INTERNATIONAL JOURNAL OF ACADEMIC RESEARCH IN BUSINESS & SOCIAL SCIENCES

Vol 14, Issue 10, (2024) E-ISSN: 2222-6990

Static Security Analysis of Government and Non-Government Android Mobile Applications in Malaysia: A Comparative Study Using MobSF and OWASP Mobile Top 10

Dhanesh a/l Paramasivam[1], Noor Lees Ismail[2], Abdulaziz Al-Nahari3

[1]Faculty of Business and Technology, UNITAR International University, Kelana Jaya, Selangor,
[2]School of Information Technology, UNITAR International University, Kelana Jaya, Selangor,
[3]Information Technology Department, The University of Technology and Applied Sciences, Nizwa, Oman

Email: [1]mc211015323@student.unitar.my, [2]abdulaziz.alnahari@utas.edu.om
Corresponding Author Email: lees@unitar.my

To Link this Article: http://dx.doi.org/10.6007/IJARBSS/v14-i10/22955 DOI:10.6007/IJARBSS/v14-i10/22955

Published Date: 26 October 2024

Abstract

In the current era, mobile phones have become a priority in everybody's life as they are commonly used for communication, business purposes, and other important reasons. Mobile applications are software created to run on a mobile device, and they have become part of our day-to-day lives that many things can be accomplished such as shopping, social networking, banking, and gaming, among others. Since mobile applications deal with sensitive and confidential information that could be misused by malicious agents, security in mobile applications is a crucial issue that must be tackled. In this context, we have performed a static security analysis using a well-known and recognized tool called MobSF Security Framework on Malaysian government and non-government Android mobile applications. This study's novelty is the static analysis to compare the security statuses of both categories of mobile applications based on OWASP Mobile Top 10 and the tools' scoring system. The most common vulnerabilities for both government and non-government mobile applications were identified based on OWASP Mobile Top 10 and security recommendations for each domain were discussed. On the other hand, scoring results from MobSF Security Score identified the safest and the least safe mobile applications among the tested applications.

Keywords: Mobile Applications, Static Analysis, Mobsf Security Framework, Owasp Mobile Top 10, Government And Non-Government Apps

Introduction

Mobile phones are highly capable mobile computing gadgets that have significantly transformed how individuals do business, communicate with online services, and obtain entertainment. Several services offered by a mobile phone are further boosted by an internet connection, commonly supplied by a cellular network or Wi-Fi network. Internet access permits the mobile phone to deliver and accept messages, surf the internet, backup files, etc. Mobile phone's functionality is heightened by an ecosystem of mobile applications covering the whole range of functionality. A noteworthy rise has been observed in the utilization of smartphones over the past era (PewResearch.Org, 2021). The instant advancement of technology has made communication and data transfer exceptionally quick, easy, expedient, and effortless. In this digital age, people anticipate being able to get the latest information and news right away with a single tap of a button. Adopting these technological advancements not only provides convenience and comfort but also offers critical infrastructure and financial gains to local governments, as it eliminates the need for more costly or complicated physical communication networks, especially in rural communities.

The vast acceptance of smartphones has also accelerated the development of mobile applications that run on these gadgets. Mobile applications are primarily distributed via accessible central repositories known as mobile application stores or marketplaces. Mobile application stores allow third-party application developers to distribute their mobile applications for users to download and use. Typically, application developers devote their time to making mobile applications with the prospect of getting a return on their investment. The mobile application industry has expanded exponentially in the last era, the number of unique mobile phone subscribers is estimated to increase from 6.1 billion at the end of 2021 to 6.7 billion by the end of 2027, according to the Ericsson Mobility Report (Ericsson, 2022). Android and iOS are known to be the two extremely common mobile phone operating systems in the mobile application marketplace. Based on another study from Statista (Statista, 2022a), Android retained its standing as the prominent mobile phone operating system globally within August 2022, conquering the mobile operating system market with a near 70 percent portion, whereas iOS occupies about 28 percent from the mobile operating system marketplace. As of June 2022, Google Play Store had 2.65 million mobile applications (Statista, 2022b).

As the technological world persists in developing and evolving, the government needs to maintain pace with these differences by designing multiple channels for providing government-related services. The way the government is employing to enhance the manner they communicate information to citizens is through creating mobile applications. With mobile applications, government organizations can communicate valuable information such as crisis alerts or specific news announcements swiftly and effortlessly. According to 42matters (42matters AG, 2022), there are more than 8,071 applications developed by Malaysian developers on Google Play from the 3,117,565 applications and the Government of Malaysia is sitting in the second position of the largest Malaysian application publishers. Malaysian government mobile applications provide a variety of services such as general information, announcements and news, financial services, government documents, crowdsourcing programmes, well-being, and safety knowledge and learning facilities.

As the developing technology has undoubtedly made life simpler and more useful for the whole world, it has additionally created many new risks, threats, and security problems and opened the prospect door for cybercriminals. With mobile phone technology's improved approach to communicating and networking with people all across the world, it has additionally been overlaying the path for simpler entry to our confidential and private information. Mobile phones have undoubtedly provided considerable advantages to users, but they also contribute their reasonable share of disadvantages, such as increases in security threats and the destruction of user confidentiality. Information security is undoubtedly the most prominent fear among mobile phone users as mobile applications generally deal with highly confidential personal and private information that can be exploited and misused by hateful threat actors that could damage users' gadgets, personalities, along their status. Security weaknesses in mobile applications can specifically platform or application; additional issues involve server or client-side and system-associated incidents.

There are coding problems and security misconfigurations, system information that could be stolen whilst being transferred or at static, etc. All through the research described here, it can be observed that numerous safety issues are occurring in numerous mobile applications, and it is essential to carry out assessments that allow us to assess the toughness of the mobile applications versus the challenges. Though security assessment is not the only important means to thwart security challenges, it assists in identifying as well as locating security flaws occurring in mobile applications. Commonly, the absence or lack of security is envisioned by the lack of security measures taken by software developers (Macy Bayern, 2019), which indirectly means that many of the software that goes into the production stage is susceptible to one or more categories of cyberattacks. Likewise, cybersecurity attacks like phishing emails (Sharma et al., 2020) and social engineering approaches for spear phishing (Krombholz et al., 2014) are progressing actively with technical development which is flooring the path for assailants to trick users into slipping into their system to obtain access to their private data. In opposition to this, organizations like OWASP are continuously working to discover, evaluate, and establish ways to alleviate software flaws and vulnerabilities, contributing to the approaches, measures, classifications, and different ways of the community to resolve problems due to software weaknesses. As many mobile applications depend on customer data and continuously interconnect via network systems with remote devices like API and servers, it is vital to make certain that the data is safeguarded inside the mobile gadget while it is being communicated over a message transfer channel like Bluetooth, Wi-Fi, etc.

