A Penetration Testing on Malaysia Popular e-Wallets and m-Banking Apps

Md Arif Hassan*, Zarina Shukur, Masnizah Mohd
Center for Cyber Security, Faculty of Information Technology
National University Malaysia (UKM), 43600 UKM, Bangi, Selangor, Malaysia

***Abstract*—e-Wallets and m-banking apps became more and more popular in the developed world, approaching a point of tipping. This can be due to the global use of big and small merchants of paying equipment and the ubiquity of e-wallet and m-banking apps adoption. Many consumers are using e-wallets and m-banking apps that can be an effective cybercrime option. e-Wallets and m-banking apps allow financial transactions via smartphones that give cybercriminals a lucrative opportunity. Mobile technology has become increasingly mainstream and continually strengthening, with the focus on mobile apps protection and forensic analysis developing. In this paper, the security aspect of five popular e-wallets in Malaysia were analyzed. This paper also provides a security analysis of another five leading m-banking apps. The security analysis is based on a security principle that is recommended by Open Web Application Security (OWASP) under Mobile Security Testing Guide (MSTG) and Mobile Security Threats (MST). The static analysis has been done by using three mobile application-testing tools. This study included a variation of vulnerability scanning, code review and, most significantly, penetration testing. Each app complied with the security requirement, but their security features and characteristics, such as encryption, security protocols, and app services, are different to each other. This study was carried out using a DELL computer with Intel Core i7 CPU, 3.40 GHz CPU, 6 GB RAM. Finally, the results revealed the secure e-wallet and m-banking apps among the selected apps.**

Keywords*—*Electronic payments; e-wallet; m-banking; android; static analysis; security analysis

I. Introduction

Nowadays, the development of technology advances has brought one of the pioneers of innovation in financial institutions. With the development of Fintech worldwide, there will still be enough challenges for those interested in adopting the technology. As part of their everyday transaction payment choice, several countries have already introduced the use of electronic payments. The payment methods used by consumers have great impacts on the future of the financial system and the business model of a country. Mobile payment services are increasingly popular in the banking world and are capable of replacing cash and becoming the most popular platform in the coming years. Fintech developments have changed payment systems in Malaysia. Malaysia has taken seriously the development of cashless societies in particular. The Central Bank of Malaysia intends to migrate towards a new cashless sector, in alignment with its financial plan 2020, with the intention of increasing efficiency in the financial sector [1]. For current stage, the most commonly used cashless payment methods are credit cards/debit cards, internet banking and cheques [2]. e-Wallet appears to be a new trend of mobile payment in recent years. In effort to enhance the use of e-wallet in Malaysia, the e-wallet users included in the Malaysia Budget 2020 are granted an RM30 reward [3]. e-Wallet is a modern age of technologies that easily recognizes consumer interest, making our transactions very convenient and efficient [4-5]. Security is among the most crucial factors influencing consumers' determination to use e-wallet apps [6-7]. Cybercrime is a challenge to mobile payment systems, and is obviously not the only concern, although many consider that it is the greatest challenge in the field of mobile privacy. Many e-wallet application developers are financially motivated, and as a result, it is common to overcome challenges quickly in order to save time and money. Cybercrime possibilities are becoming more challenging, hence humans want to share their understanding of certain ways hackers could interfere with e-wallet apps. Those who hope this point of view will help people realize how cyber criminals think, because if everyone know that, then users will continue to defend ourselves by securing certain points of attack.

In addition to the design of the e-wallet apps implementation, it is necessary to preserve the protection and privacy of user data in general [8]. A safety intelligence survey found [9] that 400 leading companies, 40% of them don't even scan their code for security flaws. Besides all these apps, mobile money applications effectively cover high-level private financial and sensitive information, where protection needs to be of the extreme significance as vulnerabilities or risks cannot be tolerated, as absolute protection is necessary. This paper studies e-wallet products associated with e-money issuers listed in National Bank of Malaysia websites, by focusing on its mobile payment feature. There are 42 e-money certificates has issued by Central Bank of Malaysia, including 5 banks and 37 non-banks [10] and [11]. Security in general has become an increasing concern currently, particularly with the evolution of mobile payments, and the almost daily vulnerabilities found in operating systems and applications are what makes it more challenging [12]. This study conduct a penetrating testing using three static analysis tools which is recommended by Open Web Application Security (OWASP). The recommendation of OWSAP related to three static tools are show in Table I.

TABLE I. Static Tools Security Principle

Tools	Security principle	Suggested by
MobSF	OWSAP	Mobile Testing Guide
MARA	OWSAP	Mobile Security Threats
AndroBugs	OWSAP	Mobile Testing Guide

*Corresponding Author.

In this study, our main contributions are as follows:

- We perform a static analysis among five e-wallet and m-banking apps, specifically on security issues targeted for Android applications.
- In order to detect repackaging threats, we evaluate successful solutions and recognize the vulnerabilities.
- We discover the most secure bank and non-banks mobile application among selected application using static analysis tools.

This paper is bifurcated into four sections. Introducing the payments and its related study, which is already discussed above. The second section presented the five bank and non-bank issuers in Malaysia and its association with e-wallet products. Section 3 discussed the proposed methodology of the analysis. Section 4 presented the experimental result of the methods and Section 5, provides the discussion of the findings and finally, the conclusion b presented in Section 6.

II. Literature Review

Electronic payment systems have risen in popularity in the last 20 years because of their significant contribution in modern electronic transactions. Electronic transactions are a financial exchange between buyers and sellers available on the internet [13]. The payment system electronically originally referred to as a payment process through an electronic network [14] which a user may use to make online payments for products and services [15]. Among electronic payment system nowadays, e-wallet and m banking is one of the most famous payment system. The definition and their functionality is described in the next subsection.

A. e-Wallet and m-Banking

e-Wallet is the new invention of finance technology that make our transaction and payment easy and fast. e-Wallet is a virtual storage system [16] that can capture your identity and digital credentials and offer to an electronic gadget or online service that pro-vides a person to commit electronic purchase [17-18]. The e-wallet includes two elements, namely software and information. The software holds all the information contained in a wallet that encrypts confidential personal data. In comparison, the data are all information, such as customer ID, card information and shopping addresses, provided by the customer. There are quite variety of e-wallet services established worldwide.

For the past several years, mobile banking has grown in popularity across many segments of society. M banking is a subcategory of electronic banking that combines both the basics of banking and the distinct features of mobile payment [19]. M banking refers to the delivery and use of banking and financial services using mobile communications de-vices [20]. The range of services available might include the ability to conduct bank and stock market transactions, manage accounts, and obtain personalized data. Mobile banking, often known as m banking, is a terminology for using a mobile device such as a phone to perform balance checks, account transactions, payments, and credit applications. It is the convenient, simple, secure, anytime and everywhere in the world. The functionality of the apps for each bank and non-bank issuers is listed in Table II and Table III.

