier (UUID). Usually, the BLE beacons are used to advertise marketing content to other BLE-enabled devices that are nearby. Also, the content is dependent of the UUID or the MAC Address of each beacon. So, if the UUID can be changed through the Estimote mobile app, a problem that can be easily spotted regards the utilization of the same UUID by different beacons. However, a malicious person could only do that if he/she possessed the Estimote account credentials of the beacons' owner. Besides, this security issue is not directly related with the weWallet ecosystem. On the weWallet ecosystem, the UUID is only used to validate the identification of the beacon within the system, and does not directly influence the dissemination of the marketing content through nearby BLE-enable devices. On the worst scenario, a non-registered UUID could be assigned to a beacon, making the system completely unware of the beacon's existence.

However, the weWallet system have passed the security test, since on this system, the content disseminated by the beacons is dependent of their MAC Address. This address is not changeable, and diminishes the possibility to disseminate malicious content among the consumers. In order to replicate the **threat number thirteenth**, we used the Wireshark network protocol analyzer to inspect the communication between the mobile device of the consumer and the BLE beacons. Besides, we also have activated the "Enable Bluetooth HCI snoop log" option on the "Developer Options" of the consumer's Android mobile device to allow the creation of a log file with all the Bluetooth packets that arrived on the Bluetooth interface of the mobile device. After obtaining the file, we opened it on the Wireshark, where we analyzed each Bluetooth packet without retrieving any highly sensitive information (only the Bluetooth address of the beacon; the UUID is not presented in plain text). For the **threat number fourteen**, we used two smartphones: one with the weWallet application (place owner app) installed and another smartphone without any weWallet application installed (place owner or wallet owner application) that serves as the attacker on this test scenario. So, to perform the test, we opened the weWallet place owner application on the smartphone, and manually introduced a value that would correspond to the price of a product. Next we generated a payment through NFC, where the transaction information is passed to the smartphone NFC tag. Usually, and in order to perform a valid transaction, a client's smartphone with the weWallet owner application should touch the place owner's smartphone; but on this test scenario, we used a smartphone without the weWallet owner application installed. So, despite reading the NFC tag, the smartphone of the attacker could not correctly identify the information present on the NFC tag, since the tag identifier was unknown for the NFC reader available on the smartphone. Then, the transaction was not processed and the security test was passed. Lastly, for the **threat number fifteen,** we used a smartphone with the weWallet client application installed, and tried to read an invalid QR code that should be correspondent to a payment validation. After reading the QR code presented on the smartphone of the place owner, the client's smartphone presented a message indicating the QR code was invalid. So, the security test was passed, since it was demonstrated that it is impossible to process transactions through invalid QR codes. This happens because, at the time the QR code is read, it is compared with the QR code generated at the backend. If the QR codes are not equal, the transaction is not processed.

VI. Discussion and Open Issues

On this paper we presented a security testing framework for a novel mobile wallet ecosystem, which involved tests related with the wireless communication technologies used on the system, the mobile wallet applications, and the cloud infrastructure. In order to conceptualize the security testing framework, we had to identify the main threats that could affect each one of these components. Then, for each threat we indicated and performed a possible security test.