Though new developments and innovations of technologies could offer the potential of productivity improvements and new competencies, they may also present new dangers. Hence, the situation is crucial for mobile application creators and security researchers to know the current security status of mobile applications in Malaysia and the vulnerabilities that are common, pervasive, and serious in mobile applications that are used by many Malaysian citizens for their daily tasks. Intended for this purpose, we present a study that includes all these characteristics together with the absent matters observed in the literature evaluated.

Problem Background and Related Work

Cyber-attacks are becoming complicated, with attackers utilizing a broader range of strategies to perform the attacks. A recent study (Ruth, 2021) described that mobile applications published in the first quarter of 2021 through 18 of the best well-known classifications have 39 vulnerabilities or weaknesses per application on average which can influence the security and confidentiality of the mobile application users. Mobile phone's sophisticated processing abilities permit most mobile applications to store and deal with sensitive and confidential identifiable information. Mobile phone users are utilizing various types of mobile applications for their regular living routine, even with no understanding of the possible adverse effects. Mobile phones' increasing acceptance in users' day-to-day pursuits is defining and identifying mobile computing outpouring. To keep this condition under control, it is important to focus on mobile applications' confidentiality problems and detect important worries from confidentiality and safety viewpoints. According to a recent news article from the New Straits Times (Mokhtar, 2022), a group of hackers identifying themselves as the 'grey hat cybersecurity organization' allegedly claimed that they could break into the civil servants' payroll system known as ePenyata Gaji (ePaySlip) to demonstrate that there are vulnerabilities in the system. The information that could be retrieved from the system includes full name, identification number, position, salary, mobile phone number, and email address. There is also, an alleged data leak (New Straits Times, 2021) comprising the data of 22.5 million Malaysian citizens who were born from 1940 to 2004, seemingly retrieved from the National Registration Department (NRD) was being sold for US$10,000 on the dark web.

The data was allegedly hacked and stolen from the NRD via MyIdentity's application programming interface. According to another news article from Bloomberg (Yantoultra Ngui, 2021), a vulnerability in the API (application programming interface) of Malaysia's Covid-19 contract tracing application, MySejahtera was allegedly utilized by cybercriminals to send fake emails to the users. Complaints about getting unknown OTP and fake emails from MySejahtera application were lodged by the application users after they received OTP at unusual hours and even emails saying that they had been confirmed Covid19 positive. As government mobile applications could deal with a significant amount of Malaysian resident's information and personal data, the impact of a cyber-attack could be devastating. Based on a study by Homeland Security on the security of mobile devices, dangers, and perils to the Government's usage of mobile devices are factual and occur throughout all components of the mobile phone environment (Homeland Security, 2017). With vulnerable government mobile applications, sensitive citizen data, such as Personally Identifiable Information (PII) could get easily compromised, which might be severe for users. One could argue that by not sufficiently safeguarding their applications, governments are putting citizen's sensitive personal data at risk. In this context, security is considered a major part of a mobile application. Hence, the assessment of the security of government mobile applications should be thoroughly conducted to ensure it is safe to be used by citizens of Malaysia. Therefore, the main objective of this paper is:

- To evaluate and compare the current security status of the Malaysian government and non-government mobile applications based on OWASP Mobile Top 10 risks and MobSF Security Score

OWASP Mobile Top 10

The Open Web Application Security Project (OWASP) is a virtual society of security professionals that has developed easily accessible knowledge resources, documents, and tools to better develop safe web and mobile applications. The OWASP foundation offers security recommendations and suggestions for software security. Every one of the OWASP tools, documentation, and conferences are available for free and accessible to everyone concerned with enhancing and also understanding application security. A significant contribution of OWASP to mobile application security is the documentation of the top 10 flaws and vulnerabilities that are present in mobile applications under the name OWASP Mobile Top 10. The OWASP Mobile Top 10 (OWASP, 2016) list comprises common security weaknesses as well as vulnerabilities in mobile applications and provides the best recommendations to assist in remediating and reducing these security concerns. The OWASP Mobile Top 10 list is an exceptional source for mobile application creators who choose to develop safe and reliable mobile applications. This list is revised and updated based on data from numerous security-dedicated individuals and organizations. Weaknesses consist of flaws, vulnerabilities, coding issues, bugs, errors in implementation, and every other fault that can cause a cyber-attack. This list is important to help draw attention to security vulnerabilities in mobile applications and develop appropriate defenses that can cope with security attacks that actively exploit application functionality.

In this perspective, the OWASP Foundation has collected numerous methodologies and the most widespread weaknesses in mobile applications. Consequently, OWASP established the top ten mobile threats in its final edition in the year 2016 as a beginning standard for application developers and mobile security researchers to assess mobile applications. Ashleigh Lee (2018) tested mobile applications accessible on the Google Play Store and Apple Store and uncovered that 85% of the mobile applications breach at least an item 15 listed in the OWASP Mobile Top 10 threats. Among the tested mobile applications, half of them have insecure data storage, and just about an equal number of applications utilize insecure communication.

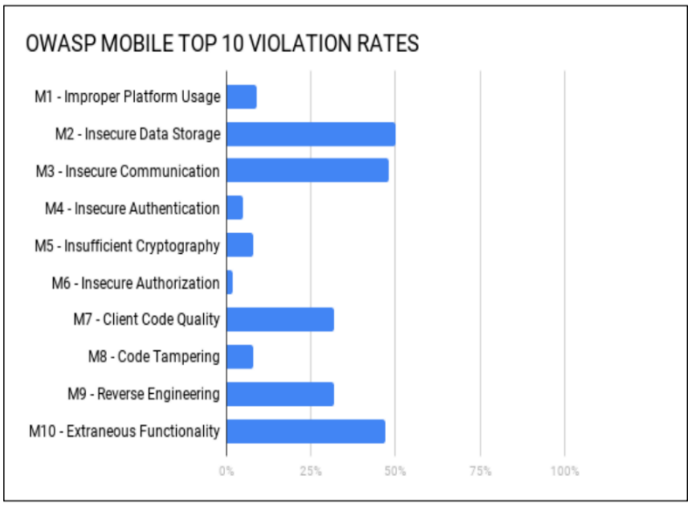

Figure 1. OWASP Mobile Top 10 violation rules

Based on surveys and feedback gathered from the international mobile and security community, OWASP has compiled a list of the top ten mobile security risks (OWASP, 2016a). A summary of the OWASP Mobile Top 10 categories is as follows:

Improper Platform Usage

Improper platform usage represents the condition in which mobile application creators fail to employ or improperly use an element of the mobile phone's operating system (OWASP, 2016b). According to the report from Positive Technologies, one-third of weaknesses in Android mobile applications are from setting shortcomings (Positive Technologies, 2019). These vulnerabilities generally are because of mishandling of the platform application interface programming (API) and the deficiency of employing appropriate safety mechanisms recommended by OWASP. It also involves mismanagement of Android intents, platform permissions, or security mechanisms such as TouchID or the Keychain.