- App based system: Application settings are also important since they allow user to personalize the program to his own needs. This will feature profile, payment, and security options, among other things.
- Fingerprint: Fingerprint identification is one of the most well-known and used bio-metric technologies. The fingerprint biometrics is useful and cost effective. Moreover, it can be quickly installed and used under any environmental conditions.
- Bills: This is an important e-wallet function because today's consumers like to pay all of their bills online, including utilities, mortgages, loans, rent, and tuition, to mention a few. e-Wallets are becoming a vital aspect of daily life as digital cash becomes more prevalent, and they should be able to give an easy bill payment option, whether it is a prepaid or postpaid payment service.
- Transfer of money: Transfer money between the payer and payee wallets in seconds rather than hours or days. This function has many advantages, including the ability to make payments at any time and from any location, the ability to make funds available immediately, and the ability to manage personal and business funds.
- Payment history: Any registered member will be able to view all the transaction de-tails in these features. After a successful login, the customer will be able to check or-der history.
- User account: To carry out a transaction from e-wallet to another e-wallet, the customer must be a user account. The user has been using a registration form. In order to register, the user must fill out all the fields required in the form. The user can access a variety of services using their user account.
- Pin Code: A personal identification number (PIN) is an unique code that must be in-put in order to complete certain banking transactions with a mode of payment. The purpose of a personal identification number (PIN) is to increase the level of security in electronic transactions.
- Top-up: By using these features, users could access the multiple bank list for top-up money. In these features, users need to choose how much they want to update. User chose their favorite bank account whenever. The payment gateway is submitted to the recipient of the application directly. The features help to users to add their balance into their e-wallet.
- QR code: In-store payments can be made using e-wallets using contactless methods such as near-field communication and Quick Response (QR) codes. QR Code is a form of 2D bar codes [21]. The QR code may be readily scanned with a smartphone camera [22]. The barcode is read by the smartphone, which then launches an associated apps or website.

- Open loop: Open loop mobile payment systems allow customers to pay from a centralized e-wallet at many different places. It is easy to comprehend closed payments as a gift card and open payments as a credit card [23].
- Add Money: Add money is used only for the logged-in users. It is connected to a payment gateway. The user can add or choose their bank account before transaction. User can add money with their registered bank account details or debit/credit cards using this function.
- Request money: Anyone may send his or her friend or family member a message to ask him or her to pay his or her money.
- Withdraw money: Users can transfer cash from their account to their connected bank account through the withdraw money functionality. Users may digitally withdraw money from the wallet into any bank account without the inconvenience of paper bills or currency, such as receiving money from sources, collecting money from distributors, in-store or online payments from consumers, or collecting money from sources.
- Voucher: Marketers and merchants are fully aware of the value of coupons and re-wards. e-Wallets are a great platform for providing these benefits to value customers in a timely manner. As a result, features that make it simple to create and manage coupons, discounts, tickets, and loyalty points are essential for an e-wallet solution and may help e-wallet software stand out in the market.

In the following Table, II and III means- the apps have the properties, whether 0 means the apps do not have the properties. 0 means the apps do not have the properties whether 1 means the apps have the properties.

Table II and Table III revealed e-wallet apps functionality by bank and non-bank issuer where, most of them have same features except Non-Bank5. It shows that there are common and additional properties of e-wallet apps. There are 10 common e-wallet apps functionality by bank and non-bank issuer. They are pin code, fingerprint, bills, transfer of money, top-up, and payment history, add money, and voucher, Request money, and Withdraw money. Each e-wallet playing its own role and position in electronic payment system, several e-wallets such as Non-Bank1, Non-Bank3, and Non-Bank4 focusing on withdraw money from ATM booth, while others focus on online transaction.

B. The Open Web Application Security Project

The OWASP is a non-profit organization that works to improve software security. OWASP relies on an ‘open community' concept, that enables anyone to participate in and assist to projects, events, online forums, and other services. Every three years, OWASP identifies the 10 most critical web application security risk types. This "top ten" list shows an agreement on the most serious security problems. The following were the top ten vulnerabilities as reported in OWASP 2017 are:

1) A1—Injection (SQL, OS, and LDAP).
2) A2—Broken Authentication.
3) A3—Cross-Site Scripting (XSS).
4) A4— Sensitive Data Exposure.
5) A5—Security Misconfigurations.
6) A6—Sensitive Data Exposure.
7) A7— Broken Access Control.
8) A8— Using Components with Known Vulnerabilities.
9) A9— Insufficient Logging and Monitoring.
10) A10— Insecure Deserialization.

The Open Web Application Security Project guide must be followed in some circumstances while developing web applications. The details of each vulnerability report can be found in [24].

TABLE II. THE FUNCTIONALITY OF BANK ISSUER M-BANKING APPS

Properties	**Bank1**	**Bank2**	**Bank3**	**Bank4**	**Bank5**
App based system	1	1	1	1	1
Fingerprint	1	1	1	1	1
Bills	1	1	1	1	1
Transfer of money	1	1	1	1	1
Payment history	1	1	1	1	1
PIN code	1	1	1	1	1
User account needed	1	1	1	1	1
top-up	1	1	1	1	1
QR code	1	1	1	1	1
Open loop	1	1	1	1	1
Voucher	1	1	1	1	1
Request money	1	1	1	1	1
Add money	1	1	1	1	1
Withdraw money	1	1	1	1	1

TABLE III. THE FUNCTIONALITY OF NON-BANK ISSUER E-WALLET APPS

Properties	**Non-Bank1**	**Non-Bank2**	**Non-Bank3**	**Non-Bank4**	**Non-Bank5**
App based system	1	1	1	1	1
Fingerprint	1	0	1	1	0
Bills	1	1	1	1	1
Transfer of money	1	1	1	1	0
Payment history	1	1	1	1	1
PIN code	1	1	1	1	0
User account needed	1	1	1	1	1
top-up	1	1	1	1	0
QR code	1	1	1	1	1
Open loop	1	1	1	1	1
Voucher	1	1	1	1	1
Request money	1	1	1	1	0
Add money	1	1	1	1	1
Withdraw money	0	1	1	1	1

C. Android Platform

Android is an open source smartphone operating system that was originally developed by Android Inc. and then acquired by Google with financial support. The first beta edition of Android was launched in November 2007 and the first stable version 1.0 followed in September 2008. Android is the world's most used mobile operating system, dominating the smartphone industry with an 82.8 percent share in 2015 [11]. That is good reason for this paper to perform the study on Android by itself. With the rising number of providers, each with its own Android OS version, the Android environment has been massively decentralized in recent years, ensuring that each has possible different vulnerabilities on top of some android platform problems, which is the total contrast on the IOS side where Apple maintains that a more compact closed ecosystem is accessible. IOS, though, is still suffering from smartphone protection concerns, and the android suffering is even higher. In comparison, being an open source based on Linux, Android makes it even easier to deal with, because it is much easy and popular for Android to remove the source code, scan the files, and include vulnerable code in applications.

D. Static Analysis

We chose static analysis as our vulnerability measurement technique despite reported vulnerabilities for several reasons. Unlike human code review or penetration testing, which results in reported vulnerabilities, static analysis is a purpose, repeatable, and scalable method for evaluating vulnerabilities. Static analysis tools employ a certain algorithms and rule sets every time and may scan a project in hours rather than days or weeks. Vulnerabilities can remain latent in code for years before a researcher discovers and re-ports them, thus the number of reported vulnerabilities is likely to be underestimated by an unspecified amount. Using static analysis, we could investigate the evolution of an application's vulnerabilities details. The number of vulnerabilities we uncovered with our static analysis technique far exceeded the number disclosed for the group of applications. Vulnerabilities of the same type in the same application version must be combined into a single item because of the Common Vulnerabilities and Exposures (CVE) criteria, which explain some differences in performance. Fig. 1 depicts the vulnerability static analysis process.