On the total, we identified 15 threats, which we tested on the weWallet ecosystem. The tests allowed us to discover some issues on this mobile wallet ecosystem. One major issue concerned the non-use of SSL/TLS cryptographic protocol on the communication between the mobile wallet devices and the backend. This issue resulted on the interception of sensitive information flowing between the consumer's mobile devices and the backend systems (Table I – threat number 4). Despite most of the sensitive fields being encrypted, it is very important to add an additional security layer to the communication link, since a malicious person could perform a brute force attack to decrypt the encrypted information that is intercepted. Another issue that we identified was related with the access to information stored on mobile devices that had the weWallet application installed. Despite the access to this information not being trivial (the consumer's smartphone is necessary), an experienced user could use Adb commands on a terminal to get private information about a weWallet user stored on a mobile device (Table I – threat number 3). This issue could be avoided by storing the sensitive information on a secure partition of the mobile device, such as the Secure Element, or by not storing the information on the mobile device at all. The other security test that has failed comprehended the access to sensitive information exposed by APIs (threat number 11), since some values related with payments could be intercepted in plain text. In order to avoid this issue two measures must be applied: *(i)* encrypt all the information related with the consumer, payments, and identification; and *(ii)* use SSL/TLS protocol to encrypt the communication link. Other possible issues reported with the physical communication technologies were dependent of other variables involving the devices to be used (for example, the Estimote beacons and the Estimote mobile app), and could not directly be controlled on the weWallet ecosystem. One of the tests made at the cloud infrastructure, related with the load balancing at the backend system, was passed; however, more tests would be needed to study the system behavior with a greater number of users trying to access the system.

So, in order to improve the security on the mobile wallet ecosystem that we tested, some functionalities could be added: *(i)* Use of Tokenization systems to protect sensitive information flowing in the system (Primary Account Numbers, for example); *(ii)* Use HTTPS (SSL/TLS encryption) to protect the communication between the consumers' mobile devices and the cloud infrastructure; *(iii)* Encrypt all the consumer related information; *(iv)* Only store information on the mobile device if it is going to be stored on the Secure Element.

The security framework for mobile wallet ecosystems that we propose on this paper is intended to help development teams and researchers understand the most important security measures to be included on those ecosystems. The security tests that we indicate are crucial for mobile wallet ecosystems, since users need to feel secure when using the mobile wallet applications in order to improve their acceptance on this kind of mobile money/marketing/identification systems.

Acknowledgements

This work is part of the weWallet project, co-funded by COMPETE/P2020/EU, in the context of the Portuguese Sistema de Incentivos à I&DT Empresarial (project 010463).

References

[1] S. Allums, *Designing Mobile Payment Experiences: Principles and Best Practices for Mobile Commerce*, 1st ed. O'Reilly Media, 2014.

[2] Google, "Android Pay," 2015. [Online]. Available: https://www.android.com/intl/en_us/pay/. [Accessed: 21-Feb-2017].

[3] Apple Inc., "Apple Pay," 2014. [Online]. Available: http://www.apple.com/apple-pay/. [Accessed: 21-Feb-2017].

[4] PayPal, "PayPal Wallet," 2017. [Online]. Available: https://www.paypal.com/us/webapps/mpp/pay-in-stores. [Accessed: 22-Feb-2017].

[5] S. Williams and D. Yu, "Digital Wallets Don't Deliver What Consumers Need," 2015. [Online]. Available: http://www.gallup.com/opinion/gallup/184094/digital-wallets-don-deliver-consumers-need.aspx. [Accessed: 19-Feb-2017].

[6] Board Of Governors of the Federal Reserve System, "Consumers and Mobile Financial Services," 2016.

[7] ISACA, "Mobile Payment Security : Perceptions and Behaviors," 2015.

[8] E. U. A. for N. and I. Security, "Security of Mobile Payments and Digital Wallets," 2016.

[9] P. DSS, "Securing the Future of Payments Together," 2017. [Online]. Available: https://www.pcisecuritystandards.org. [Accessed: 27-Mar-2017].

[10] T. Caldwell, "Locking down the e-wallet," *Comput. Fraud Secur.*, vol. 2012, no. 4, pp. 5–8, 2012.

[11] M. Roland, "Software Card Emulation in NFC-enabled Mobile Phones: Great Advantage or Security Nightmare?," in *Fourth International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use (IWSSI/SPMU 2012)*, 2012, p. 6.

[12] M. Roland, J. Langer, and J. Scharinger, "Applying relay attacks to Google Wallet," in *5th International Workshop on Near Field Communication*, 2013, pp. 2–7.