Insecure Data Storage

Insecure data storage is the subsequent vulnerability from the OWASP Mobile TOP 10. According to the report from Positive Technologies, 76% of mobile applications contain this security flaw (Positive Technologies, 2019). Problems associated with unsafe information storage and unintentional data leaks are included in this group. Failure to encrypt or safely store data is a popular issue that can lead to data leaks. File systems are effortlessly available for an attacker that either acquired the physical device or with the assistance of malware has access to sensitive data storage. The data leakage could also occur because of weaknesses in frameworks, hardware, or because of rooted devices (Sai et al., 2019). Mobile applications deprived of appropriate checks for information leaks could endure common exploitable weaknesses or flaws.

Insecure Communication

Insecure communication issues involve transmission as well as inter-process transmission problems. In accordance with the report from Positive Technologies, 35% of mobile applications transmit information to external systems insecurely and twenty-nine percent mobile applications use inter-process communication insecurely (Positive Technologies, 2019). These security problems occur once the information being transferred fails to be properly safeguarded during communication with trustworthy endpoints. It also arises when network traffic is sent over an unsecured communication channel which could leak sensitive data to be captured or eavesdropped by a threat actor.

Insecure Authentication

Based on OWASP, a mobile application demonstrates insecure authentication once it communicates with sources of a server without exhibiting credentials like user ID and password (OWASP, 2016c). Mobile applications working with private and confidential information such as individual information, health, or financial records must employ adequate authentication of a user. According to the report from Positive Technologies, 53% of mobile applications fail to keep authentication information in a safe approach, and 41% are found to execute local authentication (Positive Technologies, 2019). This category encompasses issues that arise after the authentication and session organization of end users are performed inadequately. Authentication vulnerabilities permit threat agents to bypass identity control systems by presenting service call to the applications backend server. Poor or non-existent authentication systems allow an adversary to gain access to confidential data or sensitive application functionality. One instance of insecure authentication is the situation when the

mobile application allows the appliance to accomplish a back-end API call without the access token.

Insufficient Cryptography

According to OWASP, insufficient cryptography happens when a mobile application employs a faulty method or a vulnerable algorithm (OWASP, 2016d). These security issues occur when cryptography is employed on the confidential information being kept or transmitted, however, the cryptography that is employed itself does not comply with the necessary security requirements. Application of weak encryption algorithms or not complying with security best practices in the encryption process will make it potential for threat agents to exploit and break the encrypted data transferred or kept on the device.

Insecure Authorization

Based on OWASP, this category includes issues linked to authorization such as the existence of Insecure Direct Object Reference (IDOR) weaknesses, obscure endpoints, user role or authorization transmissions, etc (OWASP, 2016e). Through authorization, the mobile application must validate if the authenticated user holds the authorization to execute a particular process. Failure to do so could put the mobile application and the back end in danger. Concerns with the permission of application users to make use of various elements of the application are incorporated within this group. Application of inadequately designed authorization schemes can lead to threat agents evading permission controls and gaining access to confidential data or functionality that is preserved for users with sufficient permission levels. An attacker or a rogue user could compromise the security of the mobile application in a situation where the mobile application permits the user to execute API requests with no appropriate authorization.

Client Code Quality

According to OWASP, this group comprises code-level complications such as memory leaks or buffer overflow (OWASP, 2016f). Applications that permit nontrusted code to be accepted as input and then executed are in danger of attacks that can result in memory leakages or corruption. The code-level errors could facilitate the assailant to take advantage of business logic to evade the security controls in place.

Code Tampering

Mobile applications operate in a setting that is not supervised by the application developers (Sai et al., 2019). This permits assailants to alter the application program code to modify API requests or inject a backdoor to obtain confidential information. Assailants could alter the application code and rebuild the mobile application before issuing it again to users. Making modifications to the sources or application code in the mobile application binary or some other manner that alters the initial application is included beneath this group. An unauthorized and altered version of an application source code can present a threat as an attacker may introduce malicious code into the repackaged application and upload it to the mobile application stores.

Reverse Engineering

Based on OWASP, this category is linked to the extraction of the source code, algorithms, libraries, and other resources from the binaries and executables archives (OWASP, 2016g).

Mobile applications could be reverse-engineered, and the application code, resources, and additional assets could be abused to obtain private or confidential data like intellectual property, information related to the cryptography implementation, and back-end endpoints. By decoding and evaluating an application's core binary, it is likely for an attacker to determine the application's source code and resources implanted within the mobile application. The position possibly will permit the assailant to ascertain and identify the logic of the application in position as well as the deduction of application code, libraries, or additional resources in the application. After defining the application logic, the assailant will be able to find weaknesses to utilize in the model and stream of data.

Extraneous Functionality

This category is linked to the application of additional functionalities that are not needed or not planned in the mobile applications by the developer. Unintentional backdoor functions that are malicious or another internal development problem that could cause security abuse are included in this classification. Occasionally, extra code or features are left inside the source code that is not completely exposed to users. Creators may have left it to incorporate obscure backdoor functionality to backends or other internal expansion mechanisms that could be exploited.

Static Analysis by MobSF

The static analysis method is commonly used to evaluate the security level of mobile applications (Li et al., 2017). During static assessment, the application code will be analyzed to understand the application path to identify application elements. Automated tools such as Mobile Security Framework (MobSF) permit to automate this process. Static analysis generally means decompiling the APK file of a mobile application to its corresponding XML and Java files. Static analysis defines the technique employed to examine a mobile application without execution. The objective is to identify vulnerabilities and flaws in the application. Typically, automated tools are employed to ensure the procedure gets simpler. The said tools examine the code and go through the browsable activities, application permissions and more extra functions. This might, for instance, be confidential information that is hardcoded such as credentials or cryptographic keys (Lindström & Marstorp, 2018). To execute static analysis, a static analysis tool should include features to analyze XML with accuracy and correctness. Java files can be extracted by a de-compiler known as DEXtoJar (Skylot, 2019). A static analysis tool decompiles the computer code of an APK file to human being readable format for it to understand the code and detect the vulnerabilities that a mobile application could have. The plus points of static analysis include:

a) It is faster compared to the dynamic approach.
b) It involves fewer hardware sources compared to dynamic analysis.
c) The complete application code and manifest information which comprises of metadata about the application are thoroughly examined.

A significant number of weaknesses could be identified using static assessment like confidential data leakages, illegal access to important and confidential resources, and intent infusion. This can be also employed to discover misuse of authorization, power utilization, duplicate identification, and test generation. Lastly, it can also detect cryptographic application issues and code verification as well (Li et al., 2017). Mobile Security Framework

(MobSF) is an automated and open-source collective tool that could execute static as well as dynamic analysis on Android and iOS platforms. MobSF is commonly used by application developers and security researchers to carry out safety assessments on mobile applications (Papageorgiou et al., 2018). Moreover, previous research has indicated MobSF's ability to identify a comprehensive scale of Android security problems (Ranganath and Mitra 2020). MobSF is also a more user-friendly and easier-to-use static code analyzer based on a study comparing the strengths of multiple static analyzers available (Joseph, 2021).