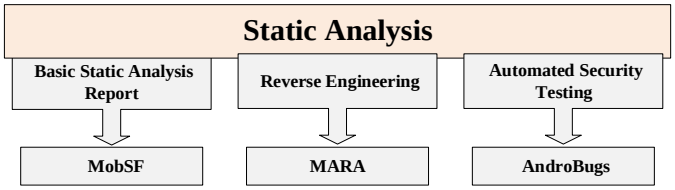


Fig. 1. Static Analysis Process.

1) MobSF basic static analysis tools: Mobile Security Framework that is an automated pen-testing framework capable of per-forming static and dynamic analysis. The framework also includes REST APIs that enable developers to use continuous integration and continuous delivery (CI/CD) pipelines to test their apps automatically with each build. MobSF v2.0 is open-source and written in Python 3.7. It is released under the GNU Public License v3.0. The study's installed version was a Docker container provided and maintained by the authors of MobSF. This simplified the installation process and allowed the service to be dismantled or rebuilt on demand. Users can interact with MobSF's graphical user interface by navigating to localhost: 8000 when the Docker container has successfully started up. The security analysis of MobSF is depends on the following properties:

- Signer Certificate: A signing certificate encrypts an app, ensuring that the code underlying it is protected and that no one is defrauded.
- Application Permission: In Mobile application, several permissions that are classified as dangerous or acceptable. Understanding which permissions can lead to further damage is critical from the perspective of a security analyst. For example, if an application has access to external media and stores essential information on the external media, the files stored on the external media are globally accessible and writable, which might be harmful. Android app permissions can give apps control of users phone.
- Network Security: Details on network security issues relating to the application can be found in the network security section. These flaws might lead to critical attacks like man in the middle attack.
- Android API: The Android operating system gives a framework API for apps to interface with the Android underlying system.
- Browsable Activities: Browsable Activities can control how the device should react when the user clicks on a link in the web browser.
- Security Analysis: Details about security issues relating to the application can be found in security. These problems can lead to critical attacks. The security analysis is including Network Security, Manifest Analysis, Code Analysis, Binary Analysis, NIAP Analysis and File Analysis.
- Manifest Analysis: Manifest Analysis may extract a variety of data from an Android manifest file, such as which activities are exported, if the app is debug gable, data schemas.
- Code analysis: The code analysis part of the MobSF tool is one of the more interesting aspects. MobSF has analyses, evaluated parts of the application’s behavior using industry security standard practices such as OWASP Mobile Security Testing Guide (MSTG), and mapped the vulnerabilities using OWASP Top 10. Furthermore, Common Weakness Enumeration and Common Vulnerability Scoring System scores (CVSS) are stated, which might be helpful in different analyst scenarios and make the development of reports a bit more straightforward for developers and analysts.

- Malware Analysis: Malware analysis is the domain malware check. MobSF extracts the hard-coding or application-using URLs/IP addresses, presents the malware status, and uses the ip2location to indicate its Geo location. APKiD is used to identify different packers, compilers, and hypocrites.
- Reconnaissance: Reconnaissance is used to identify the applications URLs, firebase DB, emails, trackers, and string and hard-coded secrets. This is all done using the decompiled source code.
- Components: Components are used to identify the details information regarding activities services. This summarizes the android APK skeleton. The components are including Activities, Services, Receivers, Providers, Libraries and Files.

In contrast to traditional desktop and web apps, mobile applications have unique security challenges. With mobile app security, the MobSF tool is widely employed. According to MobSFs, security score, CVSS and trackers' detection determine the outcome.

- Security Score: Security scoring is one of the important features to calculate the overall result. This security scoring is based on, if it introduced an issue to an app that al-ready has higher average CVSS than that issue, app would actually have higher score than before even though it now has more issues. The developer improves the tool's security score. Since the average CVSS score is used, an app with one major issue and several minor ones may score higher than one with only one major issue [25]. Currently, app score is calculated as:

avg_cvss = round(sum(cvss_scores) / len(cvss_scores), 1)

app_score = int((10 - avg_cvss) * 10)

- CVSS: The CVSS score can be utilized to determine the severity of vulnerabilities found in apps. The CVSS Calculator can be used to calculate the CVSS score. The formulas given in the CVSS specification are used in the calculation [26]. The CVSS Metric Values are shown in Table IV. The details CVSS scores provided by MobSF can be seen in supplementary file (Table I-III) [26-27].
- Tracker Detection: Each app may make use of third-party trackers. MobSF analyses the detected tracker in the system's APK using the open source Exodus-Privacy web tool. Tracker analysis can be found in two ways, such as crash reporters and analytics. Crash reporters are those that look into crashes that happen when the program is running normally. Alongside, analytics tracker collects information on how users interact with the app, such as how much time they spend in it, which features they utilize [28].

2) Reverse engineering: The method of extracting technology or design information from something man-made is known as reverse engineering [29-30]. The theory behind, it has for centuries been understood that destroying something would help you understand it, evaluate it and even twist it to achieve another task. In the computer security industry, reverse engineering is commonly used to analyze and exploit viruses and malware, vulnerability detection, binary code auditing, and development [30-31]. In this paper, reverse engineering was introduced after automated security tested as a second part of the static vulnerability analysis.

Reverse engineering is in particular used to see how engineers have constructed this specific system and how they perform essential protective activities and specifications [32]. These are some examples of what we can look for in Android security tests while using reverse engineering: database link, DB name, or DB password, certain hard-coded usernames or passwords that is used to accessing the database [33]. The application's APIs to see whether any of them are compromised, or the API key, and to check for a known vulnerable method after downloading the source code. This section will discuss the tools used in Reverse Engineering, and then the procedure followed to identify vulnerabilities in the selected applications, and will finally go over the results obtained from reverse engineering. During the reverse engineering, the method was initially focused exclusively on the Mobile Reverse Engineering & Analysis framework (MARA) which takes an APK file and delivers the source code in a language that is easily understood in Smali. Fig. 2 indicates the process that follows an application for reverse engineering.

TABLE IV. CVSS METRIC VALUES

No	Metric	Metric Value	Description
1	Attack Vector	Network Adjacent Network Local Physical	The attack vector defines the circumstance in which a vulnerability can be exposed.
2	Attack Complexity	Low High	This metric specifies the conditions that must exist outside the attacker's control in order to exploit the vulnerability. Depends on the situation, unique conditions that need a measurable amount of preparation or execution are necessary for the exploitation of the vulnerability, the metric's score might be low (L) or high (H).
3	Privilege Required	None Low High	The level of privilege necessary to exploit the vulnerability is defined by this metric. Its value is None (N) if the attacker does not need any permission; Low (L) if the attacker just needs basic privileges to change user-owned settings and files; or High (H) if the attacker needs major privileges to affect component-wide configurations and files.
4	User Interaction	None Required	This metric can have the values None (N) or Required (R) depending on whether the vulnerability is exploited with or without user participation.
5	Impact*Conf*, Impact*Integ*, Impact*Avail*	High Low None	This metric are refer to the Confidentiality Impact, integrity and availability impact.