[13] P. Crosman, "Why mobile wallets still use QR codes—and why maybe they shouldn't," *Payments Source*, 2016. [Online]. Available: https://www.paymentssource.com/news/why-mobile-wallets-still-use-qr-codesand-why-maybe-they-shouldnt. [Accessed: 29-Mar-2017].

[14] K. Krombholz, P. Fruhwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. Weippl, "QR code security: A survey of attacks and challenges for usable security," *Lect. Notes Comput. Sci.*, vol. 8533 LNCS, pp. 79–90, 2014.

[15] S. C. Alliance, "Bluetooth Low Energy (BLE) 101: A Technology Primer with Example Use Cases," 2014.

[16] H. Jun Tay, J. Tan, and P. Narasimhan, "A Survey of Security Vulnerabilities in Bluetooth Low Energy Beacons," 2016.

[17] Underwriters Laboratories, "Mobile Payment Security discussion paper," 2014.

[18] M. Rahman and W. M. Cheung, "Analysis of Cloud Computing Vulnerabilities," *Int. J. Innov. Sci. Res.*, vol. 2, no. 2, pp. 308–312, 2014.

[19] C. S. Alliance, "Cloud Computing Top Threats in 2016: The Treacherous 12," 2016.

[20] S. V. Hatwar and R. K. Chavan, "Cloud Computing Security Issues and Countermeasures," *Int. J. Comput. Appl.*, vol. 119, no. 17, pp. 46–53, 2015.

[21] C. Hyperion, "HCE and Tokenisation for Payment Services discussion paper," 2014.

[22] PCI Security Standards Council, "Information Supplement : Third-Party Security Assurance," 2014.

[23] S. Pandy, "The Future of Mobile Security: Understanding the Risk Environment for Mobile Payments," 2013.

[24] G. Delac, M. Silic, and J. Krolo, "Emerging security threats for mobile platforms," in *Proceedings of the 34th International Convention MIPRO*, 2011, pp. 1468–1473.

[25] M. Sujithra and G. Padmavathi, "Mobile Device Security: A Survey on Mobile Device Threats, Vulnerabilities and their Defensive Mechanism," *Int. J. Comput. Appl.*, vol. 56, no. 14, pp. 24–29, 2012.

[26] J. Casal, D. Monteiro, L. Sousa, P. Santos, J. Santos, and J. Ramos, "Requirements Elicitation for a Holistic Mobile Wallet Ecosystem," in *Unpublished*, 2017.

[27] OWASP, "OWASP Top Ten 2017," 2017. [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_Top_Ten_2017_Project. [Accessed: 22-May-2017].

[28] J. G. Kevin Johnson, Tony DeLaGrange, "MobiSec Webpage." [Online]. Available: https://sourceforge.net/p/mobisec/wiki/Home/. [Accessed: 03-May-2017].

[29] V. Pandey, "APK Studio Webpage." [Online]. Available: http://vaibhavpandey.com/apkstudio/. [Accessed: 03-May-2017].

[30] C. Tumbleson, "APK Tool Webpage." [Online]. Available: https://ibotpeaches.github.io/Apktool/. [Accessed: 03-May-2017].

[31] B. Pan, "Dex2Jar Webpage." [Online]. Available: https://sourceforge.net/p/dex2jar/wiki/UserGuide/. [Accessed: 03-May-2017].

[32] P. Kouznetsov, "JAD Tool Webpage." [Online]. Available: http://www.javadecompilers.com/jad. [Accessed: 03-May-2017].

[33] PortSwigger, "Burp Suite Webpage." [Online]. Available: https://portswigger.net/burp/. [Accessed: 03-May-2017].

[34] Smartbear, "SoapUI Webpage." [Online]. Available: https://www.soapui.org. [Accessed: 03-May-2017].

[35] Estimote, "Estimote Webpage," 2013. [Online]. Available: https://estimote.com. [Accessed: 11-May-2017].