MobSF is built based on Python MobSF offers a graphical user interface to upload the mobile application files and execute a thorough static analysis on the uploaded application file. It presents the assessment findings in the web interface portal, and it could additionally produce the output in JSON format over Rest API. MobSF separates all the metadata of the mobile application comprising the application title, the launcher activity, package title, minimum and maximum SDK, built name, and version code. It also traces the manifest alongside every activity, receiver, service, and provider for the mobile application. It executes numerous categories of security assessment such as examining the signer certificate, testing authorizations, manifest assessment, binary assessment, file assessment, code assessment, and malware assessment. Amongst the assessments that the tool offers, the manifest assessment and code assessment comprise the outcomes of the static evaluation associated with the majority of the safety flaws and weaknesses (Maharjan, 2020)

Research Methodology

This paper aims to carry out a static security analysis on five Malaysian government and five non - government mobile applications with a focus on only Android apps using a tool recommended in OWASP Testing Guide (OWASP, 2022). The methodology in this research starts by choosing the aimed mobile applications and gathering the APKs. Assessment will be carried out on the program code of the mobile application by static analysis approach which will be performed by utilizing a comprehensive security assessment tool. Once the examinations are finished, the outcomes will be documented and tabulated in the findings section and appropriate recommendations will be discussed.

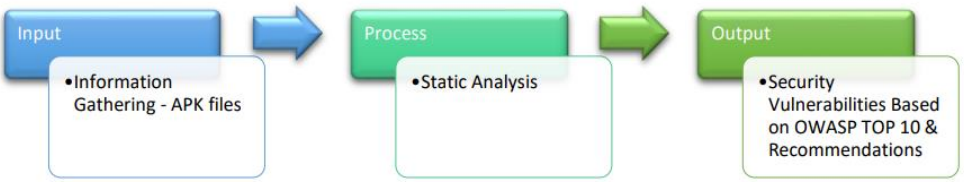

Figure 2. Static analysis process

Information Gathering

For this study, a total of ten mobile applications comprised of five Malaysian government and five non-government mobile applications available in Malaysia were chosen. The government mobile applications assessed in the experiment were chosen based on the Malaysian Government Mobile Application Gallery (GAMMA) Portal. GAMMA is an archive for mobile applications created and distributed by Malaysian Government organizations. The government of Malaysia has established GAMMA to make it simpler for Malaysians to download applications necessary for government-related matters. Table 1 describes every of

the selected mobile applications together with the built number utilized in the assessment, the category and the download count.

Table 1
Selected Applications

No	APP Name	Category	Developer	Version	Downloads
1.	MySejahtera	Health & Fitness	Government of Malaysia	2.0.6	10M+
2.	MyUBAT	Medical	Government of Malaysia	1.7.7	100K+
3.	GAMMA.my	Productivity	Government of Malaysia	1.0.11	10K+
4.	MyGov Portal	Social	Government of Malaysia	2.1	10K+
5.	Penghijauan Malaysia	Education	Government of Malaysia	1.2.1	5K+
6.	M2U MY	Finance	Malayan Banking Berhad	9.1	10M+
7.	TNG eWallet	Finance	TNG Digital Sdn Bhd	1.7.81	10M+
8.	i-Akaun	Finance	KWSP	6.6	5M+
9.	Pizza Hut Malaysia	Food	Pizza Hut Digital Venture	2.0.6	1M+
10.	myTNB	Business	Tenaga Nasional	2.6.7	1M+

Static Analysis

During static analysis, the mobile application's source code is examined and analyzed to make sure proper implementation of security controls is in place. The APK files of all the mobile applications were utilized to execute static analysis through the chosen tool to identify potential vulnerabilities in it. The chosen tool operates thorough evaluations of the mechanisms of the mobile application, tests for vulnerabilities, and examines the code to achieve complete code analysis. The code analysis tests for many kinds of security problems including hardcoded information, certificate issues and blacklisted malicious domains, and cryptography issues. A popular security assessment tool recommended in the OWASP Testing Guide (OWASP, 2022) will be primarily utilized in the evaluation and examination of this security assessment. Mobile Security Framework (MobSF) (Github, n.d.-a) is an automated application designed to undertake security assessments on mobile applications for the most popular mobile operating systems such as iOS and Android. The framework was created as a web service with a graphical user interface that features a dashboard to describe the findings.

This framework employs special features upon performing assessment and vulnerability analysis, such as mobile application component detection and configuration discovery such as activities and services. For the Android operating system, it performs Manifest file assessment, de-compilation of the source code, permission evaluation, and categorization. Toward the end of the static assessment, MobSF produces a report along with the detected vulnerabilities and a score known as the MobSF Security Score which is calculated based on the detected flaws. This paper focuses on those vulnerabilities that have a corresponding mapping in the OWASP Mobile Top 10 along with the calculated security scores.

After the execution of static analysis, the findings will be thoroughly analyzed and mapped using OWASP Mobile Top 10 together with recommendations. The Open Web Application Security Project (OWASP) is a non-commercial, worldwide, community-managed, open-source software project with many members and participants engaged together to enhance software safety measures. It functions as a resource for application creators and developers to safeguard mobile applications and web applications. The plan was originally published to assist in securing web applications from vulnerabilities. Alongside the development of innovative mobile applications, OWASP also published the mobile application edition comprising the topmost widespread mobile application threats. The current work includes all threats demonstrated in the OWASP listing in the most recent version released in 2016. The OWASP Mobile Top 10 (OWASP, 2016a) lists the security weaknesses and vulnerabilities developers should be informed of when developing mobile applications. OWASP Mobile's Top 10 list of security issues are grouped based on an international survey of security practitioners and application developers. The objective is to identify the key areas of concern in terms of mobile application security vulnerabilities.

Findings

The scoring results of static analysis on the government and non-government mobile applications based on MobSF Security Score are presented in Table 2 and Table 3. After MobSF performs static analysis on a mobile application, it produces a score representing its evaluation of application security known as the MobSF Security Score. The MobSF Security Score is the tool's specific grading procedure that analyses the application, and a security score will be given by taking into consideration the application signer certificate, authorizations, manifest assessment, and analysis of the application's source code.

Table 2
MobSF Security Score of Government Applications

Application	MobSF Security Score
MySejahtera	52/100
MyUBAT	51/100
GAMMA.my	45/100
MyGov Poral	55/100
Penghijauan Malaysia	52/100

Table 3
MobSF Security Score of Non-Governement Applications

Application	MobSF Security Score
M2U MY	55/100
TNG eWallet	55/100
i-Akaun	52/100
Pizza Hut Malaysia	56/100
myTNB	52/100

One of the most fascinating elements of the MobSF tool is the code analysis portion. In this section, it can be observed that MobSF has examined and analyzed the components and behaviors of the application based on best practices and baselines such as OWASP MSTG and plotted the weaknesses discovered with OWASP Mobile Top 10risks. The mobile application assessment is primarily based upon documentation and techniques enclosed in OWASP Mobile Top 10, which comprises ten known issues in mobile applications that are explored and frequently stumbled upon in the creation of mobile applications. The results from MobSF list the identified vulnerabilities within the targeted mobile application as seen in Table 4 where the issues discovered have been matched against OWASP Mobile Top 10. Table 4 summarizes the security vulnerabilities found during the static assessment of the mobile applications with MobSF.