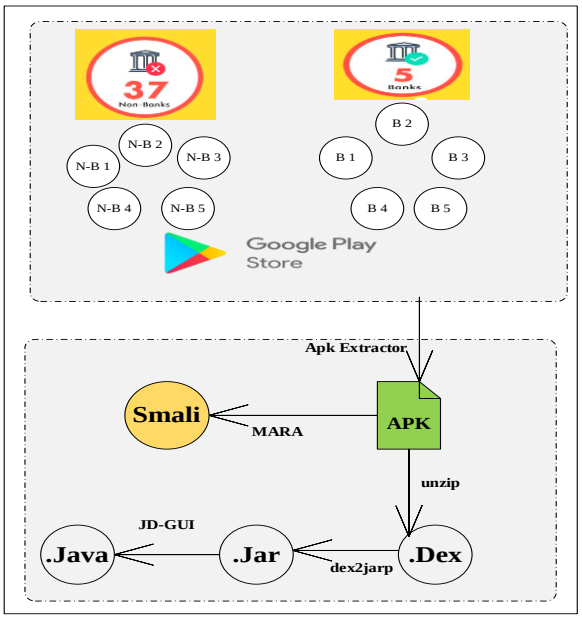


Fig. 2. Reverse Engineering Process [29].

The Mobile Application Reverse Engineering and Analysis (MARA) framework was the technology utilized to execute penetration testing. MARA brings together commonly used reverse engineering and analysis tools for mobile applications to test mobile apps against OWASP mobile security threats and vulnerabilities. MARA is a bash script that combines several prominent android reverse engineering and vulnerability analysis tools everything into solution. The goal was to make the workflow easy to utilize for security researchers and developers. MARA can do dynamic and static analysis requiring no additional post-installation configuration. MARA does not have a graphical user interface; it was created only for terminals. Most of the sub-tools are developed in various Python versions. MARA is evaluated based on six security attributes, which are APK Reverse Engineering, APK Deobfuscation, APK Analysis, APK Manifest Analysis, Domain Analysis, and Security Analysis. Each of the following six security attributes can be found in details [34].

Critical, High, Medium, Low, Info, and Detection issue are the different levels of se-verity [35]. MARA reversed all the chosen applications and retrieved the complete source code. The reverse results obtained by checking the source code individually are provided in Section 4.

3) Automotive testing: This segment would present the automated security checks carried out in the chosen banking and non-bank payment methods, processes and outcomes. Automated protection testing is an essential part of the paper's static vulnerability review, which is valuable since it provides a summary of where vulnerabilities can be found in the application. This chapter starts with a summary on chosen tools, and then explains how they are used then shows the results of automatic safety monitoring. All the apps checked for AndroBugs security have received an initial description of where to check for vulnerabilities, while the findings presented by AndroBugs seemed to provide a clear understanding of the system build and possible vulnerabilities in most instances.

In November 2015, Yu-Cheng Lin released the AndroBugs framework, an open source vulnerability scanner for Android apps. AndroBugs is a Python-based static analysis tool that analyzes for common vulnerabilities in Android apps, it also checks the code is missing best practices and checks dangerous shell commands [29]. AndroBugs seemed to provide a clear understanding of the system build and possible vulnerabilities in most instances. The details of calculated CVSS for AndroBugs can be seen in supplementary file (Table IV) [26]. The possible vulnerabilities can be found in all apps were the following:

- Runtime Command: This is because AndroBugs establish in the code a serious function "Runtime.getRuntime ().exec ("…")". This feature allows a user to enter the shell and then modify the commands within it.
- Fragment Vulnerability: Since AndroBugs found a 'Fragment' or 'ActionbarSherlock Fragment' in the software, which was vulnerable to Android before 4.4 on phones. Any intruder who runs a code capable of eroding the Android Sandbox, which means access to confidential information not accessible by the application, is vulnerable to using this application on an old Android device.
- SSL Certificate Verification: However, this application does not validate the SSL certificate validation, so it causes the self-signed Common Name (CN) certificates for Secure Sockets Layer (SSL) to be expired or to be unacceptable. This is undoubtedly a vital weakness, since it enables attackers to carry out Man in the middle (MITM) attacks.
- SSL Implementation: That ensures that such a self-defined application will accept all common names as "HOSTNAME VERIFIER." This allows any attacker with a valid certificate to carry out MITM attacks.
- Implicit Service: In other words, this application uses an implicit decision to conduct a service, which is dangerous, since the answering service is not identifiable and the user cannot see which service.
- Web View Vulnerability: This implies that AndroBugs find the "addJavaScriptInter-face" method in an application code, which a weakness that JavaScript may use in devices is running android before 4.2 to manage the application.
- Android Manifest: This shows that this app has high privileges for AndroBugs. An-droBugs find that the "Mount Unmount FileSystems" android permission is included in this program, which has not been justified as the permission authorization permits removable mounting and mounting of file systems and the Android developer website notes that the application is not used by third party applications. The application is not mounted.

- Key Store Protection: AndroBugs therefore find that this application does not adequately secure its Key Store because it appears that it uses byte array and SSL pinning using a hard-coded certificate information. The details of the reverse engineering process are mentioned in Section 2.4, respectively. The existing work and their finding is shown in Table V.

TABLE V. THE EXISTING WORK AND THEIR FINDING

Authors	Method	Country	Finding
Reaves et al. 2015 [39]	Manual analysis	USA	This article completed manual analysis on 7 Android m-banking apps. These apps were tested for SSL/TLS bugs, cryptography, and identity leakage and access control. The findings confirm that the majority of these apps fail to provide the protections needed by financial services.
Filiol and Irolla, 2015 [37]	Static and Dynamic analysis	France	This study executed static and dynamic analysis on 50 Android m-banking apps
Zheng et al. 2017 [38]	Repackaging Attack	Australia	This article examine common security attacks on mobile apps, whether they are performing preliminary tests to determine the effectiveness and complexity of mobile device security attacks using repackaging attacks to obtain victim information.
Chothia et al. 2017 [36]	TLS testing methods	UK	This article presents a security analysis of the 15 m-banking apps issued by leading UK banks. The primary goal was to find the bugs in these apps' TLS implementations.
Chanajitt et al. 2018	Forensic analysis	Thailand	This articles focus on seven Android m-banking apps in Thailand. Several of the applications examined do not perform root device identification, do not encrypted user data, or may be modified and installed as repackaged apps.
Bassolé et al. 2019 [40]	Vulnerability assessments	Africa	This article analyzed the vulnerability of mobile banking and payment applications on Android platforms. This article undertakes vulnerability assessments, allowing for a more informed analysis of the information security and privacy threats that African mobile banking and payment applications face. They specially evaluate login credentials and code vulnerability of these apps in particular to assess the risks of attacks connected to privacy and data confidentiality.
Yang et al. 2019 [41]	Comprehensive analysis	China	This article examines the existing third-party mobile payment ecosystem and identifies possible security concerns by doing an in-depth assessment against China, the world's largest mobile payment market. Aside from that, this article also uncovers seven incidences of security rule violations on the Android and IOS platforms.
Verderame et al. 2020 [45]	Static and Dynamic Analysis	Italy	This article describes a unique methodology based on a successful mix of static analysis, dynamic analysis, and machine learning techniques for determining whether a particular app either) has a Google Play privacy policy and ii) accesses privacy. This article also involves examining the compliance of third-party libraries that are incorporated in existing applications.
Majeti et al. 2021 [46]	Cryptographic primitives	India	This article looks at how cryptographic primitives are used in Indian mobile finance apps. They chose 36 apps from three distinct categories and evaluated the flaws separately.
Our study	Static analysis	Malaysia	To perform static analysis of 5 m-banking and non-bank e-wallet apps. The static analysis has been done by using three mobile application-testing tools that is recommended by OWASP.

III. METHODOLOGY OF PROPOSED STUDY

This section presents the methods, processes and results of the automated security tests on the applications pre-selected. The automated safety testing is part of the research paper static vulnerability analysis, with an overview of where vulnerabilities can be found. The analysis is important. This chapter starts with a briefing on selected the analysis tools, and then demonstrates the method and results of the automated security testing. Some of the previous study conducted static or dynamic analyzes among various country leading m-banking apps. The focus of this study is to analyze popular Malaysian e-wallet apps and m-banking apps to identify the security.