Table 4
Summary of vulnerabilities detected by MobSF

Mobile Application	Category	Severity	Vulnerability	OWASP Top 10
MyUBAT	Government Application	High	The app uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is vulnerable to padding oracle attacks.	M5: Insufficient Cryptography
Penghijauan Malaysia	Government Application	High	The app uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is vulnerable to padding oracle attacks.	M5: Insufficient Cryptography
GAMMA.my	Government Application	Warning	The app can read/write to External Storage. Any App can read data written to External Storage.	M2: Insecure Data Storage
MyGov Portal	Government Application	Warning	The app can read/write to External Storage. Any App can read data written to External Storage.	M2: Insecure Data Storage
MyUBAT	Government Application	Warning	The app can read/write to External Storage. Any App can read data written to External Storage.	M2: Insecure Data Storage
Penghijauan Malaysia	Government Application	Warning	The app can read/write to External Storage. Any App can	M2: Insecure Data Storage

			read data written to External Storage	
MyUBAT	Government Application	Warning	The app creates temp file. Sensitive information should never be written into a temp file.	M2: Insecure Data Storage
Penghijauan Malaysia	Government Application	Warning	The app creates temp file. Sensitive information should never be written into a temp file.	M2: Insecure Data Storage
MyGov Portal	Government Application	Warning	The app uses SQLite Database and executes raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also, sensitive information should be encrypted and written to the database.	M7: Client Code Quality
MySejahtera	Government Application	Warning	The app uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database.	M7: Client Code Quality
MyUBAT	Government Application	Warning	App uses SQLite Database and execute raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also sensitive information should be encrypted and written to the database	M7: Client Code Quality
MyGov Portal	Government Application	Warning	Files may contain hardcoded sensitive information like usernames, passwords, keys etc	M9:Reverse Engineering
MySejahtera	Government Application	Warning	Files may contain hardcoded sensitive information like usernames, passwords, keys etc.	M9:Reverse Engineering
MyGov Portal	Government Application	Warning	Insecure WebView Implementation. Execution of user-controlled code in Web View is a critical Security Hole	M1: Improper Platform Usage
MyUBAT	Government Application	Warning	The App uses an insecure Random Number Generator.	M5: Insufficient Cryptography
Penghijauan Malaysia	Government Application	Warning	The App uses an insecure Random Number Generator	M5: Insufficient Cryptography
i-Akaun	Non-Government Application	High	The App uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is	M5: Insufficient Cryptography

			vulnerable to padding oracle attacks.	
M2U MY	Non-Government Application	High	The App uses the encryption mode CBC with PKCS5/PKCS7 padding. This configuration is vulnerable to padding oracle attacks.	M5: Insufficient Cryptography
i-Akaun	Non-Government Application	Warning	App can read/write to External Storage. Any App can read data written to External Storage	M2: Insecure Data Storage
M2U MY	Non-Government Application	Warning	App can read/write to External Storage. Any App can read data written to External Storage	M2: Insecure Data Storage
myTNB	Non-Government Application	Warning	App can read/write to External Storage. Any App can read data written to External Storage	M2: Insecure Data Storage
i-Akaun	Non-Government Application	Warning	App creates temp file. Sensitive information should never be written into a temp file	M2: Insecure Data Storage
M2U M	Non-Government Application	Warning	App creates temp file. Sensitive information should never be written into a temp file	M2: Insecure Data Storage
i-Akaun	Non-Government Application	Warning	App uses SQLite Database and executes raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also, sensitive information should be encrypted and written to the database.	M7: Client Code Quality
M2U MY	Non-Government Application	Warning	App uses SQLite Database and executes raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also, sensitive information should be encrypted and written to the database	M7: Client Code Quality
myTNB	Non-Government Application	Warning	App uses SQLite Database and executes raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection. Also, sensitive information should be encrypted and written to the database	M7: Client Code Quality
Pizza Hut Malaysia	Non-Government Application	Warning	App uses SQLite Database and executes raw SQL query. Untrusted user input in raw SQL queries can cause SQL Injection.	M7: Client Code Quality

			Also, sensitive information should be encrypted and written to the database	
i-Akaun	Non-Government Application	Warning	Files may contain hardcoded sensitive information like usernames, passwords, keys etc	M9: Reverse Engineering
M2U MY	Non-Government Application	Warning	Files may contain hardcoded sensitive information like usernames, passwords, keys etc	M9: Reverse Engineering
myTNB	Non-Government Application	Warning	Files may contain hardcoded sensitive information like usernames, passwords, keys etc	M9: Reverse Engineering
Pizza Hut Malaysia	Non-Government Application	Warning	Files may contain hardcoded sensitive information like usernames, passwords, keys etc	M9: Reverse Engineering
i-Akaun	Non-Government Application	Warning	Insecure WebView Implementation. Execution of user-controlled code in WebView is a critical Security Hole	M1: Improper Platform Usage
i-Akaun	Non-Government Application	Warning	SHA-1 is a weak hash known to have hash collisions.	M5: Insufficient Cryptography
M2U MY	Non-Government Application	Warning	SHA-1 is a weak hash known to have hash collisions.	M5: Insufficient Cryptography
Pizza Hut Malaysia	Non-Government Application	Warning	SHA-1 is a weak hash known to have hash collisions.	M5: Insufficient Cryptography
i-Akaun	Non-Government Application	Warning	The App uses an insecure Random Number Generator.	M5: Insufficient Cryptography
M2U MY	Non-Government Application	Warning	The App uses an insecure Random Number Generator.	M5: Insufficient Cryptography
myTNB	Non-Government Application	Warning	The App uses an insecure Random Number Generator.	M5: Insufficient Cryptography

From the vulnerabilities listed in Table 4, it can be observed that the detected issues can be categorized into 5 categories from the OWASP Mobile Top 10 as listed below.

M1: Improper Platform Usage

This classification includes the abuse of platform elements and the non-success to utilize platform protection mechanisms. From the perspective of this study, the assessment revealed that one application from both government and non-government applications is vulnerable to this weakness because they offer insecure WebView implementations. This weakness can weaponize numerous vulnerabilities, which vary according to the underlying vulnerable

function. In the existence of the stated vulnerability, a cybercriminal might feasibly strategize man-in-the-middle attacks and perform Cross-Site Scripting (XSS) attacks (Security Tips, n.d.).

M2: Insecure Data Storage

There are numerous techniques to examine insecure data storage, including assessing the security of log files, cookies, XML data, binary data, and SQL databases (SQLite) as well as internal and external storage, content service providers, and log files and external storage. Attackers and malicious programmes may exploit the saved data to obtain sensitive and private information, which may result in data misuse. Developers frequently utilize MODE_WORLD_READBALE & MODE_WORLD_WRITABLE to keep information in applications to be retrieved by another application and usually fail to utilize encryption for the safety of confidential data. From the standpoint of our research, the analysis revealed that 7 out of 10 mobile applications are vulnerable to this weakness. There are two findings discovered under this category. First, the program creates a temporary file and has access to external storage. Generally, a convenient way to store files is to utilize external file storage which normally presents a greater amount of disc space compared to internal storage. However, files created on the external storage are readable and writable which means a malicious application having the permissions WRITE_EXTERNAL_STORAGE or READ_EXTERNAL_STORAGE could possibly read sensitive data from the files that other applications have saved on the external storage. External storage could also possibly be removed by the user, making the files unobtainable to the application. The second finding that relates to the category is the application generates temporary files. It is advisable to never enter sensitive data into a temporary file. Generating and employing insecure temporary files can make application and system data vulnerable to attacks. The best practice is to prevent storing sensitive data in temporary files as it is a very popular reason for the majority of attacks on applications (Richard Lewis, 2006).

M5: Insufficient Cryptography

The use of cryptography on information or data utilized in a mobile application is enormously suggested so the information can be secured and will not be easily retrievable by attackers. Poor cryptography implementation will not be able to protect the data or confidential information since attackers can easily get the data or information. Examinations for cryptography are made for an explanation of weak cryptographic instances. In view of our research, the analysis revealed that 6 out of 10 mobile applications are vulnerable to this weakness. There are three findings discovered under this category. Firstly, the application utilizes the encryption method CBC along with PKCS5/PKCS7 padding. The padding oracle attack is susceptible to this arrangement. Padding is a general practice in cryptography that involves adding data at the start, central or end of a message before it is encrypted, being essential for aims of compatibility of the cryptographic algorithm that necessitates, for example, an exact number of characters or bytes for the right operational of the cryptographic function. Oracle Padding is an attack that decrypts messages and reveals, the padding validity state, permitting the attacker to retrieve sensitive information or escalate their privileges, without the need to get the key used in cryptographic operations (Team Sesame, 2017). The attack is frequently linked with the CBC (Cipher-Block Chaining) encryption mode that encrypts data merely in blocks of particular sizes, so it is essential to use padding for the proper functioning of its algorithm. The padding schemes generally used in these operations are PKCS5 and PKCS7. In the source code contained in the analyzed applications, it was found that CBC mode encryption is employed with PKCS5 or PKCS7 padding, creating a potential

Oracle Padding attack vector in the applications. Secondly, some applications use an insecure random number generator. When a pseudo-random number generator is used, an attacker can compute and anticipate the security-sensitive value that will be generated. Thirdly, some of the assessed mobile applications are also found to be signed with SHA-1 hashing algorithm. A weak hash known to have collisions is SHA-1 (Stevens et al., 2017). It is a legacy cryptographic hashing algorithm that is no longer considered secure. Digital certificates that use the SHA-1 hashing technique may make it possible for attackers to fake material, carry out phishing attacks, or launch man-in-the-middle assaults.

M7: Client Code Quality

A lack of quality control in the code may result in problems that prevent the programme from running properly or even at all. Applications that don't filter input allow malevolent users to modify the application process, execute services without authentication or authorization, and do anything they want. From the perspective of our research, the analysis revealed that 7 out of 10 mobile applications are vulnerable to this weakness. The affected mobile applications were found to be using SQLite Database and performing raw SQL queries. Raw SQL queries that contain untrusted user input may result in a local SQL injection vulnerability in the mobile application. The best strategy is to use prepared SQL statements that are outside of the user's control (OWASP, n.d.-b). Sensitive data should also be encrypted before being written to the database.

M9: Reverse Engineering

In the field of application development, reverse engineering is a technique frequently used to alter behavior and analyze source code. The mobile application's flow, the cryptography used, the application algorithms, the backend server information, and other essential information can all be gathered via reverse engineering techniques. From the perspective of our research, the analysis revealed that 6 out of 10 mobile applications are vulnerable to the weaknesses under this category. The impacted applications may have files with sensitive information such as usernames, passwords, keys, etc. hardcoded in them, according to the analysis of MobSF. Hardcoding sensitive data exposes it to the risk of being accessed by attackers, including usernames, passwords, server IP addresses, and encryption keys (Piątek, 2022). Anyone with access to the class files can find the sensitive data by decompiling them and using them in future attacks. Therefore, mobile applications must not hardcode sensitive information.

Conclusion

The market for mobile applications is growing, and security issues follow suit. Due to the sensitive user data that mobile applications manage, security is a primary issue for smartphone users and is no longer a voluntary subject. Information theft could lead to major issues like fraud and identity theft. In terms of mobile application security, the OWASP foundation has issued a list of the 10 most frequent hazards of mobile application platforms that emphasizes the security flaws & vulnerabilities developers must watch against while creating mobile applications.

In this paper, we have performed static analysis using a tool known as MobSF on government and non-government mobile applications to evaluate and compare the current security status of Malaysian government and non-government mobile applications based upon OWASP Mobile Top 10 risks. The reports that we obtained from the static analysis in

MobSF were further studied and the findings were presented. From the analysis result, we found that most of the tested mobile applications have at least one vulnerability listed in OWASP Mobile Top 10. Among the tested mobile applications, only one application from the non-government category was found to be free from vulnerabilities according to the categories of OWASP Mobile Top 10. The common issues that affect the mobile applications of both government and non-government were also identified through this study. Insecure Data Storage was found to be the biggest contributor to the vulnerabilities in government mobile applications and Insufficient Cryptography was found to be the main contributor to the vulnerabilities in non-government mobile applications.

Apart from that, the MobSF Security Score for each tested mobile application was identified, and the highest scorer was found to be Pizza Hut Malaysia from the non-government application and the least scored application was GAMMA.my from the government application. Many organizations tend to develop mobile applications internally which helps them to exponentially save costs without the need to pay external parties. This could be beneficial to the organization in saving costs but it is necessary to pay attention to the capability of the in-house developers to produce well-secured mobile applications. As most application development vendors must ensure proper security controls and policies are in place for application development, they tend to be more focused on the overall security of the application as well. This is an important part to consider when a mobile application is being developed. Proper security controls, security policies, and baselines can ensure the development of a secure mobile application. Understanding that nothing is truly secure and that security is never a given makes it essential to take precautions and stay as secure as you can.

In conclusion, this study conducted a static analysis using MobSF on Malaysian government and non-government mobile applications to evaluate their security status based on OWASP Mobile Top 10 risks. The findings revealed that most applications had at least one vulnerability, with Insecure Data Storage being the major issue for government apps and Insufficient Cryptography for non-government apps. Only one non-government app was free of vulnerabilities, while Pizza Hut Malaysia had the highest MobSF Security Score, and GAMMA.my scored the lowest. The study emphasized the importance of robust security practices, particularly for internally developed apps, to ensure a secure mobile environment. Future research could explore new methods to enhance mobile application security.

Acknowledgment
The authors would like to thank UNITAR International University for the publication of this research.