A. Information Gathering and Setup

Mobile testing tools can assist organizations in automating Android and iOS testing. The software for mobile application testing can minimize the time required for the test and the probability of human mistakes during testing. Varieties of technologies are available for testers to automate their test scripts nowadays. For the success of the objective, it is essential to choose the right path for particular apps. Various systems may have various risks. The key problem, for example, would be diversion of funds in a bank application.

For object testing, authors utilized a DELL machine with an Intel Core i7 CPU, 3.40 GHz CPU, and 6 GB RAM. The operating system is Windows 10 professional. A virtual machine has been installed name as VMware to create dual boot in the computer. The testing involved the process of first installing Kali Linux, Operating System (OS).

B. Analysis Process and Testing Object

Intruders do the test to evaluate if there are any flaws or weaknesses that can allow the penetration and exploitations during its operation [41]. The e-wallet applications for Android

Redmi 8 are executed to start research, to check if they are running without any error. The study included a variation of vulnerability scanning, code review and, most significantly, penetration testing. AndroBugs is used to automate security-testing tool, where MobSF, used for basic static analysis report. Finally, MARA tools used for reverse engineering checking. In checking mobile apps against the OWASP, MARA builds widely used reverse engineering and research techniques to test mobile applications [42].

The object contains e-wallet applications from leading and growing banks in Malaysia. The platform of Google Play Store, the official site for Android-based smartphone apps downloaded e-wallet applications on the mobile phone. The applications were then transmitted via a universal serial bus serial interface to the computer. A folder was then created with Malaysia e-wallet name, which was then dumped in the e-wallet APK for-mat. APK extension dumped on the desktop. All the selected applications have been security tested by Mobile Security Framework (MobSF), MARA and AndroBugs tool an immediate idea of where weaknesses might be found. The tools findings seem to provide a clear understanding of how the programs are designed and where it could lead to potential vulnerabilities. It was beneficial as a guide for static vulnerability analysis and to know where the weakness available. Fig. 3 shows the selected dataset of e-wallet programs with the analysis process.

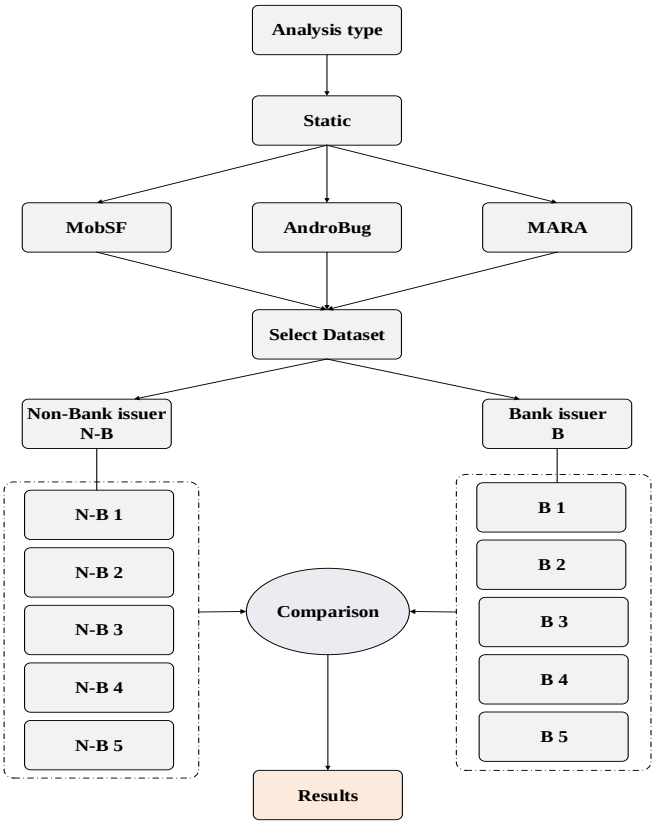


Fig. 3. Selected Testing Object for Analysis.

IV. Analysis Result

After installation of the OS, updates and patches for the operating system were then installed from the Linux public repository to update the libraries that used as prerequisites for its installation and operation. The applications were then transferred to the computer via a universal serial bus data cable. The applications were then analyzed one by one tools and reports created and dumped in their respective application folders. The result of non-banks e-wallet apps using MobSF is showing Table VI.

Table VI shows the comparison result of select non-bank e-wallet apps. The analysis report is divided into three categories, such as security score, average Common Vulnerability Scoring System (CVSS) and tracker detection. The security score refers to the overall security results where the CVSS is an open framework for interactive the characteristics and severity of software vulnerabilities. Finally, tracker detection vulnerability checks and evaluates IT network and any device linked to it against thousands of Network Vulnerability Tests (NVTs) [43]. The most active security score in the analysis report is N-B4, which was observed 45 percent of the time, with an average CVSS of 7.0 and 8-tracker detection, which was the top security score APK. The second highest positions security score obtain from N-B3 with 40%, which CVSS rate average 6.4 with 6-tracker detection. From N-B1, N-B2 and N-B5, the same security score has been identified which 10% respectively. Nevertheless, in point of view their CVSS and tracker detection are not similar to their security score. The average CVSS of N-B1 and N-B2 are similar to 6.5 whether N-B5 is 6.9. The tracker detection rates are 6/323, 7/323, 3/323 and 2/323.

Table VII shows the comparison result of select bank issuer mobile apps. The most active security score in analysis report is, B2 with 85%, which average 7.5 CVSS with 0-tracker detection, which was the top security score APK. The second highest positions security score obtain from B1, B3, and B4, with same security score which 10% respectively. Nevertheless, in point of view their CVSS and tracker detection are not similar like their security score. The average CVSS of B1 with 6.7, B3 is 6.6, and B4 is 6.1. Whether the tracker detection rates are 2/319, 3/323, and 3/319. From B5 authors could not find the security score but except Average CVSS 6.8 and Trackers Detection 8/319, respectively. The result of banking apps using MobSF is shown in Table VII. The analysis report of non-banking e-wallet apps is shown in Table VIII.

TABLE VI. Result of Non-banks e-Wallet Apps using MobSF

Wallet name	Security Score	Average CVSS	Trackers Detection
N-B1	10/100	6.5	6/323
N-B2	10/100	6.5	7/323
N-B3	40/100	6.4	6/323
N-B4	45/100	7.0	8/323
N-B5	10/100	6.9	2/323

TABLE VII. RESULT OF BANKS APPS USING MOBSF

Wallet name	Security Score	Average CVSS	Trackers Detection
B1	10/100	6.7	2/319
B2	85/100	7.5	0/319
B3	10/100	6.6	3/323
B4	10/100	6.1	3/319
B5	-	6.8	8/319

TABLE VIII. REPORT OF NON-BANKS APPS USING MARA

Wallet name	Critical	High	Medium	Low	Info	Detection Issue
N-B1	0	-	1	-	-	-
N-B2	0	-	1	-	-	-
N-B3	0	4	2	2	11	19
N-B4	0	6	1	2	11	20
N-B5	0	2	1	2	10	15

Table VIII shows each of the e-wallet application has 0 critical issues. From N-B4 with total 20 detection issues 6 high, 1 medium, 2 low and 11 info analysis report. In N-B3 total 19 detection issues has been collected where 4 high, 2 medium, 2 low and 11 info analysis report which the second highest. The third positions is N-B5 with total 15 detection issues where 2 high, 1 medium, 2 low and 10 info analysis report. N-B1 and N-B2 there is no critical issue but due to system trouble-shoot authors could not get the exact information of both wallet. The result of banking apps using MARA tool is shown in Table IX.

Table IX shows the comparison result of select bank issuer mobile Apps. Each of the banking application has 0 critical issues. From B1 total 19 detection is-sues has been collected where 4 high, 2 medium, 0 low and 11 info analysis report. From B3 and B5 collected a similar value total 17 detection issues where 3 high, 2 medium, 2 low and 10 info analysis report which the second highest, respectively. The third positions is B5 with a total 17 detection issues where 3 high, 2 medium, 2 low and 10 info analysis report. From B4 Apps there is no critical issue but due to system troubleshoot authors could not get the exact information. Finally, B1 has 0 critical issues with high threats, 2 medium and low threats along with 10 info are the most secure application compared to others. The result of banks and non-banks reports using MARA AndroBugs are shown in Table X and Table XI.

TABLE IX. REPORT OF BANKS APPS USING MARA

Wallet name	Critical	High	Medium	Low	Info	Detection Issue
B1	0	4	2	0	11	19
B2	0	0	2	0	6	8
B3	0	3	2	2	10	17
B4	-	-	-	-	-	-
B5	0	3	2	2	10	17

Table X and Table XI shows below, the comparison result of select bank and non-issuer mobile apps using AndroBugs. Each of the application has different kind of critical issue. The analysis report has been categorized into two part parts. From B4 with total 6 issues which is the most critical issue found in the bank issuer analysis report. From B1 and B5 authors collected similar value total 5 issues which the second highest, respectively. The third position is B3 with 3 issues.

TABLE X. RESULT OF BANKS REPORT USING ANDROBUGS

S/n	Properties	B1	B2	B3	B4	B5
1	Runtime Command Checking	No	No	No	No	Yes
2	Base64 String Encryption	No	No	No	No	No
3	SSL Security	No	No	No	No	No
4	Key Store	No	No	No	Yes	Yes
5	Implicit Intent	Yes	No	Yes	Yes	Yes
6	SSL Implementation Checking	Yes	No	No	Yes	No
7	SSL Connection Checking	Yes	No	Yes	No	Yes
8	SSL Certificate Verification Checking	No	No	No	Yes	No
9	<Web View>/Remote Code Execution	Yes	Yes	Yes	Yes	Yes
10	Fragment Vulnerability Checking	Yes	No	No	No	No
11	Android Manifest Critical Use Permission Checking	No	Yes	No	Yes	No

From B2 only single issues have been collected, which is the most secure using AndroBugs analysis. From non-bank, issuer author's collected total 9 issues from N-B5 which is the most critical issue found in the analysis report. The second highest positions are N-B2, with total 7 issues. From N-B3 and N-B1, similar value has been collected which are total 3 issues respectively. From N-B4, only single issues have been collected, which is the most secure using AndroBugs analysis.

V. DISCUSSION

Mobile apps are growing increasingly, with more consumers being able to access different forms of Android applications availability of a wide range of open Android markets. However, mobile apps threats are developing especially targeted towards mission critical mobile bank applications [44]. This study first analyze five types of bank and nonbank issuer e-wallet products, and then focus on the vulnerabilities and security issues based on static analysis. We evaluate the flaws in protection, critical security, Average CVSS against Malaysian 5 banking and non-banking e-wallet products publicly available. In this section, will present and analyze the outcomes of a data set security evaluation.

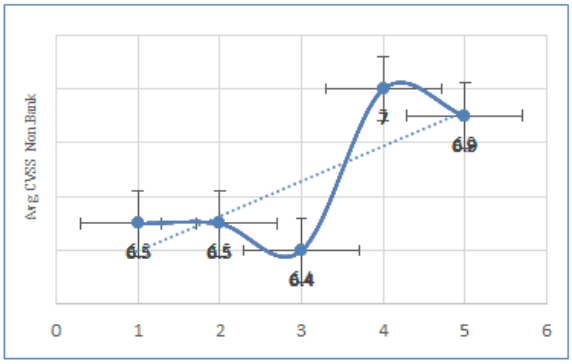
TABLE XI. RESULT OF NON-BANKS REPORT USING ANDROBUGS

S/n	Properties	N-B1	N-B2	N-B3	N-B4	N-B5
1	Runtime Command Checking	No	Yes	No	No	Yes
2	Base64 String Encryption	No	Yes	No	No	Yes
3	SSL Security	No	No	No	No	Yes
4	Key Store	No	No	No	No	Yes
5	Implicit Intent	Yes	Yes	Yes	No	Yes
6	SSL Implementation Checking	No	No	No	No	Yes
7	SSL Connection Checking	Yes	Yes	Yes	No	Yes
8	SSL Certificate Verification Checking	No	Yes	No	No	Yes
9	<Web View>/Remote Code Execution	Yes	Yes	Yes	Yes	Yes
10	Fragment Vulnerability Checking	No	No	No	No	No
11	Android Manifest Critical Use Permission Checking	No	Yes	No	No	No

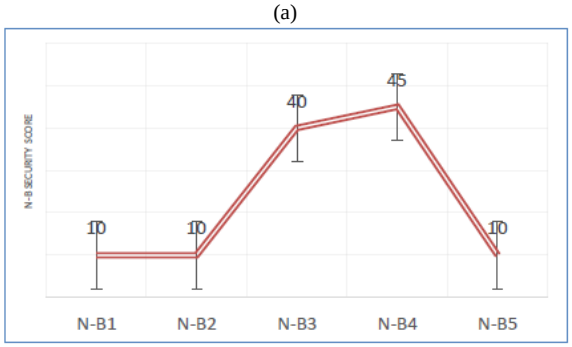
In Fig. 4 presents, the summary of MobSF results of the security tests performed on bank and non-banks e-wallet apps. The most active security score in the analysis report is N-B4, is 45%, which averages 7.0 Common Vulnerability Scoring System (CVSS) with 8-tracker detection, which was the top security score application.

It is clear from the results that N-B4 is quite secure related to the other applications. Table VIII presented the result of banks' apps using MobSF. The most active security score in the analysis report is B2, from which noticed 85%, which average 7.5 CVSS with 0-tracker detection, which was the top security score APK. After the analyzing of Table VIII, found that, N-B5 application have total 15 detection issues where 2 high, 1 medium, 2 low and 01 info analysis report which is quite secure related to the other applications is shown in Fig. 5(a) and Fig. 5(b) shown the high security banking where B3 and B5 are same result, respectively.

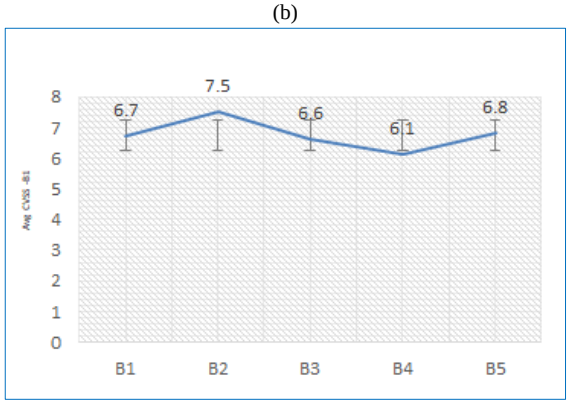
On the other hand, the low security low security apps of banking m-apps using MARA tool is B1, which is shown in Fig. 5 (b). Table IX shows the true seeing a report of banks' apps using MARA. Each of the banking application has 0 critical issues. However, the B3 and B5 have 0 critical issue with 3 high threats, 2 medium and low threats along 10 info and 27 detection issues which is the most secure application com-pared to others. Fig. 5(c) and (d) shown the low security bank and non-bank apps, respectively.