References

Alanda, A., Satria, D., Mooduto, H. A., & Kurniawan, B. (2020). Mobile application security penetration testing based on OWASP. IOP Conference Series: Materials Science and Engineering, 846, 012036. https://doi.org/10.1088/1757-899X/846/1/012036

Android Developers. (n.d.-a). Application Fundamentals. Retrieved from https://developer.android.com/guide/components/fundamentals.

Android Developers. (n.d.-b). Introduction to Activities. Retrieved from https://developer.android.com/guide/components/activities/intro-activities.

Android Developers. (n.d.-c). Services Overview. Retrieved from https://developer.android.com/guide/components/services.

Android Developers. (n.d.-d). Broadcasts Overview. Retrieved from https://developer.android.com/guide/components/broadcasts.

Android Developers. (n.d.-e). Content Providers. Retrieved from https://developer.android.com/guide/topics/providers/content-providers.

Android Developers. (n.d.-f). Intents and Intent Filters. Retrieved from https://developer.android.com/guide/components/intents-filters.

Android Runtime (ART) and Dalvik. (n.d.). Retrieved from https://source.android.com/docs/core/runtime.

Bassolé, D., Koala, G., Traoré, Y., & Sié, O. (2020). Vulnerability Analysis in Mobile Banking and Payment Applications on Android in African Countries. InterSol.

Bayern, M. (2019). 75% of developers worry about app security, but half lack dedicated security experts on their team. Retrieved from https://www.techrepublic.com/article/75-of-developers-worry-about-app-security-but-half-lack-dedicated-security-experts-on-their-team/

Benitez-Mejia, D. G. N., Sanchez-Perez, G., & Toscano-Medina, L. K. (2016). Android applications and security breach. In 2016 3rd International Conference on Digital Information Processing, Data Mining, and Wireless Communications, DIPDMWC 2016 (pp. 164-169). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/DIPDMWC.2016.7529383

Bertoglio, D. D., Girotto, G., & Neu, C. V. (2019). Pentest on an Internet mobile app: A case study using Tramonto. arXiv. https://doi.org/10.48550/arXiv.1912.09779

Buzzard, J. (2022). 2022 Identity Fraud Study: The Virtual Battleground. Retrieved from https://javelinstrategy.com/2022-Identity-fraud-scams-report.

Chanajitt, R., Viriyasitavat, W., & Choo, K.-K. R. (2016). Forensic analysis and security assessment of Android m-banking apps. Retrieved from https://gsec.hitb.org/materials/sg2016/COMMSEC%20D2%20-%20Rajchada%20Chanajitt%20-%20Forensic%20Analysis%20and%20Assessment%20of%20Android%20Banking%20Apps.pdf

Dehling, T., Gao, F., Schneider, S., Sunyaev, A. (2015). Exploring the Far Side of Mobile Health: Information Security and Privacy of Mobile Health Apps on iOS and Android. Retrieved from https://mhealth.jmir.org/2015/1/e8/PDF.

Ericsson. (2022). Ericsson mobility report. Retrieved from https://www.ericsson.com/49d3a0/assets/local/reports-papers/mobility-report/documents/2022/ericsson-mobility-report-june-2022.pdf

Filiol, E., & Irolla, P. (2015). (In)Security of mobile banking...and of other mobile apps. Retrieved from https://www.blackhat.com/docs/asia-15/materials/asia-15-Filiol-InSecurity-Of-Mobile-Banking-wp.pdf

Github. (n.d.-a). MobSF/Mobile-Security-Framework-MobSF. Retrieved from https://github.com/MobSF/MobileSecurity-Framework-MobSF

Github. (n.d.-b). Allsafe. Retrieved from https://github.com/t0thkr1s/allsafe.

Google. (n.d.). Android Releases: Android Developers. Retrieved from https://developer.android.com/about/versions/.

Hassan, M. A., Shukur, Z., & Mohd, M. (2022). A penetration testing on Malaysia popular e-wallets and m-banking apps. International Journal of Advanced Computer Science and Applications, 13, 10-15. https://doi.org/10.14569/IJACSA.2022.0130580

Hatamian, M., Wairimu, S., Momen, N., & Fritsch, L. (2021). A privacy and security analysis of early-deployed COVID-19 contact tracing Android apps. Empirical Software Engineering 26, (36). https://doi.org/10.1007/s10664-020-09934-4

Homeland Security. (2017). Study on Mobile Device Security. Retrieved from https://www.dhs.gov/sites/default/files/publications/DHS%20Study%20on%20Mobile%20Device%20Security%20-%20April%202017-FINAL.pdf. https://doi.org/10.1109/DIPDMWC.2016.7529383

Joseph, R. B., Zibran, M. F., & Eishita, F. Z. (2021). Choosing the weapon: A comparative study of security analyzers for Android applications. In *Proceedings of the 2021 IEEE/ACIS 19th International Conference on Software Engineering Research, Management and Applications (SERA)* (pp. 1-6). IEEE. https://doi.org/10.1109/SERA51205.2021.9509271

Knorr, K., & Aspinall, D. (2015). Security testing for Android mHealth apps. In 2015 IEEE Eighth International Conference on Software Testing, Verification and Validation Workshops (ICSTW) (pp. 18-24). https://doi.org/10.1109/ICSTW.2015.7107459

Knorr, K., Aspinall, D., & Wolters, M. (2015). On the privacy, security and safety of blood pressure and diabetes apps. In IFIP Advances in Information and Communication Technology (Vol. 459, pp. 503-516). https://doi.org/10.1007/978-3-319-18467-8_38

Kohli, N., & Mohaghegh, M. (2020). Security testing of Android-based COVID tracer applications. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9411579

Krombholz, K., Hobel, H., Donko Huber, M., & Weippl, E. (2014). Advanced social engineering attacks. Journal of Information Security and Applications, 19(2), 89-100. https://doi.org/10.1016/j.jisa.2014.09.005

Kumar, M. S. (2020). Driving SSDLC by adopting Mobile Security Analysis using MobSF. Retrieved from https://blogs.halodoc.io/ios-mobsf/.

Lee, A. (2018). A decade in, how safe are your iOS and Android apps? Retrieved from https://www.nowsecure.com/blog/2018/07/11/a-decade-in-how-safe-are-your-ios-and-android-apps/

Lewis, R. (2006). Security tips for temporary file usage in applications. Retrieved from https://www.codeproject.com/Articles/15956/Security-Tips-forTemporary-File-Usage-in-Applicat

Li, L., Bissyandé, T. F., Papadakis, M., Rasthofer, S., Bartel, A., Octeau, D., Klein, J., & Traon, L. (2017). Static analysis of Android apps: A systematic literature review. *Information and Software Technology, 88*, 67-95. https://doi.org/10.1016/j.infsof.2017.04.001

Lindström, H., & Marstorp, G. (2018). Security Testing of an OBD-II Connected IoT Device. Retrieved from http://autosec.se/wp-content/uploads/2018/05/MarstorpLindstrom-Security-Testing-of-an-OBD-II-Connected-IoT-Device.pdf.