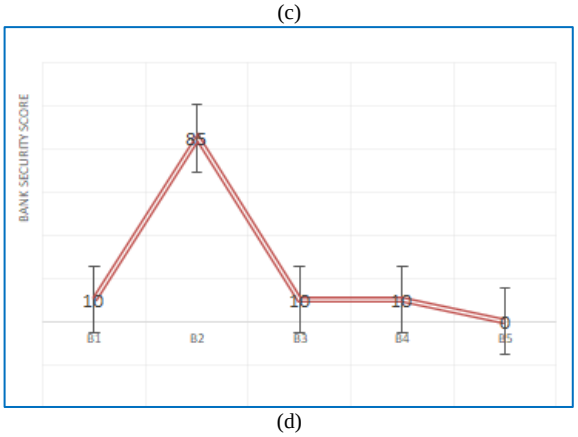
(a)



(b)

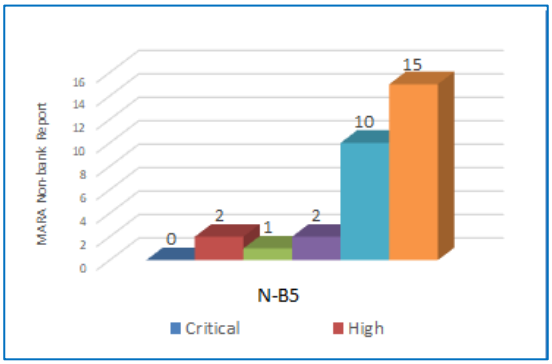


(c)

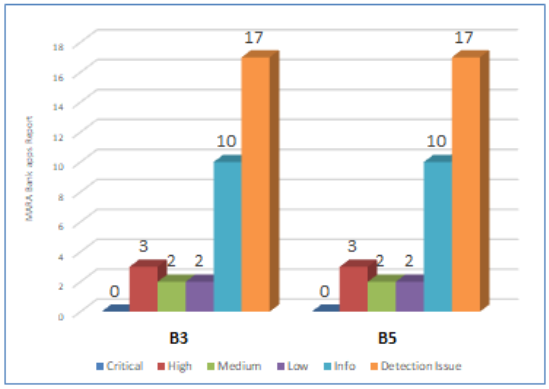


(d)

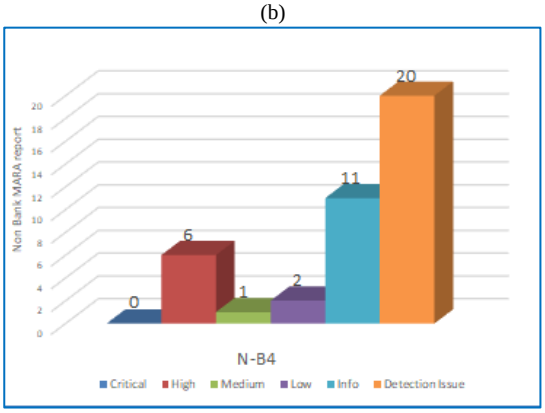
Fig. 4. (a) Average CVSS of Non-banks e-wallet Apps using MobSF Tool and (b) Non-bank e-wallet Apps Security Score using MobSF Tool and (c) Average CVSS of Banks m-banking Apps using MobSF Tool and (d) Bank m-Banking Apps Security Score using MobSF Tool.



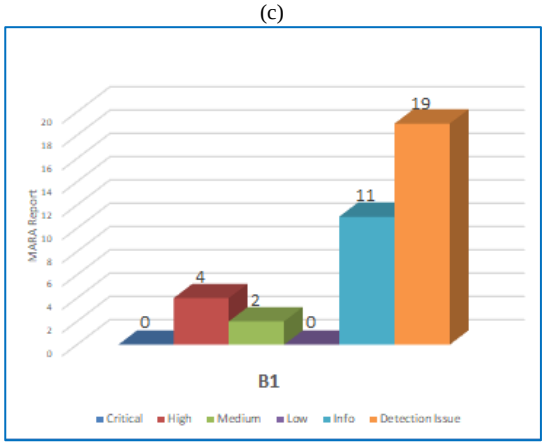
(a)



(b)



(c)



(d)

Fig. 5. (a) High security e-wallet apps using MARA tool and (b) High security banking apps using MARA tool and (c) Low security e-wallet apps using MARA tool and (d) Low security apps of m-apps using MARA tool.

VI. Conclusion

Mobile payment applications are very convenient, but the problem is that most mobile payment apps are not exactly appropriate. Companies and developers would need to limit the addition of features and services demand and continue protecting the apps. However, as explained in this paper, it is quite an impossible task to protect the apps, although nothing is 100% secured, but developers at least might make it much more difficult for hackers. This paper covers a security assessment of five non-bank e-wallet apps and five leading banks apps in Malaysian market. The Authors performed a static analysis on three pen-extraction mobile application-testing tools and compared with the Android application among them. The analysis notice that every apps have followed the security standard, but their security features and properties are different in point of view of how their customer demand. Finally, the most secure e-wallet and m-banking apps according to the three different tools, based on their security metrics, have been identified which is not our opinion. This study aims to increase research efforts on the progress of e-wallets and m-banking in Malaysia.

Acknowledgment

This research was funded by the Malaysia Ministry of Education, Universiti Kebangsaan Malaysia, under grants, KKP 2020/UKM-UKM/4/3 and FRGS/1/2021/ICT02/UKM/02/1.

References

[1] F. Nizam, H. J. Hwang, and N. Valaei, Measuring the Effectiveness of E-Wallet in Malaysia, vol. 786. Springer International Publishing, 2019.

[2] H. H. Bin Kadar, S. S. B. Sameon, M. B. M. Din, and P. 'Amirah B. A. Rafee, "Malaysia Towards Cashless Society," Lect. Notes Electr. Eng., vol. 565, pp. 34–42, 2019.

[3] L. T. H. Teoh Teng Tenk, Melissa, Hoo Chin Yew, "E-wallet Adoption: A Case in Malaysia," Int. J. Res. Commer. It Manag., vol. 2, no. 4, pp. 135–3, 2020.

[4] A. Hassan, Z. Shukur, and M. K. and A. S. A.-K. Hasan, "A Review on Electronic Payments Security," Symmetry (Basel)., vol. 12, no. 8, p. 24, 2020.

[5] M. Salah Uddin and A. Yesmin Akhi, "E-Wallet System for Bangladesh an Electronic Payment System," Int. J. Model. Optim., vol. 4, no. 3, pp. 216–219, 2014.

[6] S. Z. Jesús Téllez Isaac, "Secure Mobile Payment Systems," J. Enterp. Inf. Manag., vol. 22, no. 3, pp. 317–345, 2014.

[7] M. A. Hassan and Z. Shukur, "Review of Digital Wallet Requirements," 2019 Int. Conf. Cybersecurity, ICoCSec 2019, pp. 43–48, 2019.