Lockwood, A. (2012). Content providers and content resolvers. Retrieved from https://www.androiddesignpatterns.com/2012/06/content-resolvers-and-content-providers.html.

Maharjan, A. (2020). Ranking of Android apps based on security evidence. Retrieved from https://scholarworks.iupui.edu/bitstream/handle/1805/24775/Thesis.pdf?sequence=1&isAllowed=y

MITRE Corporation. (n.d.). Google Android: Vulnerability Statistics. Retrieved from https://www.cvedetails.com/product/19997/Google-Android.html?vendor_id=1224.

MobSF. (n.d.). Requirements. Retrieved from https://mobsf.github.io/docs/#/requirements.

Mokhtar, N. A. (2022). Hackers threaten to sell civil servants' personal data. The News Straits Times. Retrieved from https://www.nst.com.my/news/nation/2022/09/831916/hackers-threaten-sell-civil-servants-personal-data

Mukherjee, L. (2020). OWASP Mobile Top 10 Vulnerabilities & Mitigation Strategies. Retrieved from https://sectigostore.com/blog/owasp-mobile-top-10/.

New Straits Times. (2021). NST Leader: MyIdentity theft. Retrieved from https://www.nst.com.my/opinion/leaders/2021/09/732095/nst-leader-myidentitytheft.

Ng, B. (2021). Android Security Overview. Retrieved from https://medium.com/@boshng95/android-security-overview-7386022ad55d.

Nguyen-Vu, L., Chau, N.-T., Kang, S., & Jung, S. (2017). Android rooting: An arms race between evasion and detection. Security and Communication Networks. https://doi.org/10.1155/2017/4121765

Nilsson, R. (2020). Penetration testing of Android applications (Dissertation). Retrieved from http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-280290

OWASP. (2016a). OWASP Mobile Top 10. Retrieved from https://owasp.org/www-project-mobile-top-10/.

OWASP. (2016b). M1: Improper Platform Usage. Retrieved from https://owasp.org/www-project-mobile-top-10/2016-risks/m1-improper-platform-usage.

OWASP. (2016c). M4: Insecure Authentication. Retrieved from https://owasp.org/wwwproject-mobile-top-10/2016-risks/m4-insecure-authentication.

OWASP. (2016d). M5: Insufficient Cryptography. Retrieved from https://owasp.org/www-project-mobile-top-10/2016-risks/m5-insufficient-cryptography.

OWASP. (2016e). M6: Insecure Authorization. Retrieved from https://owasp.org/wwwproject-mobile-top-10/2016-risks/m6-insecure-authorization.

OWASP. (2016f). M7: Poor Code Quality. Retrieved from https://owasp.org/www-project-mobile-top-10/2016-risks/m7-client-code-quality.

OWASP. (2016g). M9: Reverse Engineering. Retrieved from https://owasp.org/wwwproject-mobile-top-10/2016-risks/m9-reverse-engineering.

OWASP. (2022). OWASP Mobile Application Security Testing Guide (MASTG) v1.5.0. Retrieved from https://github.com/OWASP/owasp-mastg/releases/latest/download/OWASP_MASTG-v1.5.0.pdf.

OWASP. (n.d.-b). SQL Injection Prevention Cheat Sheet. Retrieved from https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.htm

Papageorgiou, A., Strigkos, M., Politou, E., Alepis, E., Solanas, A., Patsakis, C. (2018). Security and privacy analysis of mobile health applications: the alarming state of practice. Retrieved from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8272037.

PewResearch.Org. (2021). Mobile Fact Sheet. Retrieved from https://www.pewresearch.org/internet/fact-sheet/mobile/.

Piątek, K. (2022). Hard-coded Tokens, Keys and Credentials in Mobile Apps. Retrieved from https://www.netguru.com/blog/hardcoded-keys-storage-mobile-app.

Positive Technologies. (2019). Vulnerabilities and threats in mobile applications, 2019. Retrieved from https://www.ptsecurity.com/ww-en/analytics/mobile-applicationsecurity-threats-and-vulnerabilities-2019/

Reaves, B., Bowers, J., Scaife, N., Bates, A., Bhartiya, A., Traynor, P., & Butler, K. R. (2017). Mo(bile) Money, Mo(bile) Problems: Analysis of Branchless Banking Applications in the Developing World. Retrieved from https://www.cise.ufl.edu/~traynor/papers/reaves-usenix15a.pdf.

Reed, B. (2022). Test of 250 Popular Android Mobile Apps Reveals that 70% Leak Sensitive Personal Data. Retrieved from https://www.nowsecure.com/blog/2019/06/06/test-of-250-popular-android-mobile-apps-reveal-that70-leak-sensitive-personal-data/.

Ruth, C. (2021). Over 60% of Android apps have security vulnerabilities. Retrieved from https://atlasvpn.com/blog/over-60-of-android-apps-have-security-vulnerabilities.

Sai, A., Buckley Lero, J., & Le Gear, A. (2019). Privacy and security analysis of cryptocurrency mobile applications. In Proceedings of the 2019 Fifth Conference on Mobile and Secure Services (MobiSecServ) (pp. 1-6).

Security Tips. (n.d.). Android Developers. Retrieved from https://developer.android.com/training/articles/security-tips#WebView.

Sharma, T., & Bashir, M. (2020). An analysis of phishing emails and how the human vulnerabilities are exploited. In Proceedings of the 11th International Conference on Applied Human Factors and Ergonomics (AHFE 2020), San Diego, CA.

Skylot. (2019). Retrieved from https://github.com/skylot/jadx.

Statista. (2019). Share of global smartphone shipments by operating systems from 2014 to 2023. Retrieved from https://www.statista.com/statistics/272307/market-shareforecast-for-smartphone-operating-systems/

Statista. (2022a). Mobile operating systems' market share worldwide from January 2012 to August 2022. Retrieved from https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/.

Statista. (2022b). Number of available applications in the Google Play Store from December 2009 to March 2022. Retrieved from https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

Stevens, M., Bursztein, E., Karpman, P., Albertini, A., & Markov, Y. (2017). The first collision for full SHA-1. In J. Katz & H. Shacham (Eds.), Advances in Cryptology – CRYPTO 2017 (Vol. 10401, pp. 1-22). Springer, Cham. https://doi.org/10.1007/978-3-319-63688-7_19

Team Sesame. (2017). Padding Oracle Attacks. Retrieved from https://tlseminar.github.io/padding-oracle/.

Vermeulen, R. (2019). Security evaluation of glucose monitoring applications for Android smartphones. Retrieved from https://www.os3.nl/_media/2018-2019/courses/rp1/p41_report.pdf.

Yang, W., Zhang, Y., Li, J., Liu, H., Wang, Q., Zhang, Y., & Gu, D. (2017). Show me the money! Finding flawed implementations of third-party in-app payment in Android apps. In NDSS Symposium 2017.

Ngui, Y. (2021). Malaysia's Covid-19 App Reports 'Malicious Script' Misuse.