[8] R. Kaur, Y. Li, J. Iqbal, H. Gonzalez, and N. Stakhanova, "A Security Assessment of HCE-NFC Enabled E-Wallet Banking Android Apps," Proc. - Int. Comput. Softw. Appl. Conf., vol. 2, pp. 492–497, 2018.

[9] IBM, "IBM Sponsored Study Finds Mobile App Developers Not Investing in Security." [Online]. Available: https://www-03.ibm.com/press/us/en/pressrelease/46360.wss. [Accessed: 15-Nov-2020].

[10] EcInsider, "The e-wallet infinity war in Malaysia - Everything you need to know about e-wallet starts here." [Online]. Available: https://www.ecinsider.my/2018/12/malaysia-ewallet-battle-landscape-analysis.html. [Accessed: 01-Nov-2019].

[11] A. Hassan, Z. Shukur, and M. K. Hasan, "Electronic Wallet Payment System in Malaysia," Data Anal. Manag., vol. 54, pp. 711–736, 2021.

[12] Y. Wang et al., "Identifying vulnerabilities of SSL/TLS certificate verification in Android apps with static and dynamic analysis," J. Syst. Softw., vol. 167, 2020.

[13] P. Aigbe and J. Akpojaro, "Analysis of Security Issues in Electronic Payment Systems," Int. J. Comput. Appl., vol. 108, no. 10, pp. 10–14, 2014.

[14] M. A. Kabir, S. Z. Saidin, and A. Ahmi, "Adoption of e-payment systems : a review of literature," Proc. Int. Conf. E-Commerce, no. May 2016, pp. 112–120, 2015.

[15] Rancha and P. Singh, "Issues and Challenges of Electronic Payment Systems," Int. J. Res. Manag. Pharmacy(IJRMP), vol. 2, no. 9, pp. 25–30, 2013.

[16] R. Batra and N. Kalra, "Are Digital Wallets the New Currency?," 2016.

[17] M. Olsen, J. Hedman, and R. Vatrapu, "E-wallet properties," Proc. - 2011 10th Int. Conf. Mob. Business, ICMB 2011, pp. 158–165, 2011.

[18] M. A. Hassan Z. Shukur, M. K. Hasan "An efficient secure electronic payment system for e-commerce," Computers., 9(3), 66, 2020.

[19] R. Safeena, H. Date, A. Kammani, and N. Hundewale, "Technology Adoption and Indian Consumers: Study on Mobile Banking," Int. J. Comput. Theory Eng., no. December, pp. 1020–1024, 2012.

[20] S. Shaju and V. Panchami, "BISC authentication algorithm: An efficient new authentication algorithm using three factor authentication for mobile banking," Proc. 2016 Online Int. Conf. Green Eng. Technol. IC-GET 2016, pp. 1–5, 2017.

[21] J. Juremi, "A Secure Integrated E-Wallet Mobile Application For Education Institution," Int. Conf. cyber Relig., 2021.

[22] J. Zhang and Y. Luximon, "A quantitative diary study of perceptions of security in mobile payment transactions," Behav. Inf. Technol., vol. 0, no. 0, pp. 1–24, 2020.

[23] M. H. Sherif, Protocols for Electronic Commerce, vol. 53, no. 9. 2016.

[24] OWASP, "Owasp-Asvs," no. October, 2015.

[25] A. Abraham, "Improve security scoring of apps." 2020.

[26] A. Maharjan, "Ranking of android apps based on security evidences," no. December, 2020.

[27] A. Abraham and S. Dominik, "Mobile Security Framework." 2019.

[28] V. Kouliaridis, G. Kambourakis, E. Chatzoglou, D. Geneiatakis, and H. Wang, "Dissecting contact tracing apps in the Android platform," PLoS One, vol. 16, no. 5 May, pp. 1–28, 2021.

[29] H. Darvish and M. Husain, "Security Analysis of Mobile Money Applications on Android," Proc. - 2018 IEEE Int. Conf. Big Data, Big Data 2018, pp. 3072–3078, 2019.

[30] PCI Security Standards Council LLC., "PCI Mobile Payment Acceptance Security Guidelines for developers," Pci Dss Inf. Suppl., no. February, pp. 0–27, 2013.

[31] Enisa, Security of Mobile Payments and Digital Wallets, no. December. European Union Agency for Network and Information Security (ENISA), 2016.

[32] T. McDonnell, B. Ray, and M. Kim, "An empirical study of API stability and adoption in the android ecosystem," IEEE Int. Conf. Softw. Maintenance, ICSM, pp. 70–79, 2013.

[33] R. Chanajitt, W. Viriyasitavat, and K. K. R. Choo, "Forensic analysis and security assessment of Android m-banking apps," Aust. J. Forensic Sci., vol. 50, no. 1, pp. 3–19, 2018.

[34] J. Due, "MARA - A Mobile Application Reverse Engineering And Analysis Framework." hacking.reviews, 2017.

[35] D. G. N. Benitez-Mejia, G. Sanchez-Perez, and L. K. Toscano-Medina, "Android applications and security breach," 2016 3rd International Conference on Digital Information Processing, Data Mining, and Wireless Communications, DIPDMWC 2016. pp. 164–169, 2016.

[36] T. Chothia, F. D. Garcia, C. Heppel, and C. M. M. Stone, "Why banker bob (Still) Can't Get TLS right: A security analysis of TLS in leading UK banking apps," Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 10322 LNCS. pp. 579–597, 2017.

[37] E. Filiol and P. Irolla, "(In)Security of Mobile Banking and of Other Mobile Apps," Black Hat Asia, pp. 1–22, 2015.

[38] X. Zheng, L. Pan, and E. Yilmaz, "Security analysis of modern mission critical android mobile applications," ACM Int. Conf. Proceeding Ser., no. October, 2017.

[39] B. Reaves, N. Scaife, A. Bates, P. Traynor, and K. R. B. Butler, "Mo(bile) Money, Mo(bile) Problems: Analysis of branchless banking applications in the developing world," Proceedings of the 24th USENIX Security Symposium. pp. 17–32, 2015.

[40] D. Bassolé, G. Koala, Y. Traoré, and O. Sié, "Vulnerability Analysis in Mobile Banking and Payment Applications on Android in African Countries," Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST, vol. 321 LNICST. pp. 164–175, 2020.

[41] S. A. Chaudhry, M. S. Farash, H. Naqvi, and M. Sher, "A secure and efficient authenticated encryption for electronic payment systems using elliptic curve cryptography," Electron. Commer. Res., vol. 16, no. 1, pp. 113–139, 2016.

[42] M. G. Manoti, "Enhancing Security of Mobile Banking and Payments in Kenya," no. November, 2016.

[43] G2, "NNT Vulnerability Tracker," 2020. [Online]. Available: https://www.g2.com/products/nnt-vulnerability-tracker/reviews. [Accessed: 17-Nov-2020].

[44] M. A. Hassan and Z. Shukur, "Device Identity-Based User Authentication on Electronic Payment System for Secure E-Wallet Apps," Electronics., vol. 11, no. 1, pp. 1–29, 2022.

[45] L. Verderame, D. Caputo, A. Romdhana, and A. Merlo, "On the (Un)Reliability of Privacy Policies in Android Apps," Proc. Int. Jt. Conf. Neural Networks, 2020.

[46] S. S. Majeti, B. Janet, and N. P. Dhavale, "Analysis of Inappropriate Usage of Cryptographic Primitives in Indian Mobile Financial Applications," Lecture Notes on Data Engineering and Communications Technologies, vol. 56. pp. 211–220, 2